GUILLERMO AYALA GALLEGO

BIOINFORMÁTICA ESTADÍSTICA

ESTADÍSTICA DE DATOS ÓMICOS

UNIVERSITAT DE VALÈNCIA

Copyright ©24 de mayo de 2025
Guillermo Ayala
Guillermo.Ayala@uv.es
This work is free. You can redistribute it and/or modify it under the terms of the Do What The Fuck You Want To Public License, Version 2, as published by Sam Hocevar. See http://www.wtfpl.net/ for more details.

Índice general

I Datos **1**

1 Estadística y datos ómicos **3**
1.1 Introducción 3
1.2 Estructura de los datos 3
1.3 Problemas estadísticos 5
1.4 Sobre herramientas online 5
1.5 Paquetes transversales 6
1.6 Jerga 6
1.7 Datos ordenados 6
1.8 Bibliografía 7

2 Microarrays **9**
2.1 Introducción 9
2.2 Affymetrix GeneChip 9
2.3 Otros métodos de normalización 27
2.4 Datos en línea 27
2.5 Sobre cómo usar un ExpressionSet 27
2.6 De cómo construir un ExpressionSet 32
2.7 Ejemplos 36
2.8 Correspondencias múltiples 48
2.9 Ejercicios 50

3 RNA-seq **53**
3.1 Introducción 53
3.2 Flujo de trabajo con RNA-seq 53
3.3 Repositorios 55
3.4 Formatos 55
3.5 Alineamiento 57
3.6 Una opción 60
3.7 Control de calidad 62
3.8 Ejemplos 65
3.9 Normalización de RNA-seq 70

4 Metilación **77**
4.1 Datos y preprocesado 78

II Análisis marginal **81**

5 Expresión diferencial marginal **83**
5.1 Introducción 83
5.2 Algo de notación 83
5.3 Selección no específica o filtrado independiente 84

5.4 Fold-change 89
5.5 Expresión diferencial marginal 90
5.6 Ejercicios 97

6 Modelos lineales 99
6.1 Sobre lo que vamos a tratar 99
6.2 Regresión lineal simple 104
6.3 Análisis de la varianza con un factor fijo 110
6.4 Mínimos cuadrados 114
6.5 Modelos lineales normales 118
6.6 Muchos modelos 123
6.7 rowFtests 125

7 Modelos lineales generalizados 127
7.1 Componentes de un modelo lineal generalizado 128
7.2 Verosimilitud, ajuste y distribución asintótica de los GLMs 130
7.3 Bondad de ajuste 132
7.4 Regresión logística 134
7.5 Datos de conteo 142

8 Comparaciones múltiples 149
8.1 Introducción 149
8.2 Control fuerte y control débil del error 152
8.3 Relación entre las tasas de error tipo I 153
8.4 p valores y p valores ajustados 154
8.5 Métodos que controlan la FWER 154
8.6 Métodos que controlan el FDR 157
8.7 Utilizando genefilter y p.adjust 157
8.8 El q-valor 158
8.9 Inferencia selectiva 162
8.10 Ejercicios 162

9 Expresión diferencial con respuesta continua 165
9.1 Método SAM 165
9.2 Limma 170
9.3 Ejercicios 189

10 Expresión diferencial con datos de conteo 191
10.1 Introducción 191
10.2 Una muestra por condición 191
10.3 edgeR 193
10.4 SAMseq 208
10.5 DESeq2 211
10.6 Ejercicios 216

III Análisis de grupos de genes 217

11 Grupos de genes 219
11.1 Introducción 219
11.2 Homo sapiens 220
11.3 Grupos con levadura 222
11.4 Grupos para GSE1397 224
11.5 Grupos utilizando anotación 225
11.6 Utilizando EnrichmentBrowser 226

11.7 Utilizando DOSE 227
11.8 Ejercicios 228

12 Test de Fisher unilateral 229
12.1 Test de Fisher 229
12.2 Sobre la elección del universo de genes 231
12.3 Utilizando Category y GOstats 232
12.4 EnrichmentBrowser::sbea 238
12.5 ORA con clusterProfiler 239
12.6 tamidata::gse21942 con tami::ora 240
12.7 Ejercicios 241

13 Análisis de conjuntos de genes 243
13.1 Introduction 243
13.2 Sobre la distribución de la matriz de expresión 244
13.3 Conjuntos de genes 244
13.4 Ejemplos 245
13.5 Cuantificando asociación gen-fenotipo 247
13.6 Enriqueciendo el conjunto de genes 247
13.7 Distribuciones condicionadas a los datos 247
13.8 Usando Limma 250
13.9 GSA 252
13.10 GSEA: Gene set enrichment analysis 255
13.11 Estudiando interacciones entre genes individuales y grupos de genes 259
13.12 Un método simple pero efectivo 259
13.13 Un método basado en poblaciones finitas 260
13.14 Análisis de grupos de genes por muestra 261
13.15 gse21942 263
13.16 PRJNA297664 264
13.17 gse21942 265
13.18 PRJNA297664 266
13.19 Material complementario 266

IV Estadística 267

14 Conceptos fundamentales de Estadística 269
14.1 Verosimilitud 269
14.2 Estimación 271
14.3 Estimador máximo verosímil 273
14.4 Contraste de hipótesis 275

15 Datos categóricos 279
15.1 Inferencia con la distribución binomial 279
15.2 Inferencia para la multinomial 281
15.3 Probabilidad y tablas de contingencia 283
15.4 Comparación de dos proporciones 285
15.5 Inferencia en tablas de contingencia 287

16 Componentes principales **297**
16.1 Introducción 297
16.2 Definición 297
16.3 Componentes principales 299
16.4 Componentes principales de los datos golub 303
16.5 ¿Muestras o genes? 304
16.6 ¿Tipificamos los datos? 305
16.7 Ejemplos 305

17 Análisis cluster **309**
17.1 Introducción 309
17.2 Datos 310
17.3 Disimilaridades 312
17.4 Disimilaridades entre grupos de observaciones 315
17.5 Cluster jerárquico 316
17.6 Distancia cofenética 318
17.7 Métodos de particionamiento 318
17.8 Silueta 321
17.9 Análisis cluster y datos de expresión de gen 322
17.10 Ejemplos 323
17.11 Otras aproximaciones y problemas 326
17.12 Ejercicios 326

18 Miscelánea **329**
18.1 Algoritmo bipesado de Tukey de un solo paso 329
18.2 Median polish 330
18.3 Datos faltantes 331
18.4 Dibujo de la media-diferencia de Tukey o dibujo de Bland-Altman 332

V Investigación reproducible **335**

19 Investigacion reproducible y repetible **337**
19.1 Documento dinámico 338
19.2 LaTeX 338
19.3 Markdown 339
19.4 Pandoc 339
19.5 knitr 339
19.6 RMarkdown 339
19.7 Quarto 339
19.8 Entornos de desarrollo 340

20 Generando un informe **341**
20.1 Generando la información 342
20.2 Generando un informe en html 344
20.3 Generación de enlaces 345
20.4 Ejercicios 346

21 Lo que se hace porque sí **347**
21.1 ¿Variables u observaciones? ¿Qué estoy analizando? . . . 347
21.2 Mejor no usarlo 348
21.3 Combinando información de distintos experimentos . . . 348
21.4 Yo con la Excel me apaño 349
21.5 Datos de conteo y t-test 350

21.6 A la búsqueda desesperada del p-valor 350

VI R/Bioconductor 353

22 Todo lo que siempre quiso saber sobre R pero nunca se atrevió a preguntar 355
22.1 Sobre la instalación de R 355
22.2 Paquetes 356
22.3 Lo primero con R 356
22.4 Una sola muestra 357
22.5 Varias muestras 360
22.6 Análisis de los datos golub 362
22.7 La función apply 366
22.8 Listas 369
22.9 Funciones con R 370
22.10 Pasar argumentos a un script de R por línea de comandos 373
22.11 tamitasks 373
22.12 Actualizar paquetes 377

23 Bioconductor 379

24 Anotación 381
24.1 AnnotationDbi 382
24.2 ChipDb 385
24.3 OrgDb 387
24.4 TxDb 389
24.5 BSgenome 394
24.6 OrganismDb 395
24.7 biomaRt 396
24.8 KEGGREST 398
24.9 Tareas habituales con anotaciones 398
24.10 Ejercicios 400

A Código sin más 401
A.1 GSE198668 401
A.2 gse80200 402
A.3 gse21443 403
A.4 bcrneg 404
A.5 gse183019 404

Glosario 423

Glossary 427

Prólogo

Este texto se ocupa de la aplicación de técnicas estadística al análisis de datos ómicos. Se ocupa de la aplicación sin olvidar la teoría que fundamenta los métodos. La Probabilidad y la Estadística es un edificio y no una colección de herramientas. Es imprescindible la comprensión de los conceptos. Con frecuencia, el usuario de software lo usa sin saber que está usando. Se preocupa de que el software funciona aunque no se sepa qué hace y si es lo que pretendemos hacer.

Un investigador que publique un artículo al mes durante un año tendrá al final de ese año un total de 12 publicaciones. Después de ese año maravilloso de publicaciones, el siguiente es tan bueno como el anterior. Y sigue publicando un trabajo al mes durante este segundo año. Ya reune 24 publicaciones en dos años agotadores. Y esto mismo lo hace durante 10 años.[1] Al final de esta decada prodigiosa tendrá 120 publicaciones. Ya ha justificado su vida como investigador. Pero no, incansable sigue otros diez años y alcanza al cabo de 20 años los 240 publicaciones. Y diez años más. Exhausto él o ella (y cuantos hayan intentado leer todas estas publicaciones) ya ha llegado a las 360 publicaciones. Hay una enorme cantidad de investigadores que superan esta cantidad. La superan ampliamente. Por amor de Dios, no más. Es seguro (con probabilidad uno) que han participado activamente en todas estas publicaciones. Pero el resto de los humanos no tenemos la culpa. Podríamos llamarlo un 12/30. De hecho hay 12/40 y mucho más. Hay numerosos estudios sobre el crecimiento del número de publicaciones. Sin entrar en precisiones innecesarias, en el momento actual, se tarda unos nueve años en doblar el número de publicaciones.[2] Evidentemente el planeta no tiene un problema de superpoblación de personas. Es de publicaciones científicas. Con frecuencia leemos cómo muchas publicaciones han de retirarse.[1] Nunca es culpa de nadie. Nunca es cierto que los autores habían apostado por lo rápido. Por lo fácil. Por no controlar el trabajo. Por no limitarse a copiar el tratamiento de datos de otros sin preocuparse si era adecuado en su caso. O si lo entendían. O controlar el trabajo del precario, mal pagado y sufrido becario/a que echa demasiadas horas y todo es nuevo (y bueno) para él/ella. Necesitaban algunas decenas (centenares) de publicaciones más para que las generaciones futuras los/las recuerden. Y mucho que los recordarán. Sé que los recordarán. Sobre la evolución de las revistas científicas es interesante https://www.eldiario.es/sociedad/neoliberalismo-burocracia-robert-maxwell-revistas-cientificas-primaron-negocio_1_9952229.html.

Y debiera de ser candidato premios importantes.

[1] Asumimos que no es un superhéroe con algún superpoder producido por una pequeña araña radioactiva o algún rayo cósmico por salir sin paraguas a la calle.

[2] Un optimista diría que doblamos el conocimiento que la humanidad ha logrado atesorar cada nueve años.

En campos como las Matemáticas o la Estadística **teórica** el produc-

[1] https://retractionwatch.com/ https://www.abc.es/sociedad/abci-retirada-ultima-investigacion-reciente-premio-nobel-202001050137_noticia.html, https://elpais.com/elpais/2019/01/27/ciencia/1548629779_450088.html, https://elpais.com/elpais/2019/10/14/ciencia/1571052466_871787.html, https://www.theguardian.com/world/2020/jun/03/covid-19-surgisphere-who-world-health-organization-hydroxychloroquine?.

to está a la vista. Un teorema lleva su prueba y si hay un fallo puede ser visto y corregido por la comunidad de expertos en ese tema en algún momento posterior. Sin embargo, en campos como la Biología o la Medicina las cosas no son así. No es tan fácil comprobar la reproducibilidad de los resultados. En [14] se analizan los resultados de una encuesta a 1576 investigadores. La proporción de trabajos que no se pueden reproducir es enorme. Los factores que se comentan como importantes son los siguientes de mayor a menor frecuencia.

1. No incluir una descripción completa del trabajo.
2. Presión por publicar.
3. Trabajos sin un estudio de potencia previo y un análisis pobre de los resultados.
4. Insuficiente replicación en el laboratorio original.
5. Insuficiente supervisión.
6. La metodología y el código informático utilizado no disponible.
7. Un diseño experimental pobre o inadecuado.
8. Los datos originales no disponibles.
9. Fraude.
10. Una revisión por parte de la revista insuficiente.

Con el análisis estadístico de los datos tienen que ver los puntos 3, 6, 7 y 8. Hay que hacer bien las cosas desde el principio.

En muchas ocasiones biólogos o médicos o biotecnólogos o bioquímicos o . . . me han indicado que querían aprender más Probabilidad y Estadística. Normalmente después de que les has analizado unos datos para una tesis o un artículo. Al principio, con interés, les recomiendas algún texto esperando que les ayude. El final de la historia suele ser el mismo. Que no leen nunca el libro. Aprender Probabilidad y Estadística es muy fácil. Se coge un libro sencillo de Probabilidad y luego otro de Estadística. Cada libro lleva una página que pone el número uno. Se lee esa página. Se le da la vuelta y se lee la siguiente numerada con el 2. Y así sucesivamente. Y no hay otra.

Estas notas tratan de la aplicación de procedimientos estadísticos al **análisis de datos de alto rendimiento**. Bonito el nombre pero: ¿qué son datos de alto rendimiento? Datos que rompen lo que tradicionalmente era un prerrequisito en Estadística (multivariante). Muchos procedimientos estadísticos empiezan indicando que el número de observaciones, n, ha de ser mayor que el número de variables por observación, p. Actualmente es frecuente (mejor habitual) que los datos no los recoja un experimentador con un lápiz y un papel y luego los introduzca con mucho trabajo en una hoja de cálculo. Lo hacen disposivos electrónicos conectados con ordenadores. Por eso los datos tienen dimensiones p que marean: miles de variables frente a decenas (con suerte algo más de un centenar) de observaciones o muestras. ¿Y qué hacemos para analizar esto? Lo que se pueda.[3] De esto van estas notas, **de lo que se pueda**. Son unas notas en progreso. Se van añadiendo ideas, técnicas, software y se ve cómo incorporar estos análisis en nuestro trabajo. También[4] voy incorporando nuevos tipos de información. De momento, analizamos datos de

[3] Uno no está obligado más que a hacer las cosas lo mejor que pueda y no más.

[4] En la medida en que me voy encontrando con nuevos tipos de datos que entienda.

expresión de gen (o expresión génica) sabiendo que (la mayor parte de) lo que hacemos es aplicable en otros contextos. El tema § 2 está dedicado a este tipo de datos, sus características y técnicas de preprocesado. Es el tema específico de esta información. Si se sustituye este capítulo por otro dedicado a otra técnica de adquisición de información casi todo lo que sigue es aplicable.

Seguir el material de este manual supone unos conocimientos biológicos fundamentalmente de genética molecular. Conocimientos muy básicos para un especialista en campos como Biología, Biotecnología, Bioquímica o Medicina pero no tan presentes en personas con una formación en otros estudios no cercanos a la Biología. Para este segundo de grupo es recomendable leer alguna introducción breve y simple. En particular, recomiendo [81, chapter 1].

El objetivo es que estas notas ayuden a quien lee y no que demuestren que sabe el que las escribe.[5] En lo biológico hay imprecisión por ignorancia, en lo probabilístico y estadístico hay imprecisión porque pretenden ser unas notas para un público interesado en estos temas pero con formación en Biología, Biotecnología, Bioquímica o Medicina fundamentalmente. Hay que tener algún interés en Probabilidad, Estadística e Informática para poder seguirlas en su totalidad. Todo en el curso está basado en R y Bioconductor.

[5] Sobre esta cuestión yo diría que que el autor se apaña como puede (que no es mucho) con los conceptos biológicos.

Se han de leer artículos de revistas de diversos ámbitos científicos. Como matemático/estadístico es decepcionante intentar entender la metodología que se ha aplicado cuando lees trabajos de revistas de gran prestigio en Biología. Muchos trabajos tienden a ser una pura ilustración de los métodos propuestos. El cómo se hace realmente se relega al final en una sección suplementaria o directamente a un archivo de material suplementario. Dónde se coloca da lo mismo: No da lo mismo que no se incluya o que se incluya sin el detalle necesario. Hay una auténtica aversión, miedo, odio a la formulación matemática de las cosas. El lenguaje matemático se ha desarrollado para ser preciso. No se puede expresar todo simplemente con palabras. De hecho, a veces, es imposible saber qué se está haciendo porque no quieren poner una simple formula. Si la revista de Biología publica un trabajo metodológico ha de asumir que tiene que dar una formulación precisa y eso pasa por incluir una presentación con lenguaje matemático. De lo contrario, el artículo es inútil aunque el índice de impacto de la revista sea mayor.

En los datos que vamos a analizar lo más frecuente en que se tenga una muestra de tejido y se aplique alguna técnica ómica. Sin embargo, y cada vez más, se puede analizar cada célula individualmente. Mientras que en el caso en que la muestra se refiere al tejido el interés es comparar los datos observados entre condiciones biológicas distintas, cuando la información llega al detalle de la célula, además de poder seguir comparando los valores entre condiciones, se abre la posibilidad de clasificar la célula en distintos tipos de celulas (un problema de clasificación). Con datos a nivel de célula también permite la comparación entre condiciones biológicas. Obviamente es una información desagregada.

Estas notas utilizan sistemáticamente R/Bioconductor y hacemos algún uso de Bash y Python. Un buen punto de inicio para ver las posibilidades de R/Bioconductor en análisis estadístico de datos ómicos es https://cran.r-project.org/web/views/Omics.html.

Cuando empecé con esto de la Probabilidad y la Estadística uno leía libros y artículos, entendías aquello y luego intentabas descifrar cómo aplicaba (mejor implementaba) estas técnicas el software comercial (en

mi caso SPSS). Lo primero era bonito. Los autores intentan que les entiendas porque tratan de transmitir ideas. Sin embargo, el software comercial no piensa así (y esto no es necesariamente malo). El software comercial intenta dar un producto bueno y fácilmente utilizable para llegar a un máximo de usuarios. Solamente suelen considerar temas sobre los que hay mucho interés y muchos (potenciales) usuarios. Y esto es correcto. No es la opción elegida en estas notas. Hemos elegido trabajar con software libre. Tanto R como Bioconductor son el resultado de un gran trabajo coordinado de muchas personas.

¿ES R/Bioconductor la única opción? Por supuesto que no. Y tampoco tengo muy claro que sea la mejor. Indudablemente es buena. En estas notas nos centramos en el análisis estadístico de datos ómicos. Y desde el punto de vista de la Estadística sí que podemos afirmar que, en el momento de escribir, R es la mejor opción para análisis estadístico de datos y Bioconductor es su opción natural cuando nuestro interés es en datos ómicos.[6]

[6] Sin embargo, en http://neurolex.org/wiki/Category:Resource:Gene_Ontology_Tools (en concreto buscamos dentro de la página la expresión "Statistical analysis") tenemos una buena muestra de qué podemos hacer y sobre todo con qué hacerlo.

Es muy recomendable consultar http://www.bioconductor.org/help/workflows/ en donde podemos encontrar análisis completos de datos.

Es ingente la cantidad de publicaciones científicas que llevan tratamientos estadísticos. Los investigadores suelen saber qué quieren estudiar. Diseñan un experimento y observan datos. Lo que se hace después no lo suelen conocer. No suelen conocer las técnicas que han utilizado.[7] Citan los procedimientos estadísticos y no indican el software utilizado si está disponible en la red o, en el caso de que sea propio de los autores, dónde se puede conseguir. Sin duda alguna el control de la calidad de los tratamientos estadísticos descansa en que cada lector de una publicación científica tenga a su disposición el artículo (que básicamente es la explicación de lo que se ha hecho) así como los datos [8] y **todo** el código necesario para reproducir **todo** el tratamiento estadístico realizado con los datos. Se repite todo porque en muchas ocasiones lo que se ha descartado puede ser tan interesante como lo que se ha publicado. Este es el conocido sesgo a publicar los resultados significativos descartando los *no* significativos. Sin esto, no se puede realizar un control adecuado de un tratamiento estadístico de datos de alto rendimiento (de hecho, de ningún tipo de datos). Esto nos lleva a los conceptos de programación literaria o comentada (literate programming) propuesto por Donald K. Knuth[9] y, de un modo más genérico, a la investigación reproducible [10]. R/Bioconductor incorpora muchas herramientas para realizar investigación reproducible.[11] En particular, este texto está realizado utilizando [159, knitr]. Todos los datos que se utilizan están disponibles en bases de datos públicas como (sobre todo) GEO o ArrayExpress. No de todos los datos que se utilizados tenemos los datos sin procesado previo. Los usamos por haber sido analizados en otros textos o en ejemplos de R/Bioconductor o, simplemente, porque son bonitos de estudiar. Recientemente Nature Genetics ha refrendado el uso de Bioconductor.

[7] Y eso es un punto que no tiene solución.

[8] Los datos sin ningún tipo de preprocesamiento. Por ejemplo, si tenemos datos de expresión con Affymetric GeneChip se debiera de disponer de los ficheros .CEL.

[9] http://en.wikipedia.org/wiki/Literate_programming

[10] http://en.wikipedia.org/wiki/Reproducibility

[11] http://cran.r-project.org/web/views/ReproducibleResearch.html

Hemos de poder realizar un análisis de datos y generar un informe de un modo sencillo. Esto excluye el uso de herramientas como Word, Excel o similares. En este texto utilizaremos la opción R/Markdown/Quarto.

Se intenta explicar Estadística aplicada a la Bioinformática. Por ello es un texto que combina ambas cosas. En ocasiones se utiliza R/Bioconductor como herramienta pedagógica para ilustrar un concepto. En otras casi hacemos de manual técnico para usar un paquete de R. En principio, la idea es que vamos leyendo y ejecutando el código que se inserta. Lo lógico es tener R funcionando y, con un copiar y pegar, podemos ir

ejecutando el código.[12]

En http://www.uv.es/ayala/docencia/tami/AllTami.R tenemos un script para instalar todos los paquetes que se utilizan en estas notas.[13] En https://www.uv.es/ayala/docencia/tami/#instalación se muestra cómo instalar los paquetes de R y Bioconductor y de algunos paquetes en Debian/Ubuntu necesarios para seguir el manual.

A lo largo de las notas utilizamos muchas funciones que corresponden a paquetes distintos. Indicaremos el nombre de la función y su paquete conjuntamente. Por ejemplo, con `annotate::annotation()` estamos refiriéndonos a la función `annotation` que se encuentra en el paquete [60, annotate].

En la parte I mostramos los datos con los que trabajamos posteriormente. Se comenta el tipo de dato, dónde se puede conseguir en bases de datos públicas y cómo preprocesarlo (normalizarlo). Finalmente se ve con cierto detalle los bancos de datos que utilizamos en el resto del libro. Los datos procesados y preparados para hacer análisis estadístico los tenemos en los paquetes [**R-tamidata2**, 12, 13, tamidata]. [14] En el momento actual consideramos datos de expresión obtenidos con microarrays de DNA y con la técnica RNA-Seq.[15]

En la parte II se trata el problema de analizar cada característica de interés marginalmente. Constituye un repaso de técnicas de Estadística. También se aborda el problema de las comparaciones múltiples así como técnicas de expresión diferencial en microarrays y RNA-Seq.

En la parte III se estudia análisis de grupos de genes: cómo obtener las colecciones de grupos de genes, análisis de sobre representación y análisis de grupo de genes (o enriquecimiento).

En V se habla de investigación reproducible.

Los aspectos relativos específicamente a R o Bioconductor son tratados en la parte VI.

Finalmente se incluyen apéndices sobre matrices, Probabilidad más avanzada, métodos numéricos y código que se referencian en el resto del texto.

Asociado a este manual tenemos la página https://www.uv.es/ayala/docencia/tami/ en donde se indica cómo instalar los paquetes necesarios. La versión más actualizada de este manual está en https://www.uv.es/ayala/docencia/tami/tami13.pdf. He desarrollado algunos paquetes de apoyo: el paquete [11, tami], https://www.uv.es/ayala/software/tami_1.0.tar.gz, pretende que se pueda realizar un análisis de expresión diferencial marginal con dos grupos a comparar con un mínimo de aprendizaje. Los paquetes [**R-tamidata2**, 12, 13] contienen los datos que utilizamos en los ejemplos.

Es un material que se utiliza en tres cursos claramente diferenciados.

TAMI Uno es una breve introducción en 20 horas lectivas en tercer curso del grado de Bioctecnología de la Universidad de Valencia. En este curso se utilizan las herramientas más básicas, la opción simple y rápida. Se pretende ver cosas interesantes y mantener (en lo que se pueda) la programación con R/Bioconductor simple. Fue el origen de estas notas.

Estadística de datos ómicos Es una asignatura obligatoria de tercer curso del grado de Ciencia de Datos. Se utiliza un nivel básico con énfasis en los aspectos más computacionales y estadísticos.

Bioinformática Estadística El segundo curso es un módulo en Bioin-

[12] Solamente tener en cuenta que los caminos hay que modificarlos adaptándolos a donde tengamos los datos en nuestro ordenador.

[13] Quizás no es necesario instalarlo de una vez. Lo más conveniente es instalar los paquetes en la medida en que los necesitemos en los distintos capítulos.

[14] Son paquetes propios alojados en la Universidad de Valencia.

[15] Aunque se pretende ampliar a otros datos ómicos.

formática Estadística en el master de Bioinformática de la Universidad de Valencia. Utilizamos todo el material.

Parte I

Datos

Capítulo 1

Estadística y datos ómicos

1.1 Introducción

A diferentes procedimientos de obtención de información en Biología se les ha dado en llamar ómicas. El propio nombre como algo que lo engloba todo, que lo observa todo, es ya de por sí algo pretencioso. Las personas que vivan en los siglos que han de venir (y gracias a Dios no veremos) si son compasivos sonreirán ante esta denominación. Dato ómico es un tipo particular de dato de alta dimensión. Estas notas se ocupan de revisar y aplicar técnicas estadísticas a datos ómicos. No estamos interesados en el detalle exhaustivo del tratamiento de cada tipo de dato. Nos centraremos en lo que tienen en común y comentaremos lo que diferencia su tratamiento.

El énfasis de este texto es sobre **métodos estadísticos**. Evitaremos en lo posible métodos que no utilicen modelos probabilísticos. Lo que se ha dado en llamar métodos de **machine learning** o **Big Data**.[16] En este sentido es interesante recordar una frase (que subscribo) de un gran estadístico inglés Brian D. Ripley.

[16] He puesto a propósito los términos en inglés. Queda moderno. El autor no es un gran seguidor de esas técnicas nuevas. Soy un desfasado que solo cree en la Probabilidad. Sin modelo probabilístico no hay mucho. No hay nada.

```
fortunes::fortune(50)
```

```
To paraphrase provocatively, 'machine learning is
statistics minus any checking of models and
assumptions'.
  -- Brian D. Ripley (about the difference
     between machine learning and statistics)
     useR! 2004, Vienna (May 2004)
```

1.2 Estructura de los datos

En lo que sigue tendremos distintos tipos de datos con una estructura similar. Observaremos un gran número de características sobre un pequeño número de muestras. Por tanto tendremos muestras de dimensión alta.

Las características que observamos son de distintos tipo: fluorescencia cuando trabajemos con distintos tipos de microarrays como microarrays de DNA, metilación o microarrays de proteínas; número de lecturas alineadas cuando hablamos de procedimientos de secuenciación; presencia o ausencia de una mutación cuando estudiemos asociación.[17]

[17] Y otros que el autor desconoce.

Esta característica puede estar asociada a una sonda o a un grupo de sondas en un microarray. O bien la información corresponde a un gen o a un exon, o a un péptido, a una proteína, o a una región genómica.

La característica con la que trabajamos en cada momento se cuantifica con distintos procedimientos (con frecuencia dependientes del fabricante del dispositivo). Hablaremos de características sin más. El número lo denotamos por N donde este valor es grande (miles). Estas características las observaremos en unas pocas muestras. El número de muestras es n (decenas con suerte). Lo básico[18] es que N es mucho mayor que n: $n \ll N$. En un contexto estadístico clásico N es el número de variables y n es el número de muestras. Y justo lo conveniente es la situación contraria. De hecho, muchos procedimientos estadísticos suponen que el tamaño de la muestra supera el número de variables. En Estadística de alta dimensión[19] esto no es así. Y eso da novedad a los procedimientos. Obviamente limita las posibilidades pero abre un nuevo campo de trabajo.

[18] Y a lo largo de las notas lo repetiremos muchas veces.

[19] High dimensional statistics.

Las características las recogemos en una matriz[20] que denotaremos por

$$\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$$

donde el valor y_{ij} nos cuantifica la característica i en la muestra j. [21]

Si y_{ij} corresponde a DNA microarray entonces mide un nivel de fluorescencia y tomará valores positivos.[22] Sea positivo a no un valor mayor indicará una mayor expresión del gen. Si trabajamos con datos obtenidos con RNA-Seq entonces tendremos conteos, esto es, número de lecturas cortas alineadas sobre un gen o sobre un exon o sobre una zona genómica de interés. En definitiva el dato primario es un número entero. Más lecturas indicará más expresión otra vez. Los valores observados en una misma fila (una misma características sobre todas las muestras) se suele decir que son, en transcriptómica, *perfil*[23] (de un modo genérico perfil de expresión).

[20] Que podemos llamar matriz de expresión o de metilación o de conteos o de mutaciones, aunque no necesariamente hablamos de expresión de un gen pero no parece malo utilizar esta nomenclatura de un modo genérico.

[21] Una matriz de datos en Estadística suele ser justo al contrario, esto es, con los subíndices invertidos. Es decir, la matriz transpuesta de la que vamos a utilizar aquí. En ocasiones se utiliza en la forma clásica pero es más frecuente esta forma de disponer los datos.

[22] Algunos procedimientos de procesado previo de los datos conocidos como normalización pueden dar lugar a expresiones negativas.

[23] Expression profile.

En la matriz $\boldsymbol{y}$ los valores observados para las distintas muestras son *independientes* aunque posiblemente observados bajo distintas condiciones. No son pues réplicas de una misma condición experimental pero sí se observan independientemente. Son condicionalmente independientes. Sin embargo, las filas de $\boldsymbol{y}$ son realizaciones de vectores dependientes. Por ejemplo, los valores de expresión para las distintas filas no son independientes en una matriz de expresión ya que los genes actúan de un modo coordinado.

Habitualmente los datos de las columnas de la matriz $\boldsymbol{x}$ no son directamente comparables. Hay muchos artefactos técnicos así como ruido en la observación de la característica de interés. Se han desarrollado técnicas para corregirlo que llamaremos **preprocesado**. Veremos algunos procedimientos de preprocesado. En sentido estricto cuando realizamos estos los datos estos dejan de ser independientes. Sin embargo, esto no se suele considerar en la literatura. Los datos después de la normalización siguen considerándose independientes por columnas (muestras) y dependientes por filas.

De cada muestra tendremos información. Por ejemplo, si es una muestra control o bien corresponde a una muestra tomada bajo un tratamiento. [24] A esta información o variables que nos describen a las muestras las llamaremos los *metadatos* o *variables fenotípicas*[25] Usualmente tendremos varias variables fenotípicas. Denotaremos por $x = (x_1, \ldots, x_n)$ los valores observados de una variable en las n muestras. El caso maś frecuente de variable fenotípica será cuando tengamos dos grupos de

[24] La palabra tratamiento se utiliza en el sentido amplio de diseño experimental: tiempo, una cepa salvaje frente a una mutada por ejemplo.

[25] Entendido en un sentido amplio: Por variable fenotípica podemos estar observando el tiempo en que se ha observado la evolución de una

muestras (casos y controles). En este caso tendremos $x_i = 1$ si es un caso e $x_i = 0$ si es un control.[26] Si tenemos más de dos grupos de muestras a comparar, por ejemplo k grupos, entonces $x_i \in \{1, \ldots, k\}$ para $i = 1, \ldots, n$.

[26] Los valores 0 y 1 son arbitrarios. Podemos tomar cualquier otro par de valores.

1.3 Problemas estadísticos

¿Y qué vamos a hacer con la matriz $\boldsymbol{y}$ y con las variables fenotípicas? Como siempre lo mejor que podamos. A veces no mucho. Las técnicas estadísticas que se utilizan son aplicaciones de procedimientos diseñados en muchas ocasiones para el contexto habitual en que tienes más muestra que variables. Y se usan aquí adaptándolos con mayor o menor fortuna. Yo diría con mayor o menos sentido en ocasiones.

Un problema fundamental que abordaremos es el que se conoce como **expresión diferencial**. Nos fijamos en una variable fenotipica y en una característica (gen por ejemplo). ¿Hay asociación entre el perfil de expresion y la variable fenotípica? Veremos distintos procedimientos para responder esta pregunta *para cada característica*. Utilizaremos la denominación de *análisis de expresión diferencial marginal*. Sin embargo, en la literatura biológica es más hablar de *análisis de expresión diferencial gen-a-gen*.

Las distintas características son dependientes entre si. Una evaluación de la posible asociación entre cada característica y la variable fenotípica es *limitada*. Nos puede hacer perder información sobre procesos biológicos de los cuales distintas características nos están dando información de modo que cada una de ellas recoge una variación no muy grande pero que conjuntamente es notable. En definitiva, el problema será estudiar la posible asociación entre grupos de características (grupos de filas en la matriz de expresión) y la variable fenotípica de interés. Es lo que se conoce como análisis de grupos de genes.[27]

[27] Este problema puede encontrarse bajo distintas denominaciones como: Gene set analysis, gene set enrichment analysis, functional enrichment testing.

También se verán problemas de reducción de dimensión. En concreto la técnica más utilizada, análisis de componentes principales. Es una técnica instrumental con muchas posibilidades de utilización en un contexto como es este con datos de alta dimensión.

Y clasificaremos tanto las características como las muestras como una herramienta exploratoria. Lo que se conoce como análisis cluster.

En lo que sigue se aborda también problemas de cálculo del tamaño muestral que está ligado al problema de la potencia del test.

Y todo esto siempre atendiendo al tipo de dato que estemos utilizando. Sin duda, el más desarrollado son las datos de expresión de gen utilizando microarrays. Será nuestro tipo de dato de referencia pero vamos introduciendo otros tipos de datos como pueden ser RNASeq o datos de abundancia de proteínas.

1.4 Sobre herramientas online

En este texto la opción elegida es R/Bioconductor. Es obvio que requiere un mayor conocimiento de los procedimientos que se utilizan pero nos dan una enorme flexibilidad para trabajar. Podemos preprocesar la información y procesarla como queramos. Obviamente hay que saber cómo hacerlo. Y de esto va el texto. Sin embargo, cada vez más hay herramientas en línea[28] que merecen explorarse (aunque no sea nuestra

[28] Online en La vida moderna.

opción). En lo que sigue mostramos una enumeración con algún comentario.[29] Algunas de estas aplicaciones son:

[29] No tengo un gran conocimiento de su manejo porque insisto no es mi opción personal. Las manejo para ver qué ofrecen pero siempre encuentras que harías las cosas de otra manera.

GEPIA http://gepia.cancer-pku.cn/index.html Es una buena implementación para estudios de cáncer. Permite el análisis tanto de gen a gen (o marginal) como el análisis de grupos de genes. Todo orientado al estudio de los cánceres. Los datos no los proporciona el usuario. Realmente utilizan datos de los proyectos TCGA y GTEx.

NetworkAnalyst http://www.networkanalyst.ca Es de particular interés la tabla 1 en [157, pág. 824] en donde se muestra una comparativa entre distintas herramientas online.

DAVID http://david.abcc.ncifcrf.gov.

g:Profiler http://biit.cs.ut.ee/gprofiler/.

InnateDB con Cytoscape http://www.innatedb.ca y http://www.cytoscape.org.

Gitools http://www.gitools.org.

1.5 Paquetes transversales

Algunos paquetes que vamos a manejar pretenden cubrir una gran cantidad de contenidos que vamos a tratar: lectura de la información generando las clases adecuadas (como Biobase::ExpressionSet o SummarizedExperiment::RangedSummarizedExperiment); normalización; análisis de expresión diferencial marginal o gen a gen; análisis de enriquecimiento para grupos o bien para grafos. En fin, tienen una pretensión de análisis global. Usualmente lo que hacen es utilizar distintos paquetes según realiza una u otra funcionalidad. Un buen ejemplo es [51, EnrichmentBrowser]. Los usaremos aunque siempre suponen una limitación en el análisis que se realiza. Soy más partidario de utilizar funcionalidades concretas del paquete más que utilizarlo para todo el análisis.

1.6 Jerga

En un campo como es la Bioinformática un problema mayor es el lenguaje que se utiliza. Un texto de Estadística o un texto de Genética tiene una jerga consolidada. Son campos de trabajo maduros.[30] Prácticamente cada campo de investigación va desarrollando una jerga[31] en donde suele haber una parte interesada. Impedir el acceso a los que no son de este campo. Aquí el problema es grave porque se mezclan jergas al mismo tiempo que se desarrolla una nueva. Para intentar evitar esta barrera he incluido un glosario al final del texto que intenta aclarar los términos y sirva de referencia rápida. Sin embargo, quizás la mayor dificultad de esta disciplina sea este, entender la jerga de los demás.[32]

[30] Sin ser peyorativo, campos de investigación viejos.

[31] Mejor una jerigonza.

[32] En mi caso de los biólogos. A los informáticos los entiendo. Hablan menos de hecho.

1.7 Datos ordenados

Un problema que comparte el análisis de datos ómicos con cualquier otro problema de análisis de datos es tener unos datos de entrada *ordenados*[33] así como unas salidas de las distintas funciones que los analizan

[33] Tidy data.

que sean ordenadas, fácilmente interpretables y que puedan ser una entrada ordenada para un procedimiento posterior. El término utilizado en [149] es **tidy data**. En este texto utilizaremos distintas herramientas que prestan atención a este aspecto del análisis que ahorra mucho tiempo de trabajo y muchísimos quebraderos de cabeza. Entre otros paquetes utilizaremos [15, biobroom] y [122, broom].

1.8 Bibliografía

A lo largo del manual se presta mucha atención a citar con precisión las referencias originales del material. La comprensión precisa de las técnicas supone la consulta de la referencia original.[34] Un libro que trata cómo hacer las cosas con R/Bioconductor pero no lo que se hace o porqué se hace es [127]. Es una guía de uso muy bien elaborada. Un texto con un objetivo similar al nuestro es [64]. Un manual online que sigue una línea muy similar a este es http://genomicsclass.github.io/book/.

[34] Hay una peligrosa tendencia a creer que leyendo un resumen se conoce la técnica. No es cierto. Las referencias bibliográficas siempre hay que consultarlas.

Capítulo 2

Microarrays

2.1 Introducción

En este tema tratamos sobre datos de expresión obtenidos utilizando microarrays.[1] Nos ocupamos del procesado (o mejor, del preprocesado) de los datos desde los datos originales (o datos a nivel de sonda) hasta los datos tal y como los hemos estado analizando. En este tema nos ocupamos de este punto previo y no menor. Muchos de los análisis de expresión diferencial posterior dependen de un modo esencial de lo que hacemos antes. Con frecuencia el experimentador confía en el preprocesado que el fabricante del chip realiza. No necesariamente este preprocesado tiene porqué estar mal hecho pero en cualquier caso es preciso conocerlo y evaluarlo. En lo que sigue veremos como hay funciones que reproducen lo que el fabricante hace. En este capítulo es de interés los tres primeros capítulos de [54].

La expresión de un gen es el proceso mediante el que se transcribe el DNA en una serie de copias de mRNA. Los microarrays miden la cantidad de mRNA para cada gen. El valor de la expresión de un gen es una medida de luminiscencia relacionada con el mRNA presente. Esto es para un chip de DNA.

2.2 Affymetrix GeneChip

Es conveniente consultar [153, pag. 93 y siguientes]. Vamos a leer datos obtenidos mediante un escáner Affymetrix GeneChip. Estos arrays de oligonucleótidos tienen un pequeño número de sondas para un mismo gen. Hablaremos de conjunto de sondas refiriéndonos a todas las sondas que en un mismo chip están asociadas con un mismo gen. Cada sonda son pequeñas celdas con oligonucleótidos específicos de un gen. Estos oligonucleótidos tienen 25 pares de bases. Cada oligonucleótido es una parte (aproximadamente) específica de un gen. En principio solo el mRNA de este gen lo reconocerá como propio. Sin embargo, también puede ocurrir que el RNA de otros genes encuentre partes comunes con la sonda y se adjunte. Tendríamos una hibridación no específica. Con objeto de considerar el problema Affymetrix modifica la secuencia original. Concretamente el par de bases en la posición 13. De este modo una hibridación con esta sonda se considera un acoplamiento erróneo.

[1] La presentación asociada es https://www.uv.es/ayala/docencia/tami/presentaciones/t2_MicroarrayDNA_p.html.

Las imágenes digitales con la información original son los ficheros DAT. Una vez procesados por el software que proporciona el fabricante obtenemos un resumen por celda que los guarda en los ficheros CEL. Este fichero es el resultado del escaneo y la segmentación de la imagen obtenida. Todavía no hemos resumido estas intensidades de los píxeles en un valor por gen. En estos ficheros tenemos la media y desviación estándar de los niveles de gris así como la localización de la sonda dentro del array. Se necesita también conocer la correspondencia entre sondas y nombres de los genes. Esta información aparece en el fichero CDF.[2]

La información que se maneja se puede considerar a tres niveles:

Imagen El primer nivel es de la imagen digital (ficheros DAT en Affymetrix GeneChip). Hay que utilizar técnicas de procesado de imagen digital para obtener a partir de esta imagen otra imagen donde cada pixel tiene como nivel de gris la expresión de la sonda en esa celda.

Datos a nivel de sonda Tenemos un imagen digital donde un pixel contiene una cuantificación de la cantidad de hibridación.

Datos de expresión Ya tenemos corregido el fondo y normalizadas las muestras. Tenemos una matriz de expresión de modo que en la fila tenemos el gen y en la columna la muestra. A partir de aquí se trabaja para analizar la posible actividad diferenciada de un gen o un conjunto de genes.

En las secciones que siguen vamos a utilizar los datos con número de acceso GEO GSE21779 § 2.7.5.

2.2.1 Sonda y conjunto de sondas

En estos microarrays tenemos distintas sondas correspondientes al mismo gen. ¿Qué chip o plataforma se ha utilizado se ha utilizado?

Podemos verlo con `Biobase::annotation`.

```
pacman::p_load(Biobase,affy)
annotation(gse21779raw)
```

```
[1] "hgu133plus2"
```

El número de sondas y de muestras lo podemos conocer con

```
dim(exprs(gse21779raw))
```

```
[1] 1354896 18
```

Nos fijamos en el primer array cuyas expresiones aparecen en la primera columna. Podemos ver las intensidades del primer microarray en forma de una imagen a niveles de gris. Cuando más blanco es el punto mayor es la intensidad observada en la sonda y por lo tanto mayor la expresión del gen al que está asociada. Cada pixel corresponde con una sonda distinta. La imagen la tenemos en la figura 2.1.

```
png(paste0(dirTamiFigures,"microarray4.png"))
image(gse21779raw[,1])
dev.off()
```

```
pdf
  2
```

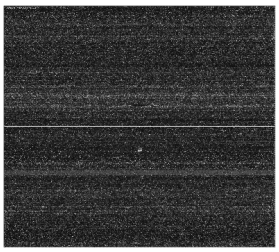


Figura 2.1: Imagen a niveles de gris correspondiente al primer array de los datos gse21779.

Podemos conocer las identificadores que Affymetrix le ha asignado a estas sondas con

```
head(probeNames(gse21779raw))
```

```
[1] "1007_s_at" "1007_s_at" "1007_s_at"
[4] "1007_s_at" "1007_s_at" "1007_s_at"
```

Nos fijamos en la sonda que aparece en la posición 400.

```
ID = probeNames(gse21779raw)[400]
```

El número de sondas que componen el conjunto asociado a este identificador es

```
sum(probeNames(gse21779raw) == ID)
```

```
[1] 11
```

Podemos ver los cardinales de todos los conjuntos de sondas que lleva el chip.

```
counts = table(probeNames(gse21779raw))
table(counts)
```

```
counts
   8 9 10 11 13 14 15 16
   5 1 6 54130 4 4 2 482
  20 69
  40 1
```

2.2.2 Variabilidad intra y entre arrays

Dentro de todo el vector con los nombres de las sondas buscamos las que ocupa la primera tanto para PM como para MM (de ahí el argumento `both`).

```
indexProbes(gse21779raw,"both",ID)[[1]]
```

```
 [1] 1088751 883561 1054203 366753 178855
 [6] 68793 490689 62456 1123802 939573
[11] 1349651 1089915 884725 1055367 367917
[16] 180019 69957 491853 63620 1124966
[21] 940737 1350815
```

¿Cuáles corresponden a PM?

```
(posiciones = indexProbes(gse21779raw,"pm",ID)[[1]])
```

```
 [1] 1088751 883561 1054203 366753 178855
 [6] 68793 490689 62456 1123802 939573
[11] 1349651
```

Tenemos 11 correspondientes a PM. Y las demás corresponden a MM:

```
indexProbes(gse21779raw,"mm",ID)[[1]]
```

```
 [1] 1089915 884725 1055367 367917 180019
 [6] 69957 491853 63620 1124966 940737
[11] 1350815
```

En figura 2.2(a) vemos dónde está localizado el conjunto de sondas[35] dentro de la imagen. En cada localización tenemos un valor de PM y otro valor MM.

[35] Probe set.

Vamos a representar los niveles de expresión PM correspondientes a un mismo conjunto de sondas de un array dado (figura 2.2(b)).

```
library(ggplot2)
pmID = probes(gse21779raw[,1],"pm",ID)
df1 = data.frame(sondas = 1:11,intensidad = pmID[,"GSM542488.CEL.gz"])
ggplot(df1,aes(x=sondas,y=intensidad))+geom_line()
```

En abscisas tenemos el número de la sonda (de un mismo conjunto) y en ordenadas su expresión. Si no hubiera ningún tipo de ruido lo que se debe observar sería una línea horizontal, unos mismos niveles de expresión. Pero no es así.

El dibujo 2.2(b) se refiere a la variabilidad **intra array** (utilizando las sondas de un mismo conjunto) para un mismo gen. Vamos superponer (una línea por muestra) las expresiones del mismo gen pero para todas las muestras. De este modo ilustramos la variabilidad entre muestras. En la figura 2.2(c) mostramos este dibujo. En el código es interesante el uso de la función `reshape::melt()` para obtener el `data.frame`.

```
pm0 = probes(gse21779raw,"pm",ID)
pm0 = data.frame(pm0,sondas=1:11)
pm1 = reshape::melt(pm0,id="sondas")
ggplot2::ggplot(data=pm1,aes(x=sondas,y=value,colour=variable))+
    geom_line()+ylab("Intensidad")
```

```
pm0 = probes(gse21779raw,"pm",ID)
pm0 = data.frame(pm0,sondas=1:11)
pm1 = reshape::melt(pm0,id="sondas")
p = ggplot(data=pm1,aes(x=sondas,y=value,colour=variable))+
    geom_line()+ylab("Intensidad")
ggsave(paste(dirTamiFigures,"gse217791arrays.png"),p)
```

En la figura 2.2(c) cada línea corresponde a un array y los puntos que representamos en cada línea es un conjunto de sondas. Este conjunto de sondas es el mismo que elegimos en todos los arrays.

¿Qué significa este dibujo? ¿Cómo lo podemos interpretar? Si las distintas sondas asociadas al mismo gen nos dieran una misma expresión cada una de las líneas debía ser horizontal. No es así como hemos visto para la primera y para todas las demás. Tenemos ahora una segunda fuente de variación. Estas muestras se han tomado bajo tres condiciones distintas. Si no hubiera diferencia entre condiciones las líneas debieran de estar muy superpuestas.

De acuerdo al diseño de este chip cabe esperar que en una misma localizacion el valor PM sea mayor que el valor de MM. Esto es lo esperable ya que PM es señal, estamos observando la hibridación específica (aquella para la que la sonda ha sido diseñada). El valor de MM lo que mide es ruido óptico e hibridación no específica, en resumen, ruido en un sentido amplio. La señal debe dominar al ruido, el valor de PM debiera de ser mayor que el valor de MM en cada celda. Esto sobre el papel. Pero no en la realidad. Veamos (figura 2.2(d)) para un mismo array y para un conjunto de genes dados los valores de PM y los valores MM.

```
library(reshape)
pmm = data.frame(sondas=1:11,
```

[2] Cuando leamos datos veremos que si no tenemos cargado el fichero CDF necesario el programa los baja sin que se lo pidamos.

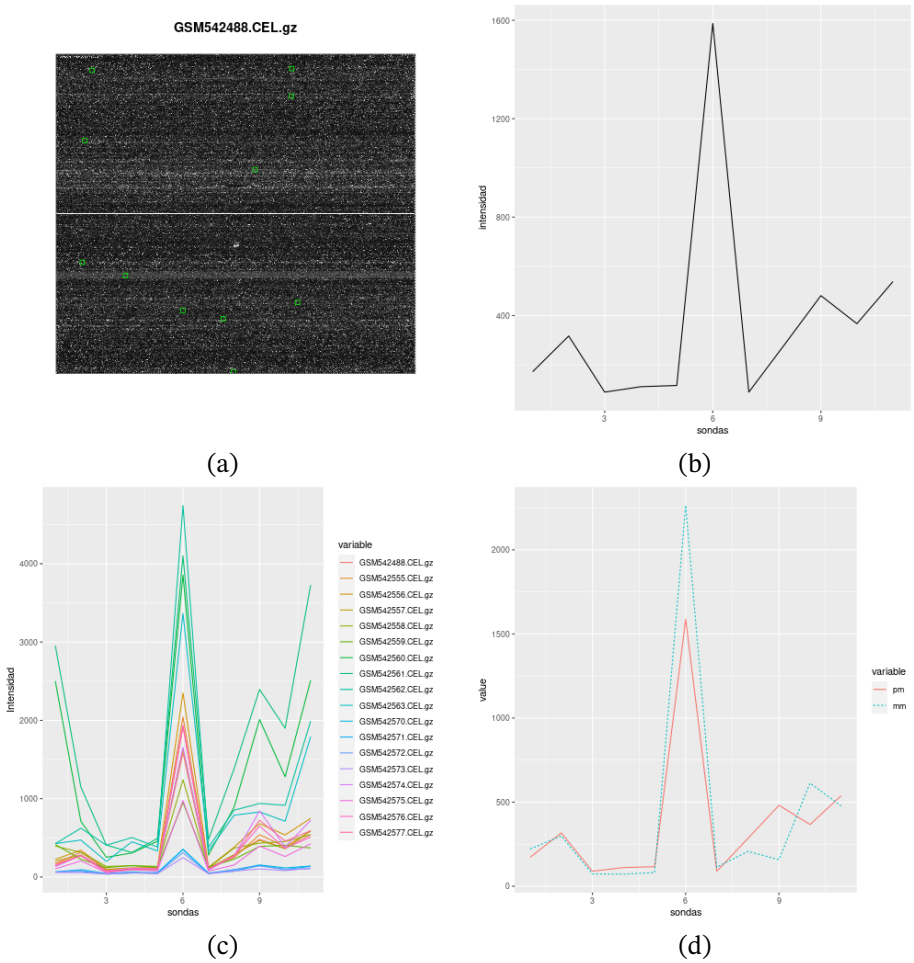


Figura 2.2: (a) Imagen CEL correspondiente al primer array. Cada pixel corresponde con una sonda distinta. Los cuadrados verdes nos localizan un conjunto de sondas asociadas al mismo gen. (b) Los valores (o intensidades) PM para un grupo de sondas en el primer array de nuestras muestras. (c) Los valores (o intensidades) PM para un mismo grupo de sondas en los distintos arrays. Cada línea corresponde con un array distinto. En abscisas tenemos la misma sonda en cada uno de los arrays. (d) En la primera muestra de gse21779 mostramos los valores PM y, en las mismas localizaciones, los valores MM. No siempre PM es mayor que MM como sería de esperar.

```
                    pm = probes(gse21779raw[,1],"pm",ID),
                    mm = probes(gse21779raw[,1],"mm",ID))
df2 = reshape::melt(pmm,id="sondas")
levels(df2["variable"]) = c("pm","mm")
ggplot(df2,aes(x=sondas,y=value,colour=variable,
                linetype=variable)) + geom_line()
```

Podemos hacer una valoración global para todos los arrays que tenemos. ¿Qué proporción de sondas son tales que el valor PM es menor que el correspondiente valor MM?

```
table(probes(gse21779raw,"pm") ≥probes(gse21779raw,"mm"))
```

```
  FALSE TRUE
4460757 6415887
```

Vemos que tenemos un 41% en este caso particular. No es poco.

2.2.3 Control de calidad

Vamos a estudiar distintos procedimientos que intentan evaluar los distintos microarrays que componen nuestro experimento. El objetivo es decidir si es necesario descartar total o parcialmente alguna de las muestras.

2.2.4 MA plots

Estos dibujos sirven para determinar si alguno de los microarrays debe de ser descartado del estudio y si necesitamos homogeneizar los valores para las distintas muestras. Son una aplicación del dibujo media-diferencia propuesto por Tukey. Es conveniente consultar § 18.4 y http://en.wikipedia.org/wiki/MA_plot.

Trabajamos, en lugar de con las intensidades originales observadas, con su logaritmo en base 2. Supongamos que u_i y v_i denotan las intensidades originales en la sonda i-ésima. Consideramos $x_i = \log_2 u_i$ e $y_i = \log_2 v_i$ y estos son los valores a comparar. En este contexto se habla de dibujos MA ya que

$$m_i = x_i - y_i = \log_2 \frac{u_i}{v_i},$$

y

$$a_i = \frac{1}{2}(x_i + y_i) = \frac{1}{2}\log_2 u_i v_i = \log_2 \sqrt{u_i v_i}.$$

Una primera opción es comparar todos los pares de microarrays utilizando el dibujo media-diferencia de Tukey.

Una segunda opción consiste en considerar una especie de microarray *típico* y comparar los demás con este. Por ejemplo, una opción que implementa el paquete [74, affy] en la función `affy::MAplot()` [36] consiste en calcular para cada sonda la mediana a lo largo de todos los microarrays que componen nuestro experimento. Por tanto, la mediana de las intensidades será x_i y el valor y_i corresponde a cada uno de los microarrays de nuestro experimento.

[36] La viñeta MAplots en el paquete [21] se dedica al uso de esta función.

Vamos a ilustrar el uso de los dibujo MA con los datos **gse21779**. En la figura 2.3 podemos ver una comparación de los dos primeros arrays.

```
MAplot(gse21779raw,pairs=TRUE,which=1:2,
        plot.method="smoothScatter")
```

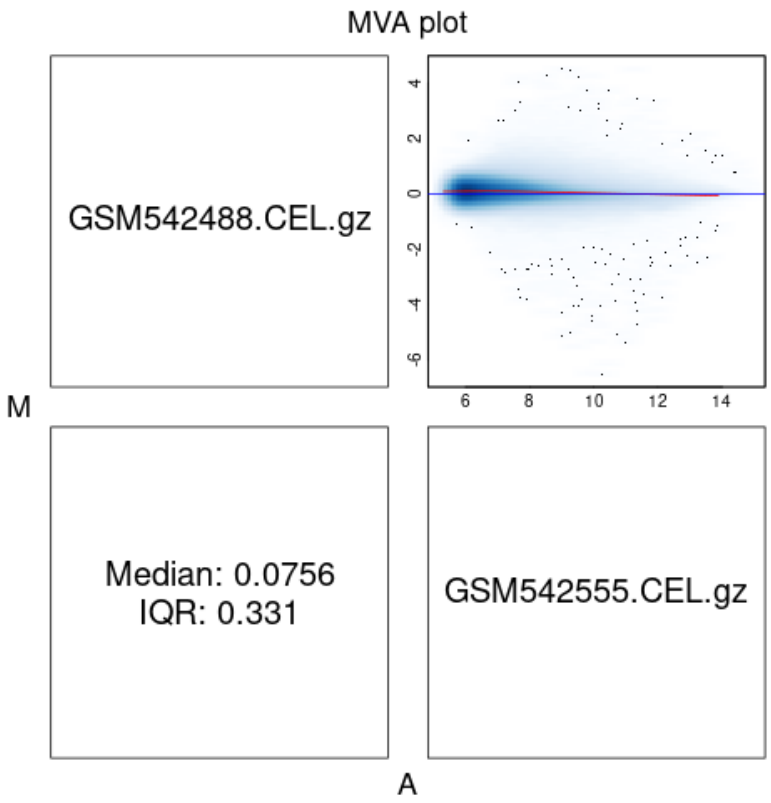


Figura 2.3: MAplot comparando los dos primeros microarrays.

Podemos obtener la comparación de todos los arrays con el de referencia.

```
MAplot(gse21779raw,pairs=FALSE,
        plot.method="smoothScatter")
```

En particular, si queremos solamente el primero de ellos lo tenemos (figura 2.4) con

```
MAplot(gse21779raw,pairs=FALSE,which=1,plot.method="smoothScatter")
```

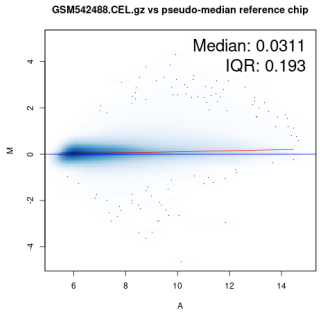


Figura 2.4: MAplot el primer array del experimento con el de referencia.

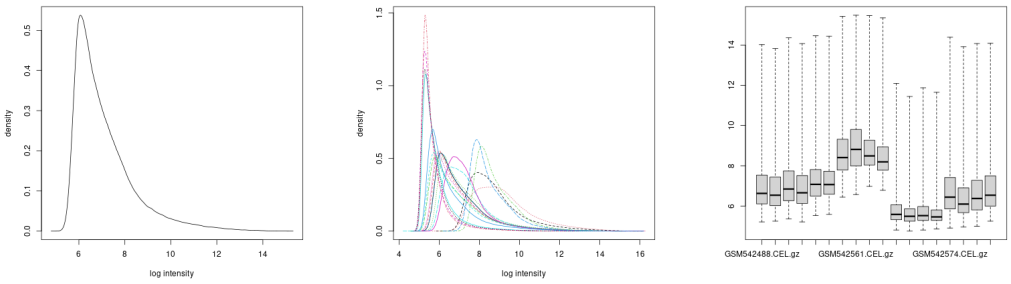
2.2.5 Densidades y diagramas de cajas

En los dos dibujos anteriores hemos ilustrado la variabilidad intra y entre arrays para un mismo gen. Tomemos un punto de vista más global. En particular, vamos a mostrar el comportamiento de los niveles de expresión para todas las sondas en un array, por ejemplo, el primero de ellos. Estamos evaluando la posibilidad de que alguno de ellos tenga un nivel de ruido inaceptable y debiera de ser eliminado del estudio.

Empezamos mostrando en la figura 2.5(a) la densidad de los niveles de expresión en el primer array. Lo hacemos utilizando un estimador kernel de la densidad.

```
affy::hist(gse21779raw[,1])
```

Aunque la función que utilizamos es `hist` (de histograma), no es histograma lo que estamos haciendo. Es una mala elección del nombre de la función. Es un estimador kernel de la densidad. Con objeto de ver (y



(a) (b) (c)

Figura 2.5: a)Estimador kernel de la densidad de las expresiones a nivel de sonda en el primer microarray. b) Estimadores kernel de la densidad de las intensidades para los distintos microarrays. Vemos que hay diferencias claras. c) Diagramas de cajas de las expresiones a nivel de sonda para los datos gse21779.

comparar) la variabilidad en los distintos arrays podemos estimar los estimadores kernel de las densidades de las expresiones en cada uno de los arrays que componen nuestra muestra. Lo tenemos en la figura 2.5(b).

```
affy::hist(gse21779raw)
```

Llama la atención la gran diferencia entre estos estimadores de la densidad para distintas muestras. En cualquier caso en todos ellos vemos una clara asimetría a la derecha. Obviamente corresponde a los genes con un alto nivel de expresión.

De un modo análogo podemos para evaluar la variabilidad las expresiones en cada arrays y entre arrays podemos utilizar diagramas de cajas. Se muestra en la figura 2.5(c).

```
affy::boxplot(gse21779raw)
```

En este caso los genes con una alta (anormalmente grande) expresión corresponden con el bigote superior en los diagramas de caja que acabamos de representar.

2.2.6 Utilizando arrayQualityMetrics

El paquete [76, arrayQualityMetrics] nos permite un control de calidad sencillo utilizando `arrayQualityMetrics::arrayQualityMetrics` ↪ `()`. Incluye opciones previamente comentadas y alguna más.

```
pacman::p_load(arrayQualityMetrics)
arrayQualityMetrics(expressionset = gse21779raw,
                    outdir = "Report_for_gse21779raw",
                    force = TRUE, ## Sobreescribe
                    do.logtransform = TRUE) ## Transforma a log2
```

2.2.7 Corrigiendo errores técnicos

Si algo queda claro en esta sección es la necesidad de tratar los datos antes de poder compararlos y obtener alguna conclusión biológica. Es decir, hemos de trabajar los valores observados a nivel de sonda antes de obtener un resumen que es la expresión del gen.

De hecho podemos ver los mínimos de PM en las distintas muestras.

```
apply(probes(gse21779raw,"pm"),2,min)
```

```
GSM542488.CEL.gz GSM542555.CEL.gz
            32 31
GSM542556.CEL.gz GSM542557.CEL.gz
            34 33
GSM542558.CEL.gz GSM542559.CEL.gz
            21 39
GSM542560.CEL.gz GSM542561.CEL.gz
            67 60
GSM542562.CEL.gz GSM542563.CEL.gz
            74 82
GSM542570.CEL.gz GSM542571.CEL.gz
            26 27
GSM542572.CEL.gz GSM542573.CEL.gz
            26 27
GSM542574.CEL.gz GSM542575.CEL.gz
            28 29
GSM542576.CEL.gz GSM542577.CEL.gz
            28 32
```

Y vemos como efectivamente no son nulos. La *corrección de fondo* pretende que aproximadamente sean nulos los valores más pequeños. Esto correspondería con aquellos genes que no tienen actividad.

También hemos visto que la distribución de las intensidades es claramente muy diferente entre arrays. Lo mostraban los estimadores kernel y los diagramas de cajas. El proceso de hacer comparables los datos entre arrays es lo que se conoce como *normalización*.

Una vez hemos corregido fondo y normalizadas las muestras, seguimos teniendo para cada gen distintas sondas en cada array. Toca resumir estos valores en un solo valor que indique la expresión del gen asociado al conjunto de sondas.

Estas tres operaciones pueden realizarse de un modo secuencial o bien hay métodos que las realizan de un modo combinado. Hay publicados muchos métodos. Lo que vamos a utilizar en este capítulo está en el paquete [74, affy]. Es especialmente interesante consultar la viñeta `builtinMethods.pdf` que aparece en la ayuda del paquete [74, affy]. Como material complementario es de interés [54, capítulo 2].

En estas notas vamos a estudiar con detalle dos métodos. El primero es un método que corrige cada chip, es el método MAS5.0 que es el que proporciona el software de Affymetrix. Muchos trabajan con los datos que directamente les da este software y por tanto es interesante conocerlo. Lo tratamos en § 2.2.8. El segundo es el método RMA. Este procedimiento trabaja simultáneamente con todos los chips del experimento. Es pues un procedimiento multiarray o multichip. Este método lo estudiamos en § 2.2.9.

El objetivo ahora es considerar procedimientos que corrigen las variabilidades entre los microarrays intentando conseguir un valor de expresión asociado a cada gen sin esos ruidos de caracter técnico.

Un comentario crítico sobre la idea de normalizar o preprocesar la información escrito desde el punto de vista estadístico. En Estadística el ruido no se corrige sino que se modeliza. La Estadística lo que hace es, utilizando un modelo probabilístico, cuantificar la señal y compararla con una cuantificación del ruido utilizando diferentes procedimientos. Por tanto el ruido ha de ser el observado y no el que podemos introducir nosotros mismos. Sin embargo, la realidad es que se ha desarrollado una extensa bibliografía que utiliza esta aproximación y hemos de seguir la tendencia. No parece lo más correcto. En cualquier caso, lo recomendable es utilizar distintas aproximaciones y ver cómo esto influye en los

análisis posteriores. La menos mala de las soluciones aunque siempre hay que tener prevención hacia estos procedimientos.

2.2.8 Método MAS5

Veremos cómo pasar de los datos a nivel de sonda a los valores de expresión del gen. Mucho de lo que luego analizamos depende de cómo se realiza este paso. Lo que se comenta es específico para Affymetrix GeneChip. En cada localización tendremos sondas que son oligonucleótidos específicos de un gen. De hecho son pares de sonda: una con la secuencia deseada y otra en la que se modifica la base en la posicion 13. La intensidad (o expresión) observada con el oligonucleótido correcto le llamamos PM (perfect matching) y el observado sobre el modificado le llamamos MM (miss matching). Se pensaba que si a PM le restábamos MM corregiríamos el ruido de fondo. Esto no es cierto, una cierta proporción de localizaciones muestran un valor MM mayor que el valor PM, lo que en principio va en contra de lo que lógicamente debería producirse.

Los datos y la información de un Affymetrix GeneChip se almacenan en distintos ficheros. En el fichero CEL tenemos la intensidad media de los pixels de la sonda, la desviación estándar e información de dónde se sitúa la sonda dentro del array. Cada array lleva asociado un fichero CEL. Para poder interpretar el significado biológico de las sondas se necesita un mapa indicando a qué gen corresponde cada gen. Esta información está en el fichero CDF. Este fichero depende del chip utilizado y no varía para distintas muestras.

En esta sección estudiamos el método MAS5.0 y nos basamos en el manual http://media.affymetrix.com/support/technical/whitepapers/sadd_whitepaper.pdf que Como casi todos los manuales técnicos no es prodigio de claridad precisamente.

Empezamos a trabajar a partir del fichero .CEL que ha sido generado por el software Microarray Suite.

Corrección de fondo

Se pretende determinar unos valores para restarlos a las intensidades originales en cada celda.

Determinan valores por zonas

1. Se divide el array en K zonas rectangulares: Z_k con $k = 1, \ldots, K$.
2. En cada Z_k se ordenan las intensidades observadas en las distintas celdas y se considera el 2% con intensidades menores. Se toma como medida de fondo la media muestral de las intensidades previamente consideradas. Denotamos esta valor como $b(Z_k)$.
3. Del mismo modo que en punto anterior, se considera en cada Z_k ese 2% de celdas con intensidades menores y calcular su desviación estándar que denotamos $s(Z_k)$.

Suavizando el ajuste Sea la celda localizada en (x, y) y vamos a denotar por $d_k(x, y)$ la distancia desde (x, y) al centro de la región Z_k. Consideramos la siguiente cantidad

$$w_k(x, y) = \frac{1}{d_k^2(x, y) + s_0}$$

donde s_0 nos evita que el cociente se pueda anular y por defecto toma el valor $s_0 = 100$. Para la celda localizada en (x, y) consideramos

$$b(x,y) = \frac{1}{\sum_{k=1}^{K} w_k(x,y)} \sum_{k=1}^{K} w_k(x,y) b(Z_k)$$

Corrección del ruido Ahora vamos a bajar las intensidades restándole una medida de ruido de fondo local. El problema que se puede producir es que nos encontremos con valores negativos. Este valor de ruido en la celda localizada en (x, y) se calcula como

$$n(x,y) = \frac{1}{\sum_{k=1}^{K} w_k(x,y)} \sum_{k=1}^{K} w_k(x,y) s(Z_k).$$

La idea ahora es quitar a las intensidades originales el fondo ($b(x, y)$) que hemos calculado pero asegurándonos que no nos surja valores negativos. La intensidad original se denota como $I(x, y)$ en la localización (x, y). El valor ajustado de la intensidad viene dado por

$$A(x,y) = \max\{I(x,y) - b(x,y), f_0 n(x,y)\}$$

El parámetro f_0 se suele fijar en $f_0 = 0.5$.[3]

Cálculo del valor de expresión

Por el procemiento descrito anteriormente podemos ajustar los valores de PM y de MM. En lo que sigue se supone que hemos ajustado estos valores.

Lo esperable y correcto es que en una celda tengamos un valor de PM mayor que el correspondiente valor MM. Esto no es así en muchas ocasiones como hemos visto. Se define un valor IM (Ideal Mismatch)[4] que intenta corregir este problema.

Denotamos por (PM_{ij}, MM_{ij}) el valor de PM y de MM de la j-ésima sonda perteneciente al i-ésimo conjunto de sondas (formado por todas las sondas asociadas a un mismo gen).

Se calcula un valor que llaman background específico para cada conjunto de sondas.

$$\boldsymbol{B}_i = T_{bi}\{\log_2(PM_{ij}) - \log_2(MM_{ij}) : j = 1, \ldots, n_i\},$$

donde T_{bi} denota el algoritmo bipesado en un solo paso (§ 18.1). Básicamente un promedio robusto de las diferencias indicadas que no son más que logaritmos de sus cocientes. ¿Qué estima este valor $\boldsymbol{B}_i$ para el conjunto de sondas i? Si el valor de

$$IM_{ij} = \begin{cases} MM_{ij} & \text{si } MM_{ij} < PM_{ij} \\ \frac{PM_{ij}}{2^{\boldsymbol{B}_i}} & \text{si } MM_{ij} \geq PM_{ij} \text{ y } \boldsymbol{B}_i > \alpha \\ \frac{PM_{ij}}{2^{\frac{\alpha}{1+(\alpha-\boldsymbol{B}_i)/\beta}}} & \text{si } MM_{ij} \geq PM_{ij} \text{ y } \boldsymbol{B}_i \leq \alpha \end{cases}$$

Por defecto se toma $\alpha = 0.03$ y $\beta = 10$.

[3] En el manual hay una igualdad incomprensible que dice $I(x, y) \max\{I(x, y), 0.5\}$. En fin, ya veremos qué es exactamente lo que quieren hacer. He puesto lo que sensatamente parece que quieren hacer.

[4] Un apaño se decía antes.

Ahora calculamos

$$V_{ij} = \max\{PM_{ij} - IM_{ij}, d\}$$

siendo el valor por defecto para d, $d = 2^{-20}$.

Transformamos este valor con

$$\mathbb{V}_{ij} = \log_2(V_{ij})$$

con $j = 1, \ldots, n_i$. Y volvemos a aplicar un estimador biponderado en un solo paso (§ 18.1).

$$SignalLogValue_i = T_{bi}(\mathbb{V}_{i1}, \ldots, \mathbb{V}_{in_i}).$$

Fijamos una señal objetivo, Sc, que por defecto se toma $Sc = 500$. Se calcula el siguiente factor de escala para el grupo de sondas i.

$$sf_i = \frac{Sc}{MediaAjustada\{2^{SignalLogValue_i}, 0.02, 0.98\}}$$

En la anterior expresión MediaAjustada indica la media ajustada (media de los valores quitando una cierta proporción de los más grandes y de los más pequeños) en donde se quita el 2% de los más grandes y el 2% de los más pequeños.

El valor final para el conjunto de sondas i es

$$ReportedValue(i) = sf_i \times 2^{SignalLogValue_i}$$

Se incorpora una posibilidad de normalización que no indicamos.

2.2.9 Robust multichip average (RMA)

Una referencia con muchos más detalles de los que vemos en esta sección es [22, pág. 41-59] En este método trabajamos con todos los arrays simultáneamente Su nombre **R**obust **M**ultiarray **A**verage indica que, a diferencia del método anterior donde se procesaba cada array independientemente, aquí vamos a procesarlos conjuntamente. El procedimiento solamente utiliza los valores PM y no utiliza para nada los valores MM.

Corrección de fondo

Se empieza calculando una corrección de fondo.[37] Asumimos (sugerido por la distribución empírica observada en las intensidades de las sondas) que los valores PM pueden considerarse como la suma de una variable Y con distribución normal con media μ y varianza σ^2 (que modeliza el ruido) y de una variable X con distribución exponencial con media α (que modelizaría la señal). Por tanto el valor observado aleatorio S sería

$$S = X + Y$$

con $X \sim Exp(\alpha)$ e $Y \sim N(\mu, \sigma^2)$. La normal se trunca en cero para evitar valores negativos. Si S denota la intensidad aleatoria en la sonda entonces, para un valor observado $S = s$, se tiene

$$E(X|S = s) = a + b\frac{\phi(\frac{a}{b})}{\Phi(\frac{a}{b})}$$

[37] Que esencialmente no se ha publicado. Aparece una descripción más o menos vaga en [72].

siendo $a = s - \mu - \sigma^2\alpha$ y $b = \sigma$ mientras que ϕ y Φ denotan las funciones de densidad y de distribución de una normal estándar (con media 0 y varianza 1). Los parámetros del modelo (α, μ, σ^2) se estiman mediante un procedimiento no paramétrico. Utilizando las intensidades observadas en las sondas PM se estima la moda de la distribución. Aquellos valores por encima de la moda son utilizados para estimar α y los puntos por debajo de la moda se utilizan para estimar los parámetros μ y σ^2.

Normalización de cuantiles

En este paso se pretende conseguir que las distribuciones empíricas de las expresiones a nivel de sonda de los distintos arrays sean la misma.[5] Esencialmente lo que buscamos es que las expresiones que se observan en los distintos microarrays sean las mismas y con las mismas frecuencias.

La normalización que se aplica en este método es una *normalización de cuantiles* (Quantile normalization [22]).

Veamos qué es con un ejemplo detallado y sencillo.

1. Nuestro experimento tiene dos conjuntos de sondas y cada una con tres sondas. Además suponemos que tenemos simplemente tres chips. En la siguiente tabla recogemos las expresiones.

Conjunto de sondas	Sonda	Chip 1	Chip 2	Chip 3
1	1	7	9	19
1	2	3	5	14
1	3	2	6	11
2	1	4	8	8
2	2	10	11	16
2	3	12	10	15

2. Determinamos los valores mayores en cada chip y calculamos el promedio.

```
(12 + 11 + 19)/ 3
```

```
[1] 14
```

Sustituimos los valores originales por estos promedios.

Conjunto de sondas	Sonda	Chip 1	Chip 2	Chip 3
1	1	7	9	**14**
1	2	3	5	14
1	3	2	6	11
2	1	4	8	8
2	2	10	**14**	16
2	3	**14**	10	15

3. Consideramos los segundos valores mayores en cada chip y los promediamos.

```
(10 + 10 + 16) /3
```

```
[1] 12
```

[5] Recordemos que en este método solamente utilizamos las expresiones de las sondas PM.

Sustituimos los valores originales por los promedios.

Conjunto de sondas	Sonda	Chip 1	Chip 2	Chip 3
1	1	7	9	14
1	2	3	5	14
1	3	2	6	11
2	1	4	8	8
2	2	**12**	14	**12**
2	3	14	**12**	15

4. Consideramos los terceros valores mayores en cada chip.

```
(7 + 9 + 15)/3
```

```
[1] 10.33333
```

Sustituimos los valores originales por los promedios.

Conjunto de sondas	Sonda	Chip 1	Chip 2	Chip 3
1	1	**10.33**	**10.33**	14
1	2	3	5	14
1	3	2	6	11
2	1	4	8	8
2	2	12	14	12
2	3	14	12	**10.33**

5. Y así sucesivamente hasta llegar al sexto valor que ya es el más pequeño en cada chip. Los datos finales son los de la siguiente tabla.

Conjunto de sondas	Sonda	Chip 1	Chip 2	Chip 3
1	1	10.33	10.33	14
1	2	6.66	5	8.66
1	3	5	6.66	6.66
2	1	8.66	8.66	5
2	2	12	14	12
2	3	14	12	10.33

Una vez ilustrado el método es sencillo de explicar.

1. Tomamos n vectores de longitud N y construimos la matriz X $N \times n$ que los tiene por vectores columna.

2. Ordenamos cada columna de X separadamente y tenemos X_s.

3. Calculamos la media por filas de la matrix X_s y creamos X'_s una matriz con la misma dimensión que X, tal que en la fila j tenemos la media de la fila de X_s repetida n veces.

4. Obtenemos X_t en donde cada columna de X'_s recupera el orden original.

¿Qué estamos haciendo desde el punto de vista probabilístico? Hemos considerado los cuantiles muestrales en cada columna. Hemos considerado la media muestral de estos cuantiles muestrales. Supongamos que F es la función de distribución de estas medias de cuantiles muestrales. Tomamos G la función de distribución muestral o empírica de las expresiones de cada uno de los microarrays. Si la expresión original la denotamos por y_i para la i-ésima sonda entonces la nueva expresión de dicha sonda viene dada por $y'_i = F^{-1}(G(y_i))$. Cuando F corresponde a la uniforme en el intervalo $[0, 1]$ esto recibe el nombre de transformación integral de la probabilidad.

Resumen

Ya hemos realizado la corrección de fondo (§ 2.2.9). A los valores obtenidos les hemos aplicado una normalización de cuantiles (38). Ahora nos queda obtener el resumen de las distintas sondas dentro de cada conjunto (de sondas) y para cada microarray. Se aplica el método median polish de Tukey (§ 18.2). Previamente necesitamos algo de notación. En cada array tendremos N sondas y suponemos que tenemos n arrays. Por tanto la matriz de expresión a nivel de sonda será: $\boldsymbol{x} = [x_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$. La sonda i-ésima en el array j-ésimo tiene una expresión o intensidad x_{ij}. Las sondas las suponemos agrupadas en grupos correspondientes a un mismo gen. Denotamos el grupo k-ésimo de sondas con S_k. Por tanto, $S_k \subset \{1,\ldots,N\}$. Por ejemplo, el conjunto $S_k = \{i_1,\ldots,i_{|S_k|}\}$, es decir, corresponde con las filas $\{i_1,\ldots,i_{|S_k|}\}$ de la matrix $\boldsymbol{x}$ original. Por simplificar la notación tomaremos $y_{rj} = x_{i_r,j}$ y por tanto

$$\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,|S_k|,j=1,\ldots,n} = [x_{ij}]_{i\in S_k,j=1,\ldots,n}.$$

En resumen, la matriz $\boldsymbol{y}$ simplemente corresponde con las filas de la matrix $\boldsymbol{x}$ correspondiente al grupo de sondas S_k y todas las columnas.

En lo que sigue nos referimos al grupo k-ésimo de sondas y lo que se asume es para cada grupo. Asumimos el siguiente modelo:

$$\log_2(y_{ij}) = \mu + \theta_j + \alpha_i + \epsilon_{ij} \tag{2.1}$$

con las siguientes restricciones

1. la mediana de $\{\theta_j : j = 1,\ldots,n\}$ es nula,
2. la mediana de $\{\alpha_i : i = 1,\ldots,|S_k|\}$ es nula
3. la mediana de $\{\epsilon_{ij} : j = 1,\ldots,n\}$ sea nula para cada i,
4. la mediana de $\{\epsilon_{ij} : i = 1,\ldots,|S_k|\}$ sea nula para cada j.

El método median polish es un procedimiento para estimar los parámetros anteriores que aparecen en las ecuaciones (2.1). ¿Cómo? Consideramos, para cada grupo de sondas (por ejemplo, el k-ésimo S_k), la matriz

$$\begin{array}{ccc|c}
\delta_{1,1} & \cdots & \delta_{1,n} & a_1 \\
\vdots & \ddots & \vdots & \vdots \\
\delta_{|S_k|,1} & \cdots & \delta_{|S_k|,n} & a_{|S_k|} \\
\hline
b_1 & \cdots & b_n & m
\end{array}$$

En esta matriz las sondas corresponden a las filas y las columnas a los arrays. Además añadimos una columna y una fila (las últimas separadas por una línea continua) en la que aparecen los efectos de fila y columna. Aplicamos el siguiente procedimiento iterativo.

1. Fijamos $\delta_{ij} = \log_2(y_{ij})$, $a_i = b_i = 0\ \forall i,j$.
2. Calculamos la mediana para cada fila a lo largo de las columnas ignorando la última columna (separada por la línea).
3. Restamos a cada elemento de la fila la mediana correspondiente. Esta mediana se suma a la última columna (formada por $a_1,\ldots,a_{|S_k|},m$).
4. Calculamos la mediana para cada columna de los valores correspondientes a las distintas filas sin considerar la última fila.

5. Restamos la mediana calculada en el paso anterior a cada elemento de la columna exceptuando la última fila. A la última fila sumamos las medianas calculadas.

6. El proceso descrito en los cuatro pasos anteriores continua iterando sucesivamente entre filas y columnas hasta que los cambios que se producen son nulos o muy pequeños.

7. Una vez finalizado el proceso iterativo tendremos $\hat{\mu} = m$, $\hat{\theta}_j = b_j$ y $\hat{\alpha}_i = a_i$. El valor δ_{ij} será el estimador de ϵ_{ij}: $\hat{\epsilon}_{nij} = \delta_{ij}$.

Una vez estimados estos parámetros los estimadores (en escala logarítmica en base 2) de la expresión del grupo de sondas k en el array j viene dada por

$$\hat{\mu} + \hat{\theta}_j.$$

Es un método que no nos proporciona estimaciones de error de los estimadores. Es muy rápido de implementar. Al utilizar medianas es menos sensibles a observaciones extremas. Sin embargo, el resultado puede ser distinto según empecemos por filas o columnas.

2.2.10 MAS5 y RMA con affy

En el paquete [74, affy] tenemos implementados los dos procedimientos comentados en las dos secciones previas. Usamos las función `affy::rma()` y `affy::mas5()`.

```
gse21779_rma = affy::rma(gse21779raw)
gse21779_mas5 = affy::mas5(gse21779raw)
```

En la figuras 2.6(a) y 2.6(b) tenemos los estimadores kernel de densidad de los niveles de expresión para los distintos chips una vez hemos aplicado el método RMA y MAS5. En las figuras 2.6(c) y 2.6(d) mostramos los diagramas de cajas correspondientes a los mismos datos preprocesados.

```
library(geneplotter)
geneplotter::multidensity(exprs(gse21779_rma))
geneplotter::multidensity(exprs(gse21779_mas5))
graphics::boxplot(exprs(gse21779_rma))
graphics::boxplot(exprs(gse21779_mas5))
```

2.2.11 Otros métodos de normalización

Hemos visto dos métodos. Sin embargo, podemos considerar métodos distintos con el paquete [74, affy] Empezamos con los métodos de corrección de fondo. Las opciones son

```
bgcorrect.methods()
```

```
[1] "bg.correct" "mas" "none"
[4] "rma"
```

Las opciones que nos muestra corresponden con:

none En la opción `none` no hacemos nada. Dejamos los valores originales.

mas Utiliza el procedimiento que hemos visto en MAS5.0.

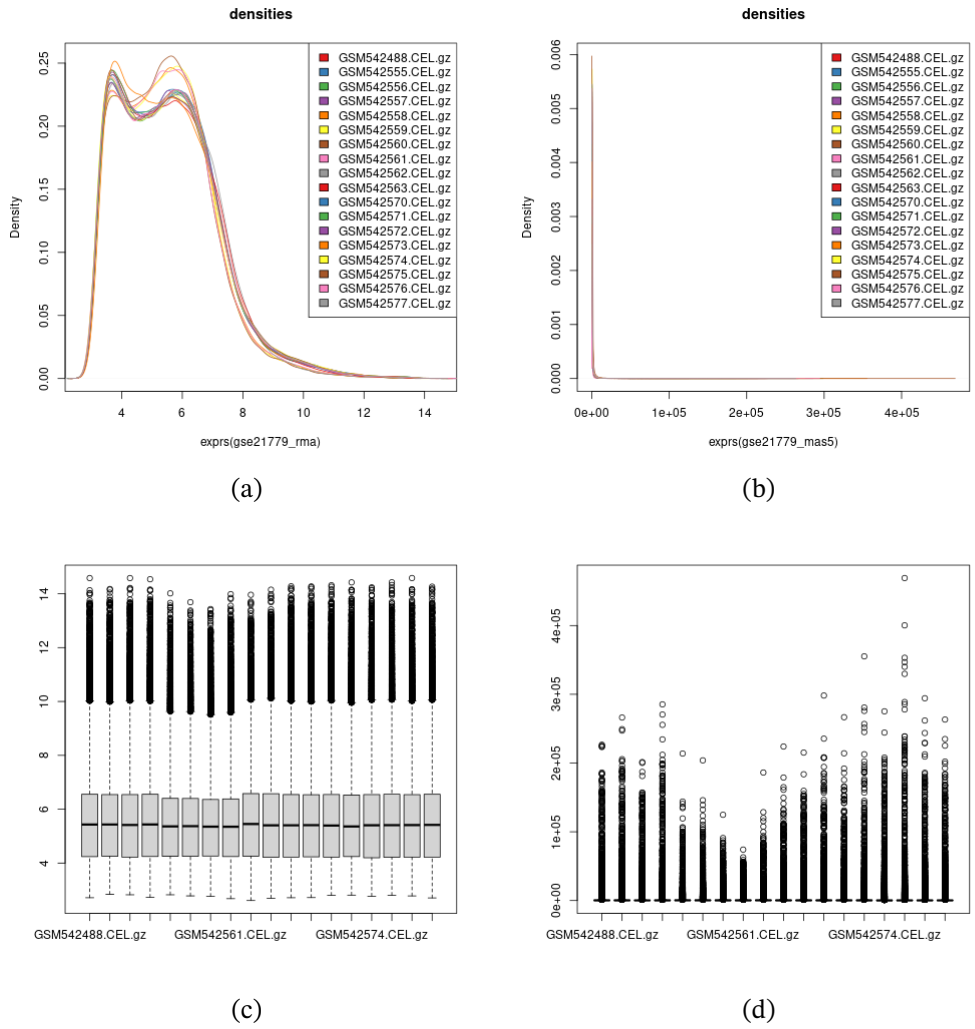


Figura 2.6: a) Estimadores kernel de densidad para los chips una vez hemos aplicado el procedimiento RMA a los datos gse21779. b) Lo mismo con el método MAS5. c) Boxplot para los niveles de expresión de los datos gse21779 con método RMA. d) Lo mismo que el punto (c) pero con el método MAS5.

RMA Utiliza el procedimiento visto en algoritmo RMA.

Nos ofrece los siguientes métodos de normalización.

```
normalize.methods(gse21779raw)
```

```
[1] "constant" "contrasts"
[3] "invariantset" "loess"
[5] "methods" "qspline"
[7] "quantiles" "quantiles.robust"
```

En particular, para los chip de Affymetrix, tenemos distintas opciones para corregir los valores PM.

```
pmcorrect.methods()
```

```
[1] "mas" "methods" "pmonly"
[4] "subtractmm"
```

Los métodos para resumir los valores de los conjuntos de sondas en un único valor de expresión del gen los podemos ver con el siguiente código.

```
express.summary.stat.methods()
```

```
[1] "avgdiff" "liwong" "mas"
[4] "medianpolish" "playerout"
```

La función `affy::expresso()` permite realizar los distintos pasos del preproceso. De hecho, la función `affy::mas5()` que hemos utilizado anteriormente no es más que una llamada a esta función.

Una explicación detallada de cada una de las opciones la podemos encontrar en la viñeta `builtinMethods` del paquete [74, affy].

2.2.12 ¿Cómo hacer un preprocesado sencillo a partir de datos de expresión?

En la sección anterior hemos tratado cómo preprocesar datos partiendo de datos a nivel de sonda muy orientado a Affymetrix. Cuando conseguimos datos de expresión (por ejemplo de GEO) en ocasiones no tenemos los datos a nivel de sonda (los .CEL en el caso de Affymetrix GeneChip) sino que tenemos unos datos de expresión con un preproceso que no controlamos exactamente. El problema es que estos datos todavían necesitan un procesado para homogeneizarlos. ¿Cómo hacerlo de un modo sencillo? Una opción rápida la tenemos con la función `limma::normalizeBetweenArrays()`.

Ejemplo 2.1 (GSE1397). *Veamos un ejemplo con los datos GSE1397. En § 2.7.6 hemos visto cómo bajarlos de GEO y cómo construir el `ExpressionSet`* ↪ .

```
load(paste(dirTamiData,"gse1397raw.rda",sep=""))
```

Son datos Affymetrix correspondientes al siguiente chip.

```
annotation(gse1397raw)
```

```
[1] "GPL96"
```

Supongamos que queremos aplicar una normalización de los cuantiles (38 y [20]). Podemos hacer

```
library(limma)
exprs1 = normalizeBetweenArrays(exprs(gse1397raw),
                                method = "quantile")
```

O bien podemos hacer que coincidan las medianas de los distintos arrays con

```
exprs1 = normalizeBetweenArrays(exprs(gse1397raw),
                                method = "scale")
```

En la ayuda de `limma::normalizeBetweenArrays()` *podemos consultar otras opciones.*

2.3 Otros métodos de normalización

El texto [131] es una excelente referencia sobre lo tratado en este tema y propone métodos adicionales. Mucho del material procede de este libro.

En [98] se propone un método genérico de normalización partiendo de una matriz de expresión en donde cada columna corresponda a una muestra. En [98, página 2] se propone simplemente estimar las funciones de distribución de las distintas muestras y aplicando la transformada inversa a las funciones de distribución estimadas tendremos los nuevos datos. La normalización de percentiles utiliza esta idea y la distribución estimada es una uniforme sobre las medias de los percentiles observados. El método está implementado en https://github.com/mengqinxue/DBNorm.

2.4 Datos en línea

Las dos bases de datos de microarrays de DNA que vamos a usar son GEO y ArrayExpress. Asociadas a las bases de datos tenemos los paquetes [41] y [77]. En las secciones que siguen vemos ejemplos de su uso.

2.5 Sobre cómo usar un ExpressionSet

Los datos de microarrays de DNA suelen venir almacenados en la clase `Biobase::ExpressionSet`. En esta sección manejamos esta clase que es básica con este tipo de información.

2.5.1 sample.ExpressionSet

El experimento `Biobase::sample.ExpressionSet` tiene 26 muestras y 500 genes. Sobre las muestras conocemos tres variables (o covariables): `sex`, `type` (caso y control) y `score`. La última covariable es continua. Estos datos de expresión están almacenados en un objeto de clase `Biobase::ExpressionSet`. Para poder utilizar esta clase hemos de cargar el paquete [53, Biobase].

```
library(Biobase)
```

Cargamos los datos.

```
data(sample.ExpressionSet)
```

¿De qué clase es el objeto?

```
class(sample.ExpressionSet)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Las variables fenotípicas, que nos describen las distintas muestras con las que estamos trabajando, las podemos obtener con

```
phenoData(sample.ExpressionSet)
```

```
An object of class 'AnnotatedDataFrame'
  sampleNames: A B ... Z (26 total)
  varLabels: sex type score
  varMetadata: labelDescription
```

¿De qué clase es el objeto?

```
class(phenoData(sample.ExpressionSet))
```

```
[1] "AnnotatedDataFrame"
attr(,"package")
[1] "Biobase"
```

Vemos que esta clase está definida en [53, Biobase] y es de tipo

```
typeof(phenoData(sample.ExpressionSet))
```

```
[1] "S4"
```

Este sistema orientado a objetos es de uso mayoritario en Bioconductor ([145, capítulo 7]). Podemos acceder a los distintos `slots` que la componen con las funciones (accesors). Veamos cómo obtener los nombres de las muestras, los nombres de las variables fenotípicas e información adicional sobre estas variables fenotípicas.

```
sampleNames(sample.ExpressionSet)
```

```
 [1] "A" "B" "C" "D" "E" "F" "G" "H" "I" "J" "K"
[12] "L" "M" "N" "O" "P" "Q" "R" "S" "T" "U" "V"
[23] "W" "X" "Y" "Z"
```

```
varLabels(sample.ExpressionSet)
```

```
[1] "sex" "type" "score"
```

```
varMetadata(sample.ExpressionSet)
```

```
     labelDescription
sex Female/Male
type Case/Control
score Testing Score
```

¿Cómo podemos acceder al `Biobase::AnnotatedDataFrame` (notemos que no es un `data.frame`) que contiene la información?

```
head(pData(sample.ExpressionSet))
```

```
    sex type score
A Female Control 0.75
B Male Case 0.40
C Male Control 0.73
D Male Case 0.42
E Female Case 0.93
F Male Control 0.22
```

¿Cómo podemos acceder a una variable fenotípica dada, por ejemplo `type`? Tenemos varias opciones.

```
pData(sample.ExpressionSet)[,"type"]
pData(sample.ExpressionSet)$type
sample.ExpressionSet$type
```

La matriz de expresión (mostramos la primera fila) se obtiene con

```
head(exprs(sample.ExpressionSet),n=1)
```

```
                  A B C D
AFFX-MurIL2_at 192.742 85.7533 176.757 135.575
                  E F G H
AFFX-MurIL2_at 64.4939 76.3569 160.505 65.9631
                  I J K L
AFFX-MurIL2_at 56.9039 135.608 63.4432 78.2126
                  M N O P
AFFX-MurIL2_at 83.0943 89.3372 91.0615 95.9377
                  Q R S T
AFFX-MurIL2_at 179.845 152.467 180.834 85.4146
                  U V W X
AFFX-MurIL2_at 157.989 146.8 93.8829 103.855
                 Y Z
AFFX-MurIL2_at 64.434 175.615
```

Lo que nos aparece a la izquierda son los identificadores de las sondas (en particular con sondas de control de Affymetrix). Podemos mostrar dos filas que no correspondan a sondas de control, por ejemplo, la que ocupa la fila 100 de la matriz de expresión del modo habitual.

```
exprs(sample.ExpressionSet)[100,]
```

```
       A B C D E
-28.99850 -30.05320 -26.97270 -23.00420 -18.31410
       F G H I J
316.92200 -21.24100 -14.67470 -22.34640 -26.95820
       K L M N O
-23.27410 -31.22210 -8.00193 -33.86130 -17.81590
       P Q R S T
-10.65560 -10.91990 -26.02170 -22.64670 -17.45640
       U V W X Y
-26.20010 -22.03100 -22.02760 -11.92950 -14.14710
       Z
-33.95400
```

Podemos ver a la izquierda de la salida anterior que aparecen los identificadores Affymetrix (PROBEID). ¿Dónde están almacenados?

```
rownames(sample.ExpressionSet)[100]
```

```
[1] "31339_at"
```

El sitio adecuado es el slot `featureNames`.

```
featureNames(sample.ExpressionSet)[100]
```

```
[1] "31339_at"
```

También tenemos un slot adicional en el que se pueden poner las correspondencias de estos identificacores con otras bases de datos. En este datos de ejemplo que manejamos no están definidos como podemos ver.

```
fData(sample.ExpressionSet)
```

```
data frame with 0 columns and 500 rows
```

¿Qué chip fue usado para obtener estos datos? Esto es necesario para conocer la correspondencia de nuestros identificadores con otros.

```
annotation(sample.ExpressionSet)
```

```
[1] "hgu95av2"
```

Finalmente veamos una breve descriptiva de las covariables que nos proporcionan información sobre las muestras y que utilizaremos para ilustrar estas notas. Para las categóricas vemos una tabla de frecuencias absolutas.

```
table(sample.ExpressionSet$sex)
```

```

Female Male
   11 15
```

```
table(sample.ExpressionSet$type)
```

```

  Case Control
    15 11
```

Y un resumen numérico de `score`.

```
summary(sample.ExpressionSet$score)
```

```
   Min. 1st Qu. Median Mean 3rd Qu. Max.
 0.1000 0.3275 0.4150 0.5369 0.7650 0.9800
```

Para tener información sobre un objeto de clase ExpressionSet es recomendable consultar [67, capítulo 2]. En la viñeta de [53, Biobase] tenemos una buena descripción de la creación y manipulación de la clase `Biobase::ExpressionSet`.

2.5.2 ALL

En esta sección pretendemos mostrar cómo utilizar funciones que manejan `ExpressionSet`. Vamos a utilizar los datos `ALL` que aparecen en el paquete [83, ALL]. Cargamos el paquete.

```
data(ALL,package="ALL")
```

Podemos ver que es un resumen de la información que tenemos en ALL.

```
ALL
```

```
ExpressionSet (storageMode: lockedEnvironment)
assayData: 12625 features, 128 samples
  element names: exprs
protocolData: none
phenoData
  sampleNames: 01005 01010 ... LAL4 (128
    total)
  varLabels: cod diagnosis ... date last
    seen (21 total)
  varMetadata: labelDescription
featureData: none
experimentData: use 'experimentData(object)'
  pubMedIds: 14684422 16243790
Annotation: hgu95av2
```

Podemos ver los niveles de expresión (en filas tenemos sondas y en columnas muestras). Los niveles de expresión han sido preprocesados y aparecen el logaritmo en base 2 de este valor ya preprocesado. Se obtienen (y no lo mostramos) con

```
exprs(ALL)
```

Los datos fenotípicos los tendremos con (tampoco los mostramos)

```
pData(ALL)
```

¿Cuál fue chip que se utilizó para obtener estos datos?

```
annotation(ALL)
```

```
[1] "hgu95av2"
```

Con `pData(ALL)` hemos visto los nombres (y los valores) de las covariables que nos describen las distintas muestras. Solamente los nombres los podemos ver con

```
names(pData(ALL))
```

Y si simplemente queremos ver los valores de la variable BT podemos hacerlo con

```
ALL$BT
```

Los datos de expressión aparecen en una matriz con filas (features que suelen corresponder a sondas) y con columnas (que corresponden con muestras). Vamos a seleccionar una parte de las muestras. En concreto, vamos a considerar las muestras tales que la variable fenotípica `mol.bio` toma el valor `NEG`. Determinamos las columnas.

```
selcol = ALL$mol.biol == "NEG"
```

Y nos quedamos con esa muestras.

```
ALL[,selcol]
```

Lo importante es ver que todo el `Biobase::ExpressionSet` tiene menos muestras y también se han modificado (eliminando) los datos de expresión y los datos fenotípicos.

2.6 De cómo construir un ExpressionSet

Supongamos (y no es mucho suponer) que tenemos los datos de expresión en un fichero, en otro fichero tenemos la descripción de las distintas muestras que hemos analizado, en otro fichero (o en la red) tenemos información sobre los genes (o sondas) que tenemos en las filas de la matriz de expresión. Tenemos información sobre las distintas muestras que componen toda la experimentación, bajo qué condiciones se consiguieron. Lo que podemos llamar la descripción fenotípica de las muestras. Y finalmente sabemos cosas sobre toda la experimentación realizada: autores, publicación principal que se derivó, objetivo de la experimentación. Y, como es habitual, lo tenemos todo por separado, en distintos ficheros. La clase Biobase::ExpressionSet tiene como objetivo tenerlo todo contenido en un mismo objeto. En esta sección vemos cómo construir este ExpressionSet a partir de la información dispersa. En ocasiones esto nos puede pasar si bajamos la información de alguna publicación en la que nos aparece como información suplementaria.

Es probable que en nuestro trabajo no tengamos que realizar todo el proceso que indicamos a continuación pero con mucha probabilidad al menos una parte sí.

ExpressionSet

Es una estructura de datos que nos permite mantener en un solo objeto toda la información que relativa a un experimento con microarrays.

Para poder utilizar esta clase necesitamos el paquete [53, Biobase]. Lo cargamos (la clase y los métodos que luego vamos a utilizar).

```
pacman::p_load(Biobase)
```

¿Qué necesitamos?

Supongamos que tenemos la información por separado y queremos construir un ExpressionSet. ¿Qué datos nos describen un experimento genómico de alto rendimiento? Podemos distinguir la siguiente información:

1. Los datos del ensayo (los niveles de expresión).
2. Datos sobre las muestras que podemos llamar de un modo genérico *metadatos fenotípicos*.
3. Anotaciones sobre las sondas u otros metadatos.
4. Una descripción del experimento.

Vamos a construir un ExpressionSet construyendo cada una de las componentes a partir de unos datos.[6]

Datos del ensayo

Son los valores de expresión posiblemente preprocesados. Es una matriz con N filas y con n columnas. El número N de filas corresponde con el número de características (features) del chip y n con el número de muestras. Leemos los datos (posiblemente con `read.table()` o `read.csv()`. Nos han pasado los datos.[38] Normalmente las funciones

[38] Como Dios quiere y nosotros hemos de averiguar.

[6] Los datos que estamos usando han sido amablemente proporcionados por Joan Climent y están ligeramente modificados respecto de los publicados originalmente en [107].

`utils::read.table()` o `utils::read.csv()` nos suelen resolver el problema.

```
pacman::p_load("tamidata")
finput = system.file("extdata","n06_expr.txt",package="tamidata")
exprs0 = read.table(file = finput,header = TRUE,sep="\t")
```

Cuando usamos estas funciones nos devuelve un `data.frame` como podemos comprobar.

```
class(exprs0)
```

```
[1] "data.frame"
```

En lo que sigue necesitamos que sea una matriz. Lo hacemos (por cierto el -1 nos sirve para quitar la primera columna de la matriz donde teníamos los nombres de las sondas).

```
exprs0 = as.matrix(exprs0[,-1])
```

Y podemos comprobar que ya lo tenemos.

```
class(exprs0)
```

```
[1] "matrix" "array"
```

¿Cuántas filas (features) y columnas (samples) tenemos?

```
dim(exprs0)
```

```
[1] 22215 54
```

Los nombres de las columnas son

```
head(colnames(exprs0))
```

```
[1] "X600MPE" "AU565" "BT474" "BT483"
[5] "CAMA1" "DU4475"
```

Y podemos ver (no la mostramos) la matriz de expresión con

```
exprs0
```

Vamos a etiquetar las filas con los identificadores de las sondas.

```
finput = system.file("extdata","n06_gene_attributes.txt",
                     package="tamidata")
rn = read.table(file = finput,header = TRUE,sep="\t")
rownames(exprs0) = rn[,1]
```

Datos fenotípicos

Es la información que tenemos de las distintas muestras. Será una matriz de datos (un data.frame) con tantas filas como muestras, n, y con tantas columnas como variables tenemos. Estas variables que nos describen las muestras podemos llamarlas covariables (una denominación muy estadística). En nuestro caso los datos fenotípicos los tenemos en el fichero `sample_attributes.txt`. Los leemos.

```
finput = system.file("extdata","n06_sample_attributes.txt",
    package="tamidata")
pd0 = read.table(finput,sep = "\t",header=TRUE)
```

Podemos ver el número de muestras (que ha de coincidir con el número de columnas de la matriz de expresión) y de covariables.

```
dim(pd0)
```

```
[1] 54 14
```

Es **necesario** que los nombres de las filas de `pd0` coincidan con los nombres de las columnas de `exprs0` ya que nos están dando covariables relativas a estas muestras.

```
rownames(pd0) = colnames(exprs0)
```

Esto es fundamental. Si no coinciden los nombres de las columnas de la matriz de expresión con los nombres de las filas de la matriz de datos de las covariables no podremos construir el `Biobase::ExpressionSet`. Y ahora es conveniente un pequeño resumen descriptivo de las covariables (no mostrado).

```
summary(pd0)
```

Tenemos covariables categóricas (en R se les llama de clase **`factor`**) y covariables numéricas. Con las covariables categóricas tenemos las frecuencias mientras que con las numéricas nos da mínimo, primer cuartil (o percentil de orden 25 %), mediana y media, tercer cuartil (o percentil de orden 75 %) y máximo. Podemos ver los tipos (o clases) de cada columna con **`sapply()`**.

```
sapply(pd0,class)
```

```
 IDENTIFIER sample_no Type in_CGH
"character" "integer" "character" "integer"
       ALL CDH1_meth SFRP1 HER2
  "integer" "character" "numeric" "character"
       KIT known.p53 BRCA order
"character" "character" "character" "integer"
  adherence per3
"character" "integer"
```

Podemos tener información adicional sobre las covariables que nos describen las muestras. Por ejemplo, un nombre de variable como `known.`
↪ `p53` puede no significar demasiado. Pero si una etiqueta como "Presencia de p53"[7] ¿Cómo podemos añadir esta información?

```
metadatos = data.frame(labelDescription=
            c("Identificador","Número de muestra",
              "Tipo", "En CGH","ALL","Método CDH1",
              "SFRP1","HER2","KIT","Presencia de p53",
              "BRCA","Orden","Adherencia","per3"),
              row.names=colnames(pd0))
```

Notemos que la columna `labelDescription` debe de estar presente. Utilizamos la clase `Biobase::AnnotatedDataFrame` que nos permite tener los valores de las covariables y los metadatos que acabamos de introducir. Lo podemos hacer del siguiente modo,

```
datosfenotipo = new("AnnotatedDataFrame", data = pd0,
                    varMetadata = metadatos)
```

o bien con

[7] No es un ejemplo muy lucidor.

```
datosfenotipo = AnnotatedDataFrame(data = pd0,
                                   varMetadata = metadatos)
```

Una vez tenemos este `AnnotatedDataFrame` podemos ver los nombres de las muestras

```
head(sampleNames(datosfenotipo))
```

```
[1] "X600MPE" "AU565" "BT474" "BT483"
[5] "CAMA1" "DU4475"
```

O también ver los valores de las covariables (mostramos los dos primeros solamente).

```
head(pData(datosfenotipo),n=2)
```

```
       IDENTIFIER sample_no Type in_CGH ALL
X600MPE 600MPE 1 L 1 1
AU565 AU565 2 L 1 1
       CDH1_meth SFRP1 HER2 KIT known.p53 BRCA
X600MPE no 3.40987 <NA> NO NO <NA>
AU565 <NA> 3.16902 <NA> NO YES <NA>
       order adherence per3
X600MPE 1 <NA> 9
AU565 2 <NA> 4
```

Anotaciones y datos de las características

Sin duda este es el punto fundamental, la descripción de información sobre las filas de la matriz de expresión. Estas filas (features) corresponden con sondas. De hecho, distintas sondas corresponden con un mismo gen. Esta información depende del chip que utilizamos. Por ello distintos experimentos pueden corresponder a un mismo chip o instrumento. Es conveniente consultar http://www.bioconductor.org/packages/2.11/bioc/html/annotate.html. La información de cada chip aparece en un paquete de metadatos del mismo. En particular, nos proporcionan el nombre del gen, el símbolo y la localización en el cromosoma. Otros paquetes proporcionan información de Gene Ontology o KEGG.

Descripción del experimento

Los experimentos los hace la gente y hay que incluir esta información en los datos.

```
datosexperimento = new('MIAME',name='Neve et alt.',
                       lab='Varios',
                       contact ='rmneve@lbl.gov',
                       title = 'A collection of breast
cancer cell lines for the study of functionally distinct
cancer subtypes',
abstract = 'An example ExpressionSet',
url = '10.1016/j.ccr.2006.10.008',
other=list(notes='Creado a partir de ficheros de texto'))
```

El formato MIAME (Minimum information about a microarray experiment) puede consultarse en http://fged.org/projects/miame/.

Y montamos las piezas

Ejemplo de ExpressionSet a partir de los distintos elementos que hemos construido.[39]

[39] Esto es un poco como el Mecano.

```
neve06 = new("ExpressionSet",exprs=exprs0,
             phenoData = datosfenotipo,
             experimentData = datosexperimento,
             annotation = "hgu95av2")
```

Equivalentemente podemos usar

```
neve06 = ExpressionSet(assayData=exprs0,
                       phenoData = datosfenotipo,
                       experimentData = datosexperimento,
                       annotation = "hgu95av2")
```

Guardamos este ExpressionSet para uso posterior.

```
save(neve06,file="neve06.rda")
```

2.7 Ejemplos

Comentaremos algunos de los bancos de datos de DNA microarrays que utilizamos en el resto del curso. Son datos de expresión de genes obtenidos utilizando microarrays de DNA. Algunos van incorporados en los paquetes de R/Bioconductor como los datos `golub` ([115]) o los datos `ALL` [83] . Se utilizan como ejemplos en muchos paquetes y artículos. También se muestra en este tema el uso de la clase `Biobase::`
↪ `ExpressionSet`. Esta clase permite el manejo de datos de expresión obtenidos con microarrays de DNA. Veremos también cómo bajar datos de repositorios públicos.

2.7.1 golub

Son unos datos antiguos y que han sido usados con frecuencia como ejemplo en la primera literatura de microarrays. Una buena explicación de estos datos aparece en el manual de uso del paquete [115, multtest]. Estos datos fueron analizados por primera vez en [63]. En estas notas se utilizan en dos versiones. La primera es la incluida en el paquete [115]. La segunda corresponde a los datos incluidos en el paquete [**R-golubEsets**]. En el manual hablaremos de `multtest::golub` o bien `golubEsets::`
↪ `golub_Train`.

Hay muestras correspondientes a dos grupos o condiciones: 27 muestras (las primeras 27 columnas) correspondientes a pacientes con leucemia linfoblástica aguda (ALL, la clase 0) y leucemia mieloide aguda (AML, clase 1). Para medir los niveles de expresión se utilizaron chips de alta densidad de oligonucleótidos Affymetrix para 6817 genes. Los datos los podemos obtener a partir del paquete [115] con

```
data(golub,package= "multtest")
```

Según se indica en el manual de [115, multtest] se realizó una selección previa de genes de carácter no específico (no se utiliza por tanto información de las dos condiciones que pretendemos diferenciar). Esta selección no específica consistió en los siguientes pasos. Se trabajó con los

datos normalizados que se pueden encontrar en http://www.broadinstitute.org. Allí podemos encontrar una detallada explicación del protocolo experimental utilizado. Una vez obtenidos los datos normalizados se realizó la siguiente selección previa.

1. Una umbralización de los niveles de expresión. Valores por debajo de 100 se les asignó el valor 100. Los valores superiores a 16000 se les asignó el valor 16000.

2. Se calculó para cada gen el valor máximo y el valor mínimo de sus niveles de expresión para todas las muestras analizadas. Si denotamos por m_1 el mínimo y por m_2 el máximo entonces se eliminaron los genes que verifican
$$\frac{m_2}{m_1} \leq 5$$
o bien que
$$m_2 - m_1 \leq 500$$

3. Se tomó el logaritmo en base 10 para los niveles de expresión de los genes que no fueron eliminados.

Finalmente se estandarizaron los niveles de expresión dentro de cada array. Es decir, se calculó para cada array la media y la desviación estándar de todos los niveles observados en este array. A los datos originales se les restó la media y se dividió por la desviación estándar. Estos son los datos finales con los que trabajamos.[8] Una vez hemos cargado los datos golub tenemos la siguiente información:

golub Es una matriz 3051×38 de niveles de expresión donde las 3051 filas corresponden a los genes y las 38 columnas corresponden a las muestras. Podemos ver su primera fila:

```
golub[1,]
```

```
 [1] -1.45769 -1.39420 -1.42779 -1.40715 -1.42668
 [6] -1.21719 -1.37386 -1.36832 -1.47649 -1.21583
[11] -1.28137 -1.03209 -1.36149 -1.39979 0.17628
[16] -1.40095 -1.56783 -1.20466 -1.24482 -1.60767
[21] -1.06221 -1.12665 -1.20963 -1.48332 -1.25268
[26] -1.27619 -1.23051 -1.43337 -1.08902 -1.29865
[31] -1.26183 -1.44434 1.10147 -1.34158 -1.22961
[36] -0.75919 0.84905 -0.66465
```

golub.gnames Es una matriz 3051×3 con identificadores de los genes. Veamos la información de la primera fila correspondiente al primer gen.

```
golub.gnames[1,]
```

```
[1] "36"
[2] "AFFX-HUMISGF3A/M97935_MA_at (endogenous control)"
[3] "AFFX-HUMISGF3A/M97935_MA_at"
```

golub.cl Es un vector de longitud 38 con la clasificación del tipo de leucemia.

[8] Es importante darse cuenta de la enorme cantidad de transformaciones de la información original que estamos haciendo y que, por lo tanto, afecta de un modo fundamental las conclusiones que obtengamos en cualquier tratamiento estadístico posterior. Y lo peor: no sabemos de un modo claro cómo afectan estos resultados.

```
golub.cl
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[24] 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
```

Con frecuencia, y con objeto de mejorar la presentación de los resultados ejecutaremos la siguiente línea de código.

```
golub.cl = factor(golub.cl,levels= 0:1,labels=c("ALL","AML"))
```

Si volvemos a ver el contenido de golub.cl veremos el cambio producido.

```
golub.cl
```

```
 [1] ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL
[12] ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL
[23] ALL ALL ALL ALL ALL AML AML AML AML AML AML
[34] AML AML AML AML AML
Levels: ALL AML
```

[40] El script de R con todo el preprocesado descrito podemos encontrarlo en http://svitsrv25.epfl.ch/R-doc/library/multtest/doc/golub.R y en la documentación del paquete [115].

[40]

En el paquete [**R-golubEsets**] tenemos los datos originales. Tanto las muestras que se utilizaron para entrenar el procedimiento como las que se utilizaron para evaluarlo.

2.7.2 ALL

Para una versión ampliada de esta sección podemos consultar [67, capítulo 1]. Los datos ALL son microarrays de 128 individuos distintos con leucemia linfoblástica aguda (ALL). De estos individuos 95 corresponden a leucemia linfoblástica precursora aguda de células B y y 33 son leucemia linfoblástica precursora aguda de células T. Son enfermedades bastante distintas y por ello se consideran por separado. Habitualmente trabajaremos con las muestras de leucemia linfoblástica precursora aguda de células B. Los datos han sido preprocesados utilizando el método RMA (robust multichip average) implementado en el paquete [74, affy] con la función `affy::rma` y están almacenados en forma de un `Biobase` ↪ `::ExpressionSet`. Empezamos cargando los datos.

```
pacman::p_load(Biobase,ALL)
data(ALL)
```

En lo que sigue no vamos a utilizar todas las muestras. Un subconjunto de muestras de interés con dos grupos son los grupos de tumores de células B que tienen la mutación BCR/ABL y los tumores de las células B sin ninguna anormalidad citogenética. Veamos cómo seleccionar estas muestras. En primer lugar seleccionamos las muestras de células B.

```
bcell = grep("^B",as.character(ALL$BT))
```

Y ahora seleccionamos las muestras correspondientes a los tipos moleculares BCR/ABL o NEG.

```
types = c("NEG","BCR/ABL")
moltyp = which(as.character(ALL$mol.biol) %in% types)
```

Ahora combinamos ambas selecciones para quedarnos con los tumores de las células B y que tienen o bien la translocación BCR/ABL o bien no tienen ninguna de las anormalidades moleculares evaluadas.

```
bcrneg = ALL[,intersect(bcell,moltyp)]
```

Habitualmente haremos uso de los datos ALL realizando previamente esta selección. Remitiremos a esta sección para consultar el código anterior.

2.7.3 Un experimento con levadura

Los datos que comentamos en esta sección son un experimento en donde vamos a considerar dos tipos de células en levadura, salvaje y mutada. Los datos los tenemos en un `Biobase::ExpressionSet` en [12].

```
data(gse6647,package="tamidata")
```

El número de genes y muestras es

```
dim(gse6647)
```

```
Features Samples
   6103 8
```

Podemos ver los datos fenotípicos.

```
head(pData(gse6647),n=2)
```

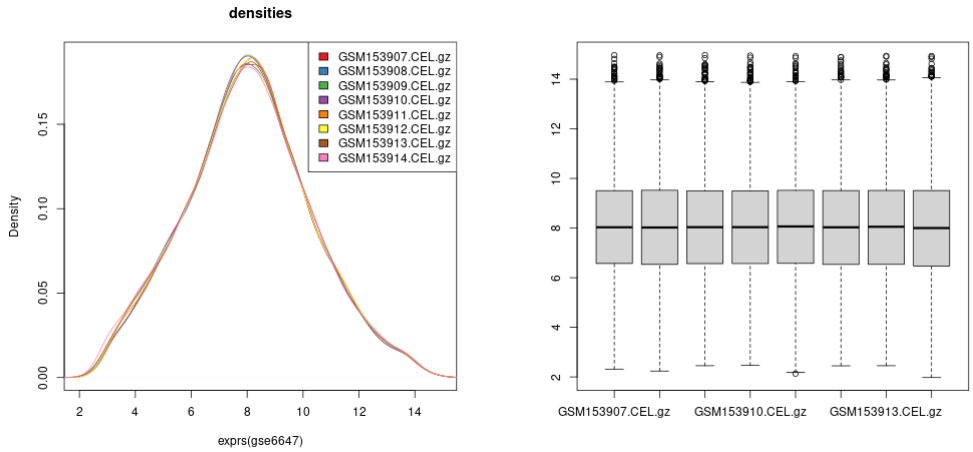
```
                type
GSM153907.CEL.gz wt
GSM153908.CEL.gz edc3D
```

Los datos han sido normalizados utilizando el método RMA. En la figura 2.7(a) mostramos las densidades estimadas.

```
geneplotter::multidensity(exprs(gse6647))
```

En la figura 2.7(b) tenemos los diagramas de cajas.

```
boxplot.matrix(exprs(gse6647))
```



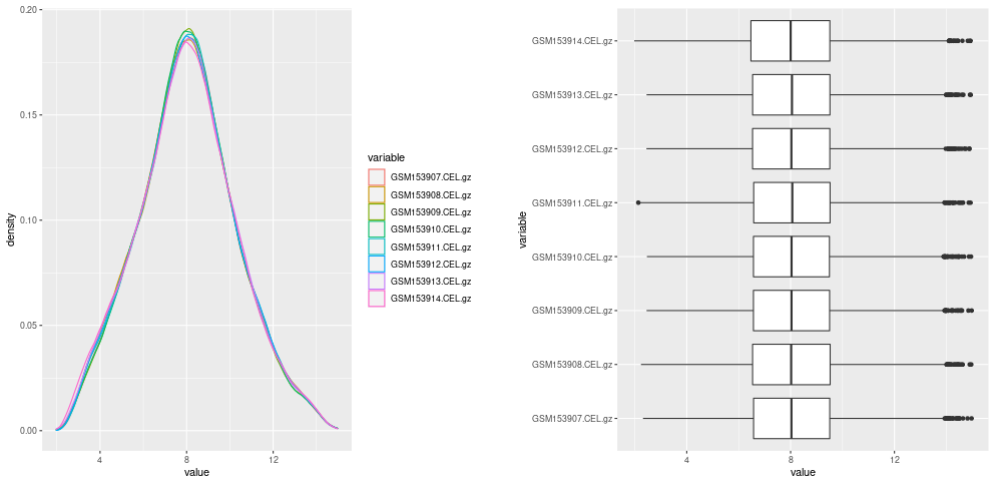
(a) (b)

Figura 2.7: Datos gse6647: estimadores de densidad y diagramas de cajas.

Vamos a reproducir los dibujos de la figura 2.7 utilizando el paquete [150, ggplot2].[41] El código es el que sigue y los tenemos en la figura 2.8.

[41] Sobre el paquete [147, reshape] es interesante consultar [148].

```
pacman::p_load(reshape,ggplot2)
df = data.frame(gene = featureNames(gse6647),exprs(gse6647))
df1 = melt(df,id=c("gene"))
ggplot(df1,aes(x=value,colour=variable,group=variable)) +
    geom_density(kernel = "epanechnikov",fill=NA)
ggplot(df1,aes(x=variable,y = value)) + geom_boxplot() +
    coord_flip()
```



(a) (b)

Figura 2.8: Datos gse6647: estimadores de densidad y diagramas de cajas utilizando [150, ggplot2].

2.7.4 El trabajo hecho

Hemos visto cómo construir un `Biobase::ExpressionSet` cuando tenemos los datos de expresión, las variables fenotípicas y la anotación. Podemos ver en [107] que los autores han subido esta información a ArrayExpress. Gran parte del trabajo hecho nos la podíamos ahorrar bajando los datos ya preparados de ArrayExpress con el paquete [77, ArrayExpress].

```
pacman::p_load(ArrayExpress)
rawset = ArrayExpress("E-TABM-157")
```

El objeto `rawset` es un `affy::AffyBatch` al cual podemos aplicar la corrección de fondo, la normalización y el resumen visto en § 2.

2.7.5 GSE21779

Utilizando GEO

Hemos elegido los datos que en GEO con identificador `GSE21779`. Podemos buscarlos utilizando la página web http://www.ncbi.nlm.nih.gov/geo/ o bien podemos acceder directamente a esta dirección http://www.ncbi.nlm.nih.gov/sites/GDSbrowser?acc=GDS4128. Bajamos los ficheros CEL (están en formato comprimido gzip).

Sabiendo el identificador de los datos podemos bajarlos con el R paquete [41, GEOquery].

```
library(GEOquery)
```

```
gcel = getGEOSuppFiles("GSE21779")
```

Los guarda en el subdirectorio `GSE21779`.[9] Cambiamos al directorio donde tenemos los datos.

```
setwd("GSE21779/")
```

Descomprimimos.

```
system("tar xvf GSE21779_RAW.tar")
```

Cargamos el R paquete [74, affy].

```
library(affy)
```

Y leemos todos los datos.

```
gse21779raw = ReadAffy()
```

Guardamos estos datos de expresión en un fichero externo.

```
save(gse21779raw,file = "gse21779raw.rda")
```

Podemos normalizar los datos con el método RMA.

```
gse21779rma = affy::rma(gse21779raw)
```

¿Qué datos fenotípicos tenemos?

```
pData(gse21779rma)
```

¿Como vemos no tenemos nada útil? Una posibilidad es bajar los datos ya procesados de GEO utilizando la función `GEOquery::getGEO()`.

```
gse = getGEO("GSE21779")
```

¿De qué clase es el objeto que hemos bajado?

```
class(gse)
```

```
[1] "list"
```

```
length(gse)
```

```
[1] 1
```

Vemos que es una `list` de longitud 1. ¿De qué clase es dicho elemento?

```
class(gse[[1]])
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Podemos ver sus variables fenotípicas (no mostradas).

```
pData(gse[[1]])
```

[9] A gusto del consumidor. Posiblemente para bajar un simple banco de datos lo mejor es utilizar el navegador.

Modificamos el `ExpressionSet`, `gse21779` que hemos obtenido después de normalizar y cambiamos su slot `pData`. Debemos comprobar que las muestras vienen en el mismo orden. Para ello empezamos quitando la extensión `.CEL.gz` de los nombres.

```
rownames(pData(gse21779rma)) =
    sapply(rownames(pData(gse21779rma)),function(x) unlist(strsplit(x,split=".CEL."))
        ↪ [1])
```

Y ahora podemos comprobar si efectivamente las muestras están en el mismo orden.

```
all(rownames(pData(gse21779rma)) == rownames(pData(gse[[1]])))
```

Ahora sustituimos el slot correspondiente.

```
pData(gse21779rma) = pData(gse[[1]])
gse21779 = gse21779rma
```

Utilizando ArrayExpress

Otra opción es bajar los mismos datos de ArrayExpress. Su código en esta base de datos es `E-GEOD-21779`. Lo hacemos con el siguiente código.

```
library(ArrayExpress)
geod21779raw = ArrayExpress("E-GEOD-21779")
```

La ventaja es que directamente nos devuelve un `AffyBatch`. Normalizamos los datos mediante el método RMA.

```
geod21779 = rma(geod21779raw)
```

Además podemos comprobar que las variables fenotípicas ya vienen incluidas.

2.7.6 GSE1397

[42] El uso de estos datos y parte del análisis aparecen en un documento que se puede encontrar en esta dirección. También he utilizado parte del análisis que propone Rodrigo Santamaría.

[42] El síndrome de Down es una enfermedad causada por la aparición de un copia extra total o parcial del cromosoma 21. Analizaremos si los genes de este cromosoma muestra una sobre expresión. Además se verá que solamente es específica de estos genes y no de otros que aparecen en otros cromosomas. Los datos se pueden obtener de GEO y el identificador del experimento es GSE1397. Los resultados que se obtienen son realmente bonitos e interesantes. Los datos a nivel de sonda (raw data) no están disponibles. Nos hemos de traer unos datos preprocesados. En el experimento se utilizaron arrays Affymetrix GeneChip U133A. Los obtenemos con

```
pacman::p_load("Biobase","GEOquery")
gse1397raw = getGEO("GSE1397")[[1]]
```

La anotación de estos datos es `GPL96` que (podemos comprobarlo entrando en GEO) corresponde con `Affymetrix Human Genome U133` chip hgu133a. El paquete que contiene sus datos de anotación es [30, hgu133a.db]. Cambiamos la anotación.

```
annotation(gse1397raw) = "hgu133a"
```

Los datos fenotípicos de las muestras los podemos ver con (no mostramos)

```
pData(gse1397raw)
```

Podemos ver las etiquetas de las covariables asociadas a las muestras con

```
varLabels(gse1397raw)
```

Fijémonos en la variable `type` que nos indica la alteración. En concreto puede ser trisomía 21 (síndrome de Down), trisomía 13 o euploide (ploidez normal) y el tejido puede ser cerebro, cerebelo, astrocito y corazón. Los valores de esta covariable la podemos ver con

```
head(pData(gse1397raw)[,"title"])
```

La matriz de expresión la tenemos (y no la mostramos) con

```
exprs(gse1397raw)
```

Vamos a seleccionar aquellas muestras que corresponden a cerebro o cerebelo y que son o bien TS21 o euploides.

```
(ts21=c(grep("T.*21.*cerebrum",
              as.character(pData(gse1397raw)[,"title"])),
        grep("TS21.*cerebellum",
              as.character(pData(gse1397raw)[,"title"]))))
(eu=c(grep("Euploid.*cerebrum)",
            as.character(pData(gse1397raw)[,"title"])),
      grep("Euploid.*[Cc]erebellum",
            as.character(pData(gse1397raw)[,"title"]))))
```

Efectivamente tenemos las columnas con las muestras que buscábamos

```
pData(gse1397raw)[eu,"title"]
```

```
pData(gse1397raw)[ts21,"title"]
```

Y ahora seleccionamos las muestras.

```
gse1397raw = gse1397raw[,c(eu,ts21)]
```

Modificamos los datos fenotípicos.

```
tissue = factor(rep(c(1,2,1,2),c(4,3,4,3)),levels=1:2,
  labels=c("Cerebrum","Cerebellum"))
type = factor(c(rep(1,7),rep(2,7)),levels=1:2,
                labels=c("Euploid","TS21"))
GROUP = as.numeric(type) - 1
pData(gse1397raw) = data.frame(tissue,type)
```

Guardamos los datos.

```
save(gse1397raw,file="gse1397raw.rda")
```

[43] Vamos a mostrar los estimadores kernel de la densidad en su escala original. Primero cargamos el paquete [21, affyPLM] que, entre otras cosas, nos permite obtener estos estimadores con facilidad.

```
library(affyPLM)
```

No tenemos los ficheros `.CEL` que necesitamos para preprocesar los datos. Por ello recurrimos a la función `normalizeBetweenArrays()` del paquete [128, limma].

```
library(limma)
gse1397 = gse1397raw
exprs(gse1397) = normalizeBetweenArrays(exprs(gse1397raw))
```

[43] El análisis que acabamos de ver desde bajar los datos hasta obtener el ExpressionSet lo podemos encontrar en http://www.uv.es/ayala/docencia/tami/Rmd/gse1397.html.

```
df = data.frame(gene = featureNames(gse1397),
                exprs(gse1397))
df1 = melt(df,id=c("gene"))
png(paste0(dirTamiFigures,"gse1397normden.png"))
ggplot(df1,aes(x=value,colour=variable,group=variable)) +
    geom_density(kernel = "epanechnikov",fill=NA) +
    xlim(0,1000)
dev.off()
```

Los autores introdujeron información adicional sobre las filas que podemos ver (no lo mostramos) con

```
fData(gse1397raw)
```

Es un `data.frame` que contiene las siguientes variables sobre los genes.

```
names(fData(gse1397raw))
```

2.7.7 GSE20986

El análisis de estos datos lo sugirió un estudio que se puede encontrar en http://bioinformatics.knowledgeblog.org/2011/06/20/analysing-microarray-data-in-bioconductor/. En lo que sigue utilizaremos parte del análisis que se propone allí. La referencia original del estudio es [26].

Empezamos obteniendo los datos. Tenemos los datos a nivel de sonda de modo que podemos realizar nuestro propio preprocesado. Una explicación detallada del protocolo experimental se puede ver en GEO consultando la información de una cualquiera de las muestras, por ejemplo, en http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSM524662. Podemos ver en esta página que el preprocesado de los datos para pasar de los datos a nivel de sonda a datos de expresión se realizó utilizando el método GC-RMA (en concreto utilizaron la función `gcrma::gcrma()` del paquete [156, gcrma].

La plataforma, código que le dan en GEO, a este chip es GPL570 que corresponde con el chip Affymetrix Human Genome U133 Plus 2.0 Array. Empezamos cargando paquetes necesarios.

```
pacman::p_load("Biobase","GEOquery")
```

Nos bajamos los ficheros `CEL`.

```
gcel = getGEOSuppFiles("GSE20986")
```

Nos ha creado el subdirectorio `GSE20986` por lo que hemos de cambiar el directorio de trabajo al nuevo directorio.

```
setwd("GSE20986/")
```

Además nos ha bajado un fichero tar del cual hemos de extraer los distintos ficheros `CEL`. En Linux lo haremos con[10]

```
system("tar xvf GSE20986_RAW.tar")
```

Leemos los datos.

```
gse20986raw = affy::ReadAffy()
```

Añadimos variables fenotípicas.

[10] En Windows podemos usar el Winrar.

```
tissue = factor(c(1,2,2,1,2,1,3,3,3,4,4,4),levels = 1:4,
                labels=c("iris","retina","choroides",
                         "huvec"))
pd = data.frame(tissue)
rownames(pd) = colnames(gse20986raw)
pData(gse20986raw) = pd
```

Guardamos los datos.

```
save(gse20986raw,file="gse20986raw.rda")
```

En la referencia original el preprocesado de los datos se realizó con el método GC-RMA [131, capítulo 3].

```
pacman::p_load("gcrma","hgu133plus2probe")
gse20986 = gcrma::gcrma(gse20986raw)
```

Y guardamos el nuevo `Biobase::ExpressionSet`. Los datos que hemos bajado con la función `getGEO` ya estaban preprocesados utilizando las mismas funciones que usamos aquí. Lo he hecho para mostrar cómo se hace o bien porque podemos querer utilizar algún otro procedimiento de preprocesado.

```
save(gse20986,file="gse20986.rda")
```

En este trabajo se comparan muestras obtenidas a partir de células endoteliales microvasculares de ojos humanos con células endoteliales obtenidas de venas umbilicales humanas (HUVEC). Del primer tipo de células hay tres subtipos dependiendo de dónde se extraen. En concreto se extrajeron del iris, la retina y la coroides. Las células HUVEC son más fáciles de obtener y por lo tanto de estudiar. Sin embargo, no es claro que lo que se estudie con ellas sea extrapolable con los resultados que se obtendrían con los otros tipos de células. En resumen, el objeto fundamental es la comparación de este grupo con los otros tres grupos.

2.7.8 GSE34764

Consideremos los datos de GEO con identificador GSE34764. Necesitamos el paquete [41, GEOquery].

```
library(GEOquery)
```

Obtenemos los ficheros CEL.

```
getGEOSuppFiles("GSE34764")
```

Cambiamos el directorio de trabajo.

```
setwd("GSE30129/")
```

Para leer los datos vamos a utilizar el paquete [33, oligo].

```
library(oligo)
celFiles = list.celfiles()
gse34774 = read.celfiles(celFiles)
save(gse34774,file="gse34774.rda")
```

2.7.9 GSE104645

Vamos a utilizar datos obtenidos utilizando un chip Agilent. Mostramos el proceso hasta producir un `ExpressionSet`. Los datos son un estudio sobre cáncer colorectal.[44] Podemos encontrar estos datos en https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE104645. Los bajamos con

[44] Siento que todo sea sobre cáncer pero es lo que hay.

```
wd = getwd()
setwd(dirTamiData)
GEOquery::getGEOSuppFiles("GSE104645")
setwd("GSE104645")
system("tar xvf GSE104645_RAW.tar")
system("gzip -d *.gz")
GSE104645raw = read.maimages(dir(".", "txt"), "agilent",
                                    green.only = TRUE,
                        other.columns="gIsWellAboveBG")
save(GSE104645raw,
       file=paste0(dirTamiData,"GSE104645raw.rda"))
setwd(wd)
```

El siguiente código lo comentamos en § 2.8 y trata del problema de las correspondencias múltiples.

```
a = AnnotationDbi::select(hgug4112a.db,
                          keys=GSE104645raw$genes[,"ProbeName"],
                          column=c("ENTREZID","ENSEMBL"),
                          keytype="PROBEID")
## Eliminamos sondas sin ENTREZID
a = a[!is.na(a[,"ENTREZID"]),]
c1 = match(unique(a[,1]),a[,1])
a1 = a[c1,]
c2 = match(unique(a1[,2]),a1[,2])
a2 = a1[c2,]
a2 = na.omit(a2)
GSE104645raw2 =
      GSE104645raw[match(a2[,1],
                              GSE104645raw$genes[,"ProbeName"]),]
dim(GSE104645raw2)
rownames(GSE104645raw2) = a2[,1]
save(GSE104645raw2,file=paste0(dirTamiData,
                                    "GSE104645raw2.rda"))
```

Hacemos la corrección de fondo.

```
GSE104645el = limma::backgroundCorrect(GSE104645raw2,
                                            method="normexp")
```

Aplicamos una normalización de cuantiles.

```
GSE104645el = limma::normalizeBetweenArrays(GSE104645el,
                                                 method="quantile")
```

Guardamos los datos.

```
save(GSE104645el,file=paste0(dirTamiData,"GSE104645el.rda"))
```

Contruimos el `Biobase::ExpressionSet`.

```
pd0 = data.frame(case = 1:ncol(GSE104645el))
rownames(pd0) = colnames(GSE104645el$E)
metadata = data.frame(labelDescription = c("case"),
                            row.names=colnames(pd0))
phenotypedata = new("AnnotatedDataFrame", data = pd0,
                          varMetadata = metadata)
experimentdata = new('MIAME',name="Okita A, Takahashi S,
  Ouchi K, Inoue M et al.",
  lab="Tohoku University Hospital",
```

```
 contact ="akira.okita.d8@tohoku.ac.jp",
 title = "Consensus molecular subtypes classification
 of colorectal cancer as a predictive factor for
chemotherapeutic efficacy against metastatic colorectal
cancer",
abstract = "Gene expression profile from 193
 formalin−fixed and paraffin embedded primary tumor
samples.",
url = "https://www.ncbi.nlm.nih.gov/pubmed/29721154",
other = list(notes = "An example of Agilent microarray"))

GSE104645 = new("ExpressionSet",exprs=GSE104645el$E,
                phenoData = phenotypedata,
                experimentData = experimentdata,
                annotation = "hgug4112a.db")
```

Añadimos el `ExpressionSet::fData` con los identificadores.

```
fData(GSE104645) = a2[,1:2]
save(GSE104645,file=paste0(dirTamiData,"GSE104645.rda"))
```

2.7.10 GSE168109

Estos datos fueron seleccionados y analizados por Ángela Roig Ferrer en el año 2024. Incluyo el código con ligeras modificaciones. Los datos los tenemos en GSE168109. Es un estudio con perros (Canis lupus familiaris) utilizando la tecnología Agilent-021193 Canine (V2) Gene Expression Microarray.

Tenemos 22 muestras procedentes de 15 individuos diferentes: 8 están sanos (control) y los otros 7 están enfermos. Los enfermos se estudian en dos condiciones diferentes: antes de ser tratados y 6 meses tras ser tratados mediante inmunoterapia específica con alérgenos (ASIT por sus siglas en ingles).

Bajamos los datos originales, procesamos el fichero tar y descomprimimos.

```
GEOquery::getGEOSuppFiles("GSE168109")
setwd("GSE168109/")
system("tar xvf GSE168109_RAW.tar")
system("gzip −d ∗.gz")
GSE168109raw = limma::read.maimages(dir(".", "txt"), "agilent",
                                    green.only = TRUE,
                                    other.column = "gIsWellAboveBG")
setwd("../")
system("rm −fr GSE168109")
```

Realizamos la anotación y eliminamos correspondencias múltiples.

```
pacman::p_load(org.Cf.eg.db)
a = AnnotationDbi::select(org.Cf.eg.db, keys = GSE168109raw$genes[,"GeneName"],
                          columns = c("ENTREZID", "ENSEMBL", "GENETYPE", "
                              ↪ GO",
                                  "ONTOLOGY", "UNIPROT"),
                          keytype="SYMBOL")
a$EVIDENCE = NULL
# Elimino las sondas que no tienen ENTREZID
a1 = a[!is.na(a[,"ENTREZID"]), ]
# Busco las posiciones de los SYMBOL que no se repiten
c1 = match(unique(a1[,1]), a1[,1])
# Me quedo con las filas de a1 sin repeticiones de SYMBOL
a2 = a1[c1,]
# Hago lo mismo para los ENTREZID
c2 = match(unique(a2[,2]), a2[,2])
a3 = a2[c2,]
```

```
# Guardo la información en un nuevo fichero
GSE168109raw2 = GSE168109raw[match(a3[,1], GSE168109raw$genes[,"GeneName"]),]
sondas = match(a3[,1], GSE168109raw$genes[,"GeneName"])
rownames(GSE168109raw2) = GSE168109raw$genes[sondas, "ProbeName"]
```

Hacemos corrección de fondo.

```
GSE168109el = limma::backgroundCorrect(GSE168109raw2, method = "normexp")
```

Aplicamos una normalización de cuantiles.

```
GSE168109 = limma::normalizeBetweenArrays(GSE168109el, method = "quantile")
```

2.7.11 GSE175728

Los datos fueron seleccionados y analizados por Patricia del Carmen Vaca Rubio en el año 2024. Lo incluyo con ligeras modificaciones en el código. Son datos relativos al virus de la inmunodeficiencia humana (VIH). El experimento es la serie GSE212331. Es un estudio publicado en 2023 para la comparación de los perfiles de expresión entre humanos con enfermedad pulmonar obstructiva crónica (EPOC, o por sus siglas en inglés, COPD) y personas sanas. El estudio consta de 87 muestras en cinco grupos: pacientes sanos (gold stage n/a), enfermos por EPOC de nivel 1 (gold stage 1), de nivel 2 (gold stage 2), de nivel 3 (gold stage 3) y de nivel 4 (gold stage 4). La gravedad de la enfermedad aumenta con el nivel. Son muestras de esputo inducido.

```
pacman::p_load("illuminaHumanv4.db")
```

```
GEOquery::getGEOSuppFiles("GSE212331")
setwd("GSE212331/")
gse212331raw = read.delim("GSE212331_non-normalized_data.txt.gz",
                          row.names=1, header=TRUE)
setwd("../")
system("rm -fr GSE212331")
```

Corregimos ruido de fondo y pasamos a datos logarítmicos en base 2.

```
gse212331raw = limma::backgroundCorrect(gse212331raw, method="normexp")
gse212331raw = log2(gse212331raw)
```

Aplicamos una normalización de cuantiles.

```
gse212331 =
    limma::normalizeBetweenArrays(gse212331raw, method="quantile")
```

2.8 Correspondencias múltiples

Necesitamos conocer la correspondencia entre sondas y genes para poder interpretar resultados o realizar análisis posteriores. Habitualmente recurrimos a paquetes de anotación que nos proporcionan la información de la correspondencia entre los identificadores del fabricante (`PROBEID`) y los identificadores en distintas bases de datos o denominaciones del gen (o entidad biológica que estemos utilizando). La opción más simple y primera a probar es utilizar un paquete de anotación. En Bioconductor es conveniente consultar https://www.bioconductor.org/packages/release/data/annotation/. En § 24 tratamos el problema del manejo de este tipo de paquetes.

En esta sección tratamos un problema al que se suele prestar poca atención pero es muy importante, las correspondencias múltiples. Distintas sondas corresponden a un mismo gen y distintos genes corresponden a una misma sonda. Esto es, no tenemos una biyección o correspondencia 1-1 entre sondas y genes. Y no es un problema menor. Vamos a ilustrar el problema con un ejemplo. Cargamos los datos bajados sin ningún tipo de normalización previa.

```
load(paste0(dirTamiData,"gse21779raw.rda"))
gse21779rma = affy::rma(gse21779raw)
```

¿Cuántas sondas tenemos después de la normalización?

```
nrow(gse21779rma)
```

```
Features
  54675
```

Establecemos las correspondencias con Entrez con `AnnotationDbi::select()`.

```
pacman::p_load(hgu133plus2.db)
a = AnnotationDbi::select(hgu133plus2.db,
                          keys=featureNames(gse21779rma),
                          columns=c("ENTREZID"),
                          keytype="PROBEID")
```

Vemos que tenemos más filas en `a` que en `gse21779rma`.

```
dim(a)
```

```
[1] 57151 2
```

Tenemos las correspondencias en forma de `data.frame`. ¿Cuántas y qué sondas tienen correspondencia con más de un gen?

```
b = which(table(a[,1]) > 1)
```

```
sel = is.element(a[,1],names(b))
table(table(a[sel,1]))
```

```
   2 3 4 5 6 7 8 9 10 11
1199 152 73 30 16 14 11 11 3 4
  12 13 14 15 17 21 22
   2 1 2 3 2 1 4
```

Una posibilidad puede ser quedarnos con la primera aparición.

```
c1 = match(unique(a[,1]),a[,1])
a1 = a[c1,]
```

Vamos a añadir esta información en los datos originales. Notemos que allí las sondas son únicas y estamos asignando la primera correspondencia que nos aparece.

```
fData(gse21779rma) = a1
```

¿Lo tenemos todo resuelto?

2.9 Ejercicios

* **Ex. 1 —** Consideramos los datos `tamidata::gse28619`. Se pide:

1. ¿Cuántas sondas tenemos?
2. ¿Cuántas muestras?
3. Las sondas de control empiezan con "AFFX": ¿Cuántas sondas de control tenemos?
4. ¿Cuántas variables fenotípicas tenemos? ¿De qué tipo son?
5. Indica el valor de expresión de la sonda en fila 2000 y columna 3.
6. Calcular para la sonda en la fila 100 la expresión media.
7. Calcular para la sonda en la fila 100 la expresión media en cada uno de los grupos definidos por la variable fenotípica `type`.
8. Determinar la sonda con una expresión media mayor?
9. Determinar la muestras con un nivel de expresión medio menor?
10. Consideremos el slot `fData`.
    (a) ¿Qué información tenemos en el slot `fData`?
    (b) ¿De qué clase es el slot `fData`?
    (c) ¿Cuántas columnas componen el slot `fData`?

* **Ex. 2 —** Consideramos los datos `tamidata::gse44456`. Repetir el ejercicio 1 con estos datos.

** **Ex. 3 —** Se pide construir un `Biobase::ExpressionSet`. Este `Biobase` ↪ `::ExpressionSet` ha de tener la siguiente matriz de expresión.

```
      [,1] [,2] [,3] [,4] [,5]
 [1,] 21.46 21.53 23.90 24.09 24.46
 [2,] 22.14 22.01 21.09 23.62 23.17
 [3,] 21.74 23.16 22.55 21.36 23.01
 [4,] 22.20 21.40 23.43 22.24 22.21
 [5,] 23.47 24.46 22.56 23.43 23.43
 [6,] 21.96 24.77 22.12 24.56 21.85
 [7,] 23.80 24.02 19.43 22.48 21.93
 [8,] 22.11 24.22 21.25 24.13 21.68
 [9,] 21.68 21.52 22.43 23.89 23.32
[10,] 23.10 23.44 22.60 21.56 22.29
[11,] 24.71 23.73 22.93 24.30 25.94
[12,] 23.74 24.16 20.91 22.77 23.80
[13,] 23.91 24.45 22.33 22.04 22.61
[14,] 25.03 22.55 21.63 24.72 23.18
[15,] 22.49 24.59 22.59 23.71 23.08
[16,] 23.90 20.81 23.09 23.79 25.32
[17,] 22.94 21.28 21.30 21.61 22.38
[18,] 22.87 21.67 21.81 22.73 23.12
[19,] 22.16 23.64 23.61 22.38 21.95
[20,] 23.74 24.69 20.60 22.89 22.38
```

Los nombres de las filas ha de ser

```
 [1] "g1" "g2" "g3" "g4" "g5" "g6" "g7"
 [8] "g8" "g9" "g10" "g11" "g12" "g13" "g14"
[15] "g15" "g16" "g17" "g18" "g19" "g20"
```

Los nombres de las muestras serán

```
 [1] "m 1" "m 2" "m 3" "m 4" "m 5" "m 6"
 [7] "m 7" "m 8" "m 9" "m 10" "m 11" "m 12"
[13] "m 13" "m 14" "m 15" "m 16" "m 17" "m 18"
[19] "m 19" "m 20"
```

Como datos fenotípicos (covariables que describen las columnas) han de ser las siguientes

```
  tipo crecimiento
1 1 0.30
2 2 0.23
3 2 0.34
4 1 0.24
5 2 0.45
```

** **Ex. 4 —** Se pide construir a partir de los datos `multtest::golub` un `Biobase::ExpressionSet`.

** **Ex. 5 —** [11] Vamos a bajar y preprocesar los datos `GSE30129` de GEO. Se pide realizar los siguientes pasos.

1. Bajar los datos a nivel de sonda utilizando el paquete [41, GEOquery] y la función `GEOquery::getGEOSuppFiles()`.
2. Leer los datos utilizando la función `oligo::read.celfiles()`.
3. En el paso anterior si no tenemos instalado el paquete `pd.mogene.1.0.st.v1` nos dará un aviso.
4. Instalar el paquete indicado y repetir la lectura.
5. Representar los estimadores de densidades y los diagramas de cajas de las expresiones a nivel de sonda.
6. Obtener los MAplots o dibujos media-diferencia de Tukey.
7. Aplicar el método RMA.
8. Repetir los dibujos de los apartados 5 y 6 para los datos procesados utilizando el método RMA.

* **Ex. 6 —** Consideremos los datos `tamidata::gse20986raw`.

1. ¿Cómo identifica los genes de las filas?
2. Cambiar los identificadores de los genes a sus códigos Ensembl.

[11] Este problema es muy similar a lo que hacemos en § 2.7.8.

Capítulo 3

RNA-seq

3.1 Introducción

En este tema trabajamos con datos obtenidos mediante la técnica conocida como RNA-Seq. En http://rnaseq.uoregon.edu/ tenemos una introducción muy simple y clara. Una buena visión general la tenemos en [112]. Un texto general que trata lo relativo al análisis de este tipo de datos es [80]. Una referencia breve pero de interés es [39] donde también da una visión global del análisis de este tipo de información.

En este manual se asume que se trabaja con un genoma de referencia. Los procedimientos que vamos a estudiar asumen este punto de partida.

De un modo análogo a § 2 comentaremos este tipo de información de expresión génica. En este capítulo mostramos algunos conceptos básicos y algunas herramientas útiles (y suficientes) para su manejo. El software y las posibilidades que ofrece es enorme y fuera del alcance de este manual (y de su autor). Pretendemos ofrecer una opción (o dos) sencillas y buenas para poder bajar los datos de un experimento y preprocesarlos hasta obtener la matriz de conteos. Una vez tengamos esa matriz de conteos y las variables fenotípicas podremos analizar nuestros datos que es el objetivo fundamental de este manual.

Una referencia de mucha utilidad para consultar cómo se obtienen y preprocesan este tipo de datos es [65].

3.2 Flujo de trabajo con RNA-seq

Un estudio de este tipo implica la realización de los siguientes pasos ([112]):

1. Diseño experimental.
2. Protocolos de extracción del RNA.
3. Preparación de las librerías. Se convierte el RNA en cDNA y se añaden los adaptadores para la secuenciación.
4. Se secuencian las lecturas cDNA utilizando una plataforma de secuenciación.
5. Alineamiento de las lecturas secuenciadas a un genoma de referencia.
6. Resumen del número de lecturas alineadas a una región.

7. Normalización de las muestras para eliminar diferencias técnicas en la preparación.

8. Estudio estadístico de la expresión diferencial incluyendo en lo posible un modelo.

9. Interpretación de los resultados desde el punto de vista biológico.

En este manual estamos interesados fundamentalmente en el análisis de expresión diferencial. Obviamente el punto 1 es básico. Lo realiza el experimentador. En la medida en que vamos a trabajar con datos de experimentos ya realizados hablaremos de cuestiones de diseño aunque no de un modo central. La extracción del RNA, preparación de librerías y la secuenciación son cuestiones no tratadas aquí.

Sí que comentaremos cómo alinear las lecturas cuando disponemos de un genoma de referencia y de los pasos que siguen en el análisis. Sin embargo, este manual se quiere centrar en problemas estadísticos en el contexto de la Bioinformática. Por ello, toda la parte de alineamiento y conteo de las lecturas lo veremos de un modo sencillo insistiendo en cómo hacerlo y no en qué se hace. Nuestro trabajo empieza a partir del momento en que tenemos los conteos y nos interesamos por estudiar si estos conteos son significativamente distintos entre distintas condiciones biológicas. En [9] tenemos el análisis detallado con todos los pasos desde las lecturas originales hasta el análisis de expresión diferencial final. El interés del trabajo es que podemos reproducir un análisis completo. Además contiene enlaces para aquellas herramientas que se utilizan fuera de R/Bioconductor como bowtie2, samtools. En http://www.bioconductor.org/help/workflows/rnaseqGene/ tenemos todo un flujo en donde se analiza un experimento cuyos datos los tenemos en [**R-airway**].

En el diseño hemos de diferenciar entre **réplicas técnicas** en la que utilizamos una misma muestra biológica, un mismo procesado y los mismos protocolos de secuenciación. Es de esperar que las diferencias de lo que observamos en estas réplicas técnicas sean menores a las obtenidas con **réplicas biológicas** en las que se utilizan distintas muestras biológicas. En los experimentos RNA-seq solemos tener réplicas biológicas y raramente réplicas técnicas.

¿Cuántas muestras? Se trabaja con muestras de tamaños extremadamente pequeños. Obviamente **no** podemos esperar ver diferencias claras entre tratamientos. Además los experimentadores suelen proponer diseños complejos (con varios factores) intentando aprovechar sus pocas muestras[45]. Un diseño experimental siempre ha de ser simple. Este es un comentario aplicable a cualquier tipo de dato pero es particularmente importante con dato ómico. Sin duda, en general y en particular en este contexto, los diseños experimentales han de mantenerse lo más simples que se pueda.

[45] Es su dinero y tienen derecho a tirarlo como quieran.

¿Qué tipos de tratamiento hemos de realizar? Es necesaria una normalización previa de las muestras que permita controlar las variaciones técnicas.

1. Por ejemplo, las muestras proporcionadas por el secuenciador tienen distintas cantidades de DNA y esto da lugar a diferencias en el número total de lecturas secuenciadas y alineadas independientemente de las diferente expresión diferencial de un gen. El tratamiento debiera afectar solamente a una fracción de los genes eva-

luados. Si la diferencia se observa en todos los genes lo que vemos es un efecto del protocolo de trabajo.

2. También hay diferencias entre genes que no se deben a diferencias de expresión. En los protocolos estandar un gen largo es secuenciado con más frecuencia que un gen corto.

La clase que en R/Bioconductor debe de utilizarse para trabajar con datos RNA-Seq es `SummarizedExperiment::SummarizedExperiment`.

En este capítulo pretendemos responder (parcialmente) las siguientes preguntas.

1. ¿Dónde podemos obtener datos de secuenciación? ¿Qué repositorios podemos utilizar?

2. ¿Cómo podemos realizar búsquedas en los metadatos con objeto de obtener experimentos que tenga que ver con lo que me interesa?

3. ¿Cómo bajarlos?

4. ¿Cómo contar las lecturas alineadas sobre intervalos o uniones de intervalos sobre el genoma?

3.3 Repositorios

Los repositorios donde podemos encontrar este tipo de datos son NCBI, EMBL y DDBJ. Cada una de estas bases de datos puede ser consultada en línea y bajar los datos desde la propia página. Las direcciones son

NCBI **NCBI-SRA** http://www.ncbi.nlm.nih.gov/sra Posiblemente es la mayor de las bases de datos. En GEO también tenemos datos de secuenciación.

EBI ENA http://www.ebi.ac.uk/ena

DDBJ http://www.ddbj.nig.ac.jp

3.4 Formatos

En § 3.4.1 y § 3.4.2 comentamos distintos formatos para almacenar secuencias tanto de ácidos nucleicos como de proteínas.

3.4.1 Formato FASTA

El formato FASTA está basado en texto y se utiliza representar secuencias bien de nucleótidos bien de aminoácidos. Tanto unos como otros son representados por una sola letra. También tiene símbolos para representar un hueco (gap) o parada en la traducción o bien que no se sabe el nucleótido o aminoácido. Es muy simple. Tiene una línea que comienza con el símbolo $>$ al que sigue una descripción de la secuencia. En la siguiente línea empieza la secuencia de bases o aminoácidos. Se recomienda que no tener más de 80 columnas y se pueden tener todas las filas que se precisen.

3.4.2 Formato FASTQ

El formato FASTQ es el más popular para datos de secuencias. Consiste de cuatro líneas por lectura con la siguiente información:

1. La primera que comienza con el carácter @ y contiene el nombre de la secuencia con alguna descripción opcional de la misma.

2. La segunda línea contiene la secuencia con las letras que correspondan dependiendo del tipo (nucleótidos, aminoácidos).

3. La tercera línea que comienza con + contiene información opcional sobre la secuencia.

4. La cuarta línea cuantifica la confianza o calidad en la determinación de cada base recogida en la segunda línea. En § 3.4.3 se comenta el índice Phred y su codificación.

5.

Un ejemplo es el siguiente:

```
@SRR1293399.1 ILLUMINA-545855_0026_FC629BG:6:1:1022:5049 length=50
ACAGGGACGCCATCGAATCCGGATCNTNNNNNNNNNNNNANNNNNNNNNN
+SRR1293399.1 ILLUMINA-545855_0026_FC629BG:6:1:1022:5049 length=50
dee\edYcdc`bbY`S]bb_]]Ua^BBBBBBBBBBBBBBBBBBBBBBBB
```

3.4.3 Phred

¿Cómo se cuantifica la confianza o precisión o calidad de la cada una de las bases que tenemos en la secuencia?[46] Se utiliza el programa Phred.[47] Este programa lo que hace es asignar los picos de fluorescencia a una de las cuatro bases (o *base call*). En [47] tenemos la explicación del método de asignación. Si P denota la probabilidad[48] para una base dada de ser mal asignada o clasificada entonces el valor con el que se trabaja es

$$Q = -10 \log_{10} P. \tag{3.1}$$

Esto significa que una probabilidad P muy pequeña de clasificación incorrecta se traduce en un valor grande de Q. Por ejemplo, una probabilidad de clasificación incorrecta de 0.01 corresponde con un valor de Q de

```
−10 ∗ log10(0.01)
```

```
[1] 20
```

Supongamos que tenemos la siguiente secuencia con sus correspondientes Q valores.

```
          C G A
5.413022e-04 1.466502e-04 3.559214e-05
          C A G
9.127808e-04 4.049188e-04 5.956737e-05
          G T T
2.221953e-05 6.478379e-04 7.081632e-04
          C
2.620369e-04
```

Los correspondientes Q valores serían

[46] Observar que en ningún momento utilizo (tomo en vano) la palabra Probabilidad.

[47] El procedimiento que utiliza aparece en [48] y cómo estima las probabilidades de error cuando asigna cada base aparece en [47].

[48] No es realmente una probabilidad. Es una cuantificación de la calidad de la asignación y no más.

```
 C G A C A G G T T C
33 38 44 30 34 42 47 32 31 36
```

Si tenemos que guardar estos valores tendríamos que almacenar los dos dígitos y el blanco que los separa. Mucha memoria cuando tenemos muchas bases a almacenar. En lugar de registrar las probabilidades de clasificación incorrecta o su transformación Q lo que se utiliza es la codificación Sanger. Consiste en guardar el valor ASCII correspondiente a la posición $33 + Q$. Los caracteres ASCII correspondientes a los valores del 33 al 100 los podemos obtener con

```
intToUtf8(33:100)
```

```
[1] "!\"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ[\\]^_`
    ↪ abcd"
```

En particular, para el ejemplo previo tendríamos la siguiente codificación.

```
[1] "BGM?CKPA@E"
```

A lo largo del tiempo el valor 33 que corresponde con $Q = 0$ tomó diferentes valores. En principio, solamente para datos antiguos podemos encontrar que lo dicho no es válido (por ejemplo, sumando 64 en lugar de 33 o no utilizando ciertos valores). Claramente la codificación indicada no está diseñada para que nosotros la leamos sino como una forma de ahorrar memoria.

3.5 Alineamiento

Tratamos el problema del alineamiento de lecturas cortas sobre genoma de referencia. De un modo más amplio el problema del alineamiento de una secuencia sobre una referencia es fundamental en Bioinformática. En https://en.wikipedia.org/wiki/Sequence_alignment tenemos un buen resumen del problema. En este capítulo estamos interesados en el problema del alineamiento de una secuencia corta sobre una larga de referencia.

Tomamos una lectura o secuencia corta y pretendemos encontrar en una secuencia larga donde es **más** similar, en que punto hay una mayor similitud respecto de la secuencia larga o de referencia. ¿Cuál es el resultado que obtenemos después de alinear? La secuencia corta es colocada sobre una parte de la secuencia de referencia indicando en qué puntos se produce una correspondencia e indicando los huecos[49] que se han tenido que introducir en una u otra de las secuencias para conseguir la mejor correspondencia posible entre dichas secuencias.

[49] Gaps.

```
Read:      GACTGGGCGATCTCGACTTCG
           |||||  |||||||||| |||
Reference: GACTG--CGATCTCGACATCG
```

Mediante la línea vertical mostramos en qué puntos hay una correspondencia correcta mientras que con un guión en una u otra secuencia indicamos en qué posiciones hemos debido de incorporar un hueco para poder hacer corresponder ambas secuencias.

Notemos que no estamos seguros de que efectivamente la secuencia corta se haya originado en este punto. Es una hipótesis. La mejor que se

puede conjeturar pero no es seguro ni mucho menos. En muchas ocasiones no se puede encontrar una correspondencia.

Dos tipos de alineamientos se pueden consideradar: **alineamiento de extremo a extremo** o alineamiento global[50] y alineamiento local.[51] En la primera opción toda la secuencia corta es alineada. En el segundo tipo permitimos que uno o los dos extremos contengan bases no alineadas.

[50] End-to-end alignment.

[51] Local alignment.

3.5.1 Bowtie2

Es conveniente consultar http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml. Es una herramienta software para alinear lecturas cortas sobre secuencias de referencia largas. Soporta distintos modos de alineamiento: con huecos[52], locales y con lecturas apareadas[53]. La salida del programa la realiza en formato SAM. El programa `bowtie2` utiliza un fichero de índices de clase Bowtie2 y ficheros con lecturas secuenciadas y produce un fichero en formato SAM.

[52] gapped

[53] Paired-end

En http://bowtie-bio.sourceforge.net/bowtie2/manual.shtml se tiene una muy buena exposición de todas las opciones que ofrece este programa y es más que recomendable su lectura.

En Bowtie2 permite alineamientos locales y globales: la opción local se indica `--local` para el alineamiento local mientras que la opción global se asume por defecto.

Hemos de evaluar la calidad del alineamiento. Para ello definimos una puntuación asociada a cada posible correspondencia. Lo que podemos llamar una función objetivo. La idea es que cuanto mayor sea la función objetivo mejor es el alineamiento. Realmente penalizamos cada error en la correspondencia. Tendremos una penalización por correspondencia errónea, una penalización por cada hueco. En el caso de alineamiento local sumaremos un valor por cada correspondencia. Finalmente el valor de la función objetivo nos cuantifica la calidad final de la correspondencia conseguida. En concreto se consideran los siguientes valores que se suman (bajando o subiendo) para calcular el valor de la función objetivo.

–ma Sumamos un valor positivo por cada correspondencia (para alineamiento local).

–mp Penalización (restamos) por una correspondencia errónea.

–np Penalización por tener un valor N (esto es una base no especificada) en la secuencia corta o en la larga de referencia.

–rdg Penalización por hueco en secuencia corta.

–rfg Penalización por hueco en secuencia de referencia.

Obviamente mediante algún procedimiento encontraremos siempre una correspondencia. No necesariamente esta correspondencia la hemos de considerar suficientemente buena. ¿Cuándo la consideraremos aceptable? Lo que se hace es fija un umbral por debajo del cual el alineamiento no se acepta. Este umbral lógicamente ha de depender de la longitud de la lectura corta. El valor de este umbral que por defecto (aunque es configurable) tiene definido Bowtie2 es, para alineamiento global,

$$-0.6 - 0.6 \cdot L, \tag{3.2}$$

siendo L la longitud de la lectura. Para alineamiento local el valor es

$$20 + 8 \cdot ln(L). \tag{3.3}$$

Lo podemos configurar con la opción `--score-min`.

¿Podemos alinear de un modo único? Por supuesto que no. Pensemos en zonas de la secuencia en donde se repiten mucho los elementos. Si tenemos secuencias cortas entonces podremos alinear (perfectamente) de distintos modos. Tendremos distintos alineamientos igualmente buenos. Se plantea el problema de elegir entre estos alineamientos. Esto se puede hacer utilizando las medidas de calidad de la correspondencia.

¿Qué son lecturas apareadas?[54] Son pares de lecturas cortas o secuencias cortas de las cuales conocemos a priori su posición relativa y la distancia que las separa en la lectura de DNA. Cuando tenemos lecturas apareadas tendremos dos ficheros que incluyen el primer y el segundo elemento del par, uno en cada fichero. En concreto la misma línea en cada fichero indica cada una de las componentes del par de lecturas. En Bowtie2 se utilizan los argumento `-1` y `-2` para indicar el primer y segundo fichero. Cuando se realiza el alineamiento de un par de secuencias cortas en el fichero SAM resultante se indica en los campos `RNEXT` y `PNEXT` el nombre y la posición del otro elemento del par. También se indica la longitud del fragmento de DNA del cual se secuenciaron los dos fragmentos. Cuando se realiza el alineamiento de un par de lecturas este se puede producir de un modo concordante o de un modo discordante. En un modo concordante los dos elementos del par se alinean verificando lo que se espera en orientación y distancia entre ellos. El alineamiento es discordante entonces tenemos para cada uno **un alineamiento único** pero no se verifican las restricciones de orientación y distancia. Por defecto, Bowtie2 busca tanto alineamientos concordantes como alineamientos discordantes. Con la opción `--no-discordant` solamente busca concordantes.

[54] Paired-end o mate-pair. No es lo mismo.

En la opción por defecto, Bowtie2 realiza un alineamiento mixto. En esta opción primero busca alinear el par de lecturas de un modo concordante. Si no lo logra entonces busca alinear cada una de ellas de un modo discordante. Si no queremos la segunda parte de la búsqueda podemos eliminarla con la opción `--no-mixed`.

Bowtie2 busca alineamientos válidos para cada lectura. Cuando encuentra un alineamiento válido sigue buscando otros que sean al menos tan buenos como el que ha encontrado. Tiene unos controles para dejar de buscar. Cuando cesa la búsqueda utiliza los mejores alineamientos que ha encontrado para evaluar la calidad de la correspondencia encontrada (el campo `MAPQ` en el formato SAM).

3.5.2 STAR

En § 3.6 proponemos dos flujos de trabajo para obtener la matriz de conteos a partir de los ficheros de lecturas originales. El segundo alineador que proponemos es STAR. Sin duda, una de las opciones más rápidas y utilizadas. En https://github.com/alexdobin/STAR tenemos información adicional. En Debian/Ubuntu podemos instalarlo con

```
apt install rna-star
```

3.6 Una opción

En esta sección se propone un flujo completo utilizando la base de datos NCBI-SRA, los programas de alineamiento STAR **o** Bowtie2, Samtools y finalmente algunos paquetes de R. Nuestro objetivo es bajar de la red un experimento de RNA-Seq y obtener la matriz de conteos de la forma más simple que podamos.

¿Qué necesitamos? Mucha paciencia y asegurarse de que tenemos (muchísima) memoria disponible en nuestro disco. Hemos de tener instalado SRA-Toolkit.

Experimento Elegimos un experimento en la base de datos NCBI-SRA (https://www.ncbi.nlm.nih.gov/sra). En el ejemplo que desarrollamos hemos seleccionado el experimento con `accession` PRJNA218851 en NCBI-SRA.

Obtención de los identificadores, datos y metadatos

1. En https://www.ncbi.nlm.nih.gov/bioproject/PRJNA218851/ tenemos el menú (final de la página) denominado `SRA Run Selector` y que en nuestro caso corresponde con el enlace https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA218851.
2. En https://www.ncbi.nlm.nih.gov/Traces/study/?acc=PRJNA218851 tenemos información sobre el experimento. Con la pestaña `Select` en donde podemos seleccionar información de toda la muestra o bien de una parte. En el enlace `Accession List` generamos un fichero texto `SRR_Acc_List.txt` en donde aparecen los nombres de los `Run`, en resumen, los nombres de los datos que utilizamos a partir de ahora. Lo guardamos, por ejemplo, con dicho nombre.
3. En la misma página elegimos en la pestaña `Metadata` el fichero `SraRunTable.txt`. Contiene las variables fenotípicas o covariables para cada una de las muestras. Lo bajamos y guardamos con el mismo nombre.
4. Bajamos los datos con `prefetch` de SRA-Toolkit.

```
prefetch --option-file SRR_Acc_List.txt
```

   Los ficheros con las lecturas los guarda por defecto en `~/ncbi/public/sra/`.

5. En mi caso los he movido (nada de copiar) al directorio `PRJNA218851/sra`.

De NCBI-SRA a FASTQ Hemos de pasar de formato NCBI-SRA a formato FASTQ. Utilizamos la función `fastq-dump` de SRA-Toolkit.

```
fastq-dump -I --split-files nombre_fichero.sra
```

Y lo repetimos para cada fichero SRA. Una opción que nos lo hace para todos es[55]

```
cat files | parallel -j 7 fastq-dump --split-files {}.sra
```

donde `files` es el fichero `SRR_Acc_List.txt`, es decir, los nombres de los ficheros NCBI-SRA sin la extensión.

[55] Previamente hay que instalar *parallel* con *apt-get install parallel*.

Control de calidad Ver § 3.7.

Alineamiento Para alinear tenemos muchas herramientas software para hacerlo. Aquí ofrecemos dos opciones: utilizando STAR que es una opción más rápida aunque con mayores requerimientos informáticos o bien utilizando Bowtie2.

STAR **Generación de fichero de índices** Lo hacemos con el siguiente código donde hemos de sustituir `genomeDir0` por el directorio donde queremos guardar el fichero de índices (hay que poner el camino absoluto, no funciona con el relativo) y `GRCh38.p13.genome.fa` por el nombre del fichero fastq donde tenemos el genoma de referencia.

```
STAR --runMode genomeGenerate --
    ↪ runThreadN 10 --genomeDir
    ↪ genomeDir0 --genomeFastaFiles
    ↪ GRCh38.p13.genome.fa
```

Alineamiento de las secuencias El siguiente ejemplo supone lecturas apareadas en los ficheros `f_1.fastq` y `f_2.fastq` y `name_file` es el nombre del fichero de salida.

```
STAR --genomeDir genomeDir0 --
    ↪ runThreadN 14 starIndex --
    ↪ readFilesIn f_1.fastq f_2.fastq
    ↪  --outFileNamePrefix name_file
```

Bowtie2 **Fichero de índices** Bajamos el fichero de índices para poder trabajar con Bowtie2. En http://support.illumina.com/sequencing/sequencing_software/igenome.html los tenemos para distintas especies. Descomprimimos el fichero de índices y sacamos los ficheros. Supongamos que lo hemos colocado en el directorio `indexDir0` y queremos alinear el fichero `name_file.fastq`.

```
bowtie2 -x indexDir0 -U name_file
    ↪ .fastq -S name_file_1.sam
```

SAM a BAM Utilizamos Samtools. Ejecutamos para cada fichero `name_file`.

```
samtools view -S -b name_file.sam > name_
    ↪ file.bam
```

Ordenamos las lecturas Sustituimos `name_file` por cada nombre original.

```
samtools sort name_file.bam -o name_file_
    ↪ sort.bam
```

Conteo Podemos usar el siguiente código de R.

```
pacman::p_load(Rsamtools,GenomicFeatures,
               GenomicAlignments,org.Hs.eg.db)
gtfFile = "/home/gag/DOCENCIA/tami-data2/GRCh38.p13.genome/
    ↪ gencode.v37.annotation.gff3"
```

```
txdb = makeTxDbFromGFF(gtfFile, format="gff3")
genes = exonsBy(txdb, by="gene")
indir = getwd()
files = list.files(indir, pattern = '*sort.bam')
bamLst = BamFileList(files, index=character(),
                     obeyQname=TRUE)
PRJNA411984 = summarizeOverlaps(features = genes,
                                read=bamLst,
                                mode="Union",
                                singleEnd=FALSE,
                                ignore.strand=TRUE,
                                fragments=FALSE)
save(PRJNA411984,file="PRJNA411984.rda")
```

En `tamidata::PRJNA266927`, `tamidata::PRJNA297664` y `tamidata::PRJNA297798` tenemos ejemplos completos del flujo de trabajo propuesto.

3.7 Control de calidad

Tenemos los ficheros FASTQ proporcionadas por el proceso de secuenciación y queremos evaluar la calidad de las lecturas. ¿Qué problemas podemos encontrarnos? Que las bases tengan un nivel de confianza bajo (que suele venir indicado con la letra N). O que tengamos un sesgo hacia ciertas secuencias (por ejemplo, un sesgo a GC). Bien adaptadores que no se han eliminado de los datos por un deficiente tratamiento de los datos. Bien contaminación de la muestra (una muestra de una especie contaminada por otra especie). Hay muchas herramientas software para corregir estos problemas. Veamos algunas posibilidades.

3.7.1 FastQC

Los ficheros que vamos a evaluar los mantenemos en su formato comprimido `.gz`. En ocasiones estos problemas no pueden evitarse. En otros casos sí podemos corregirlos.

3.7.2 ShortRead

El paquete [100, ShortRead] permite trabajar con FASTQ. En particular, también podemos utilizarlo para realizar control de calidad.

```
pacman::p_load(ShortRead)
```

Vamos a ver sus funcionalidades. Utilizamos una muestra propia denominada `11a_S3_`. Para ello lo que hacemos es elegir una muestra aleatoria de 10000 lecturas.

En el fichero `11a_S3_fq0.fastq` tenemos la muestra para poder trabajar.

Podemos leer todo el fichero con el siguiente código. Leemos con `ShortRead::readFastq()`.

```
fq1 = readFastq(finput)
```

Ahora `fq0` es un ejemplo de clase

```
class(fq0)
```

```
[1] "ShortReadQ"
attr(,"package")
[1] "ShortRead"
```

En particular podemos ver la información básica con

```
fq0
```

```
class: ShortReadQ
length: 10000 reads; width: 35..75 cycles
```

Obviamente tiene solamente 10000 lecturas. Podemos seleccionar[56] del modo habitual. Por ejemplo, la primera sería

[56] Subsetting

```
fq0[1]
```

```
class: ShortReadQ
length: 1 reads; width: 75 cycles
```

O de la tercera a la décima con

```
fq0[3:10]
```

Con `ShortRead::sread()` podemos tener la lectura. La que ocupa la posición 1000 sería

```
sread(fq0[1000])
```

Podemos ver las primeras y últimas con

```
sread(fq0)
```

Podemos ver la longitud de las lecturas con

```
head(width(fq0))
```

```
[1] 75 74 75 75 73 75
```

Con `ShortRead::detail()` tenemos una información detallada. Las medidas de calidad de la lectura 1000 las obtenemos con

```
quality(fq0[1000])
```

```
class: FastqQuality
quality:
BStringSet object of length 1:
   width seq
[1] 75 /AA/AEEEEEEEEEEEEE...EEE/AAAEA/<<E//AEE
```

Podemos saber el valor a que corresponde la codificación de la calidad con

```
encoding(quality(fq0))
```

```
 ! "  #  $  %  &  '  (  )  *  +  ,  -  .  /  0
 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15
 1  2  3  4  5  6  7  8  9  :  ;  <  =  >  ?  @
16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
 A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P
32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47
 Q  R  S  T  U  V  W  X  Y  Z  [ \\  ]  ^  _  `
48 49 50 51 52 53 54 55 56 57 58 59 60 61 62 63
 a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p
64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79
 q  r  s  t  u  v  w  x  y  z  {  |  }  ~
80 81 82 83 84 85 86 87 88 89 90 91 92 93
```

Y ahora consideramos el control de calidad de cada muestra.

```
fls ←dir("/home/gag/DOCENCIA/tami−data2/EL21/data/Raw_data/",
    "*fastq.gz$", full = TRUE)
```

```
fls = dir("/home/gag/DOCENCIA/tami−data2/EL21/data/Raw_data/",
                "*fastq.gz$",full=TRUE)
```

Vamos a realizar un control de calidad del experimento con `ShortRead` ↪ `::qa`. Lo hacemos para la primera muestra pero hemos de repetirlo para cada una de las muestras.

```
fls1.qa = qa(fls[1],type="fastq")
```

Generamos un informe y lo visualizamos.[57]

[57] A los usuarios de Windows es probable que el codigo que sigue no les funcione. ¿Les extraña?

```
browseURL(report(fls1.qa))
```

Podemos ver la información contenida en `fls1.qa` con

```
show(fls1.qa)
```

Por ejemplo podemos ver el número de lecturas.

```
fls1.qa[["readCounts"]]
```

```
                  read filter aligned
10a_S8_.fastq.gz 35918050 NA NA
```

O también podemos ver las frecuencias de cada una de las bases.

```
fls1.qa[["baseCalls"]]
```

```
                     A C G
10a_S8_.fastq.gz 17979786 17875671 18298601
                     T N
10a_S8_.fastq.gz 19778581 26486
```

O las lecturas más frecuentes.

```
head(fls1.qa[["frequentSequences"]])
```

```
                                                          sequence
1
    ↪ GTCAGTATCATGCTGCGGCTTCAAATCCGAAGTGATGTTTTGATGTGAAGTGAAATTTTAGTTGTCGTAGTAGGC
    ↪
2
    ↪ GGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGGG
    ↪
3
    ↪ GTTCATGTTCGTCCTTTTGGTGTGTGGATTAGCATTATTTGTTTGATAATAAGTTTAACTAGTCAGTGTTGGAA
    ↪
4
    ↪ CTCGAATGTGTGATATGGTGGAGGGCAGCCATGAAGTCATTCTAAATTTGTTGAAGCATACGATACTGATATTAC
    ↪
5
    ↪ CTCATCAATAAATGGAGACGTATAGGAAAAGTCAGACTACGTCTACAAAATGTCAGTATCATGCTGCGGCTTCA
    ↪
6
    ↪ CTCCGATTAGGTGTATTAATAAGTGTCCTGCAGTAATGTTAGCTGTAAGCCGGACTGCTAATGCCATTGGTTGA
    ↪
  count type lane
1 1140 read 10a_S8_.fastq.gz
2 769 read 10a_S8_.fastq.gz
3 545 read 10a_S8_.fastq.gz
4 519 read 10a_S8_.fastq.gz
5 483 read 10a_S8_.fastq.gz
6 457 read 10a_S8_.fastq.gz
```

Podemos ver la frecuencia de cada base en cada posición de la lectura. Por ejemplo, para las dos primeras posiciones.

```
head(fls1.qa[["perCycle"]]$baseCall,n=10)
```

```
  Cycle Base Count lane
1 1 A 98412 10a_S8_.fastq.gz
2 1 C 488062 10a_S8_.fastq.gz
3 1 G 368285 10a_S8_.fastq.gz
4 1 T 44463 10a_S8_.fastq.gz
15 1 N 778 10a_S8_.fastq.gz
19 2 A 131713 10a_S8_.fastq.gz
20 2 C 212603 10a_S8_.fastq.gz
21 2 G 192729 10a_S8_.fastq.gz
22 2 T 462633 10a_S8_.fastq.gz
33 2 N 322 10a_S8_.fastq.gz
```

3.8 Ejemplos

En esta sección mostramos distintos bancos de datos bien contenidos en paquetes de Bioconductor (§ 3.8.1) o bien que podemos obtener de bases de datos en línea.

3.8.1 parathyroidSE::GSE37211

Estos datos corresponden al experimento con número de acceso GEO `GSE37211`. En la viñeta del paquete [85, parathyroidSE] tenemos una descripción detallada para obtener los conteos a nivel de gen o de exon partiendo de los ficheros originales NCBI-SRA. Ya los tenemos en el paquete [85, parathyroidSE].

```
data(parathyroidGenesSE,package="parathyroidSE")
```

Son datos relativos a los genes. ¿Qué clase tenemos?

```
class(parathyroidGenesSE)
```

```
[1] "RangedSummarizedExperiment"
attr(,"package")
[1] "SummarizedExperiment"
```

Es un `SummarizedExperiment::RangedSummarizedExperiment` y será nuestra clase de referencia cuando trabajamos con datos de RNA-Seq.[58] Para poder trabajar con esta clase cargamos el paquete.

[58] Es muy conveniente leer la viñeta de [104, SummarizedExperiment].

```
pacman::p_load(SummarizedExperiment)
```

¿Cuántos genes y muestras tenemos?

```
dim(parathyroidGenesSE)
```

```
[1] 63193 27
```

La matriz con los conteos nos la da `GenomicRanges::assay()` Es la análoga a `Biobase::exprs()` para `Biobase::ExpressionSet`..

```
head(assay(parathyroidGenesSE),n=2)
```

```
              [,1] [,2] [,3] [,4] [,5] [,6]
ENSG00000000003 792 1064 444 953 519 855
ENSG00000000005 4 1 2 3 3 1
              [,7] [,8] [,9] [,10] [,11] [,12]
ENSG00000000003 413 365 278 1173 463 316
ENSG00000000005 0 1 0 0 0 0
              [,13] [,14] [,15] [,16] [,17]
ENSG00000000003 987 424 305 391 586
ENSG00000000005 0 0 0 0 0
              [,18] [,19] [,20] [,21] [,22]
ENSG00000000003 714 957 346 433 402
ENSG00000000005 0 1 0 0 0
              [,23] [,24] [,25] [,26] [,27]
ENSG00000000003 277 511 366 271 492
ENSG00000000005 0 0 0 0 0
```

Tenemos los metadatos de las columnas o variables fenotípicas o covariables asociadas a las muestras con `GenomicRanges::coldata()`. Corresponde con `Biobase::pData()`.

```
colData(parathyroidGenesSE)
```

Es un `S4Vectors::DFrame`.

```
class(colData(parathyroidGenesSE))
```

```
[1] "DFrame"
attr(,"package")
[1] "S4Vectors"
```

Por ejemplo, podemos ver los nombres de las variables fenotípicas con

```
names(colData(parathyroidGenesSE))
```

```
[1] "run" "experiment" "patient"
[4] "treatment" "time" "submission"
[7] "study" "sample"
```

Y acceder a los valores de la variable `treatment` con

```
colData(parathyroidGenesSE)[,"treatment"]
```

```
 [1] Control Control DPN DPN OHT
 [6] OHT Control Control DPN DPN
[11] DPN OHT OHT OHT Control
[16] Control DPN DPN OHT OHT
[21] Control DPN DPN DPN OHT
[26] OHT OHT
Levels: Control DPN OHT
```

o bien con (no mostrado)

```
colData(parathyroidGenesSE)$treatment
```

La información en las filas en este caso corresponde con genes.

```
rowRanges(parathyroidGenesSE)
```

```
GRangesList object of length 63193:
$ENSG00000000003
GRanges object with 17 ranges and 2 metadata columns:
      seqnames ranges strand |
         <Rle> <IRanges> <Rle> |
   [1] X 99883667-99884983 - |
   [2] X 99885756-99885863 - |
   [3] X 99887482-99887565 - |
```

```
  [4] X 99887538-99887565 - |
  [5] X 99888402-99888536 - |
  ... ... ... ... .
 [13] X 99890555-99890743 - |
 [14] X 99891188-99891686 - |
 [15] X 99891605-99891803 - |
 [16] X 99891790-99892101 - |
 [17] X 99894942-99894988 - |
       exon_id exon_name
      <integer> <character>
  [1] 664095 ENSE00001459322
  [2] 664096 ENSE00000868868
  [3] 664097 ENSE00000401072
  [4] 664098 ENSE00001849132
  [5] 664099 ENSE00003554016
  ... ... ...
 [13] 664106 ENSE00003512331
 [14] 664108 ENSE00001886883
 [15] 664109 ENSE00001855382
 [16] 664110 ENSE00001863395
 [17] 664111 ENSE00001828996
 -------
 seqinfo: 580 sequences (1 circular) from an unspecified genome

...
<63192 more elements>
```

Por tanto, las filas de un `SummarizedExperiment` es un `GenomicRanges`
↪ `::GRangesList` de modo que cada fila es un `GenomicRanges::GRanges`
↪ indicando los exones que se utilizaron para contar las secuencias de RNA. Los nombres de los genes los podemos obtener con

```
head(rownames(parathyroidGenesSE))
```

3.8.2 GSE64099

Bajamos los datos de GEO.

```
wget ftp.ncbi.nlm.nih.gov/geo/series/GSE64nnn/GSE64099/
    ↪ suppl/GSE64099_RAW.tar
tar xvf GSE64099_RAW.tar
gzip -d *
rm GSE64099_RAW.tar
```

Se tienen las siguientes muestras:

```
GSM1564328_2086.txt  GSM1564331_241.txt   GSM1564329_2433.txt
GSM1564332_1932.txt  GSM1564327_1934.txt  GSM1564330_3225.txt
GSM1564333_1940.txt
```

Construimos un `Biobase::ExpressionSet`.

```
library(Biobase)
exprs0 <-CountAll
rownames(exprs0) <-EntrezAll
colnames(exprs0) <-c("GSM1564328", "GSM1564331", "GSM1564329",
    "GSM1564332", "GSM1564327", "GSM1564330", "GSM1564333")
type <-c(2, 1, 2, 1, 1, 2, 2)
type <-factor(type, levels = 1:2, labels = c("Wild-type",
    "Smchd1"))
type <-data.frame(type)
rownames(type) <-colnames(exprs0)
datosfenotipo <-new("AnnotatedDataFrame", data = type)
sampleNames(datosfenotipo)
```

```
datosexperimento ←new("MIAME", name = "GSE64099",
    lab = "Molecular Medicine Division, The Walter and
Eliza Hall Institute of Medical Research",
    contact = "mritchie@wehi.edu.au", title = "Transcriptome profiling for genes
        ↪ transcriptionally
regulated by Smchd1 in lymphoma cell lines",
    url = "http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE64099")
gse64099 ←new("ExpressionSet", exprs = exprs0, phenoData = datosfenotipo,
    experimentData = datosexperimento, annotation = "MusMusculus")
```

Eliminamos los genes que no tienen ninguna lectura alineada. Todos estos genes no tienen interés cuando estudiemos la posible expresión diferencial.

```
nullsum = apply(exprs(gse64099),1,sum)==0
gse64099 = gse64099[!nullsum,]
```

3.8.3 GSE63776

En ocasiones los autores del estudio original han puesto a nuestra disposición los conteos. En esta sección mostramos cómo utilizar un `ExpressionSet` ↪ para almacenar la información. No es la opción mejor (la clase `SummarizedExperiment` ↪ es mejor opción) pero es válida. Tomamos como ejemplo los datos GSE63776.

1. Los bajamos de http://www.ncbi.nlm.nih.gov/geo/download/?acc=GSE63776&format=file&file=GSE63776%5FPANC1%5Fcounts%2Etxt%2Egz.[59]

2. Descomprimimos el fichero `GSE37211_count_table.txt.gz`.

3. Abrimos el fichero con un editor[60] y añadimos al principio de la primera línea la palabra `Gene`. Guardamos el fichero con la modificación.

4. Leemos los datos.

```
x = read.table(file="GSE63776_PANC1_counts.txt",
               header=TRUE)
```

5. La primera columna es el nombre del gen por lo que la eliminamos para quedarnos con la matriz de expresión (conteos).

```
exprs0 = as.matrix(x[,−1])
rownames(exprs0) = x[,"Gene"]
```

6. Construimos los metadatos o datos fenotípicos.

```
type = factor(rep(1:2,each=3),levels=1:2,
              labels=c("siControl","siTCF7L2"))
pd0 = data.frame(type)
rownames(pd0) = colnames(exprs0)
datosfenotipo = new("AnnotatedDataFrame", data = pd0)
datosexperimento = new('MIAME',name='GSE63776',
                       lab='Seth Frietze and Farnham P',
                       contact ='seth.frietze@unco.edu',
                       title = ' ',abstract = 'We have compared the
genome−wide effects on the transcriptome after
treatment with ICG−001 (the specific CBP
inhibitor) versus C646, a compound that competes
with acetyl−coA for the Lys−coA binding pocket of
both CBP and p300. We found that both drugs cause
```

[59] Casi mejor poner el código de acceso GSE63776 en Google y tenemos la dirección.

[60] Bloc de nota, gedit, emacs

```
large−scale changes in the transcriptome of HCT116
colon cancer cells and PANC1 pancreatic cancer
cells, and reverse some tumor−specific changes in
gene expression. Interestingly, although the
epigenetic inhibitors affect cell cycle pathways
in both the colon and pancreatic cancer cell lines,
the WNT signaling pathway was affected only in
the colon cancer cells.
Notably, WNT target genes were similarly
down−regulated after treatment
of HCT116 with C646 as with ICG−001.',
url='http://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE63776',
other = list(notes = 'Public on Jan 07, 2015'))
gse63776 = new("ExpressionSet",exprs=exprs0,
                phenoData = datosfenotipo,
                experimentData = datosexperimento,
                annotation = "Illumina HiSeq 2000 (Homo sapiens)")
save(gse63776,file="gse63776.rda")
```

3.8.4 TCGA

La base de datos del TCGA contiene estudios de RNA-Seq. Se pueden bajar los datos utilizando GDC Data Transfer Tool. El paquete [**R-TCGAbiolinks**] facilita el uso de esta enorme base de datos. Además podemos conseguir datos de tipo muy distinto. En esta sección lo hacemos para bajar datos de RNA-Seq.

Datos TCGA-COAD

En esta sección vamos a utilizar, entre otros, datos procedentes de TCGA. Empezamos bajando los datos.

```
pacman::p_load(TCGAbiolinks)
query ←GDCquery(
  project = "TCGA−COAD",
  data.category = "Gene expression",
  data.type = "Gene expression quantification",
  file.type="normalized_results",
  platform = "Illumina HiSeq",
  experimental.strategy = "RNA−Seq",
  legacy = TRUE
)
GDCdownload(query, method = "api", files.per.chunk = 10)
tcga_coad = GDCprepare(query)
```

Para manejar la clase necesitamos el paquete [104, SummarizedExperiment].

```
pacman::p_load(SummarizedExperiment)
```

Eliminamos aquellas variables fenotípicas que tengan más de 7 datos faltantes así como aquellas que no son útiles desde el punto de vista clínico o bien porque son constantes.

```
toremain = which(apply(is.na(colData(tcga_coad)),2,sum)
                 ≤7)
names_toremain = names(colData(tcga_coad))[toremain]
names_toremove = c("barcode","patient","sample",
                   "sample_submitter_id",
                   "sample_id","state",
                   "pathology_report_uuid",
                   "submitter_id","is_ffpe",
                   "tissue_type",
                   "synchronous_malignancy",
```

```
                        "treatments",
                        "last_known_disease_status",
                        "classification_of_tumor",
                        "diagnosis_id",
                        "tumor_grade","alcohol_history",
                        "exposure_id","demographic_id",
                        "bcr_patient_barcode",
                        "project_id")
names_toremain = setdiff(names_toremain,names_toremove)
colData(tcga_coad) = colData(tcga_coad)[,names_toremain]
## Definimos tipos
tofactor = c(1:4,6,8,9,11:22,26:30)
colData(tcga_coad)[,tofactor] =
    lapply(colData(tcga_coad)[,tofactor],as.factor)
colData(tcga_coad)[,-tofactor] =
    lapply(colData(tcga_coad)[,-tofactor],as.numeric)
save(tcga_coad,file=paste0(dirTamiData,"tcga_coad.rda"))
```

```
load(paste0(dirTamiData,"tcga_coad.rda"))
```

Tenemos 19947 genes y 328 muestras.

3.9 Normalización de RNA-seq

En esta sección se proponen distintos métodos de normalización propuestos en la literatura. Es cierto que los datos de RNA-seq tienen menos ruido que los datos de microarrays. En un principio se consideró que no se necesitaban procedimientos de normalización sofisticados [143]. No obstante, en el protocolo de preparación intervienen muchos procedimientos que introducen variabilidad: la extracción del RNA, la transcripción reversa, la amplificación y la fragmentacion.[61]

¿Cuáles pueden ser las covariables relativas a los genes y a las muestras que que pueden influir en los conteos y podamos considerar ruido técnico? [62]

[61] En resumen, que no nos libramos de normalizar.

[62] El término ruido técnico es peligroso. ¿Cuándo es puramente técnico lo que eliminamos o cuando estamos eliminando señal biológica? Sinceramente el que escribe lo desconoce.

[63] Sequencing depth, size library.

Profundidad de secuenciación o tamaño de la librería [63] Estamos muestreando del total de moléculas disponibles. El tamaño de nuestra muestra es el total de lecturas. Este total de lecturas es distinto para las distintas muestras. En algunos procedimientos para análisis de la expresión diferencial este efecto no necesitará corrección previa pues el propio método incorpora en su modelo estocástico la distinta profundidad.

Composición del RNA Mediante la técnica RNA-seq tenemos una media de la abundancia relativa de cada gen en una muestra de RNA. Pero no tenemos una medida de la cantidad total de RNA por célula. Supongamos que tenemos una pequeña cantidad de genes que se expresan mucho en una muestra pero no en otra. En la muestra en donde se expresan mucho consumen una parte fundamental de la librería de lecturas de modo que otros genes que se expresen en esa muestra pero en mucho menor medida apenas aparecerán. De hecho, veremos que se expresan mucho menos de lo que realmente lo hacen porque los otros han consumido una parte importante del total de lecturas. En otras muestras donde no aparecen tan expresados esto no ocurrirá. Si no se ajusta la composición del RNA los otros genes aparecen falsamente infraregulados en la muestra

donde unos pocos se expresan mucho. El método que vamos a utilizar para corregir este efecto es el la media ajustada de M-valores (TMM)[64] entre cada par de muestras.

[64] Trimmed mean of M-values (TMM).

Contenido GC El contenido GC de un gen no cambia entre muestras. Para un análisis de expresión diferencial no debiera influir. Algunos autores han indicado que puede que esto no sea tan cierto. La primera referencia en notar este efecto es [114].

Longitud del gen Genes más largos tenderán a tener más lecturas alineadas. Otra vez, entre muestras no cambia este factor y para un análisis de expresión diferencial no debiera tener influencia.

Sobre la longitud del gen. En [111] se comenta e ilustra el problema. ¿Qué efecto cabe experar cuando tenemos genes con distinta longitud. En RNA-seq lo que tenemos son fragmentos de transcritos. Si el transcrito es más largo el número de lecturas que producen es mayor. Es cierto que si queremos comparar entre muestras este efecto es el mismo. La longitud es la misma en todas las muestras. Sin embargo, al tener más lecturas la potencia del test es mayor para genes más largo y por ello encontraremos más genes con expresión diferencial entre los genes más largos. Un mayor tamaño muestral se asocia a una mayor probabilidad de rechazar la hipótesis nula de igualdad de las medias para una misma diferencia entre medias. De hecho, se propone en el trabajo indicado una simple formalización de este hecho. Denotamos por X la variable aleatoria que nos da el número de lecturas alineadas sobre un gen. Olvidande otros posibles sesgos podemos asumir que el valor medio de X depende del total de transcritos N y de la longitud del gen según la siguiente relación

$$\mu_X = EX = cNL,$$

siendo c un valor constante. En resumen asumimos que la media de lecturas es proporcional al tamaño de la librería y a la longitud del gen. Si se asume que la media y la varianza son iguales[65] entonces

[65] Lo cual es cierto bajo la hipótesis de que la variable X sigue una distribución de Poisson.

$$var(X) = \sigma_X^2 = cNL,$$

Si consideramos la variable X en dos muestras distintas tendremos X_i con $i = 1, 1$ y tamaños de librería N_i con $i = 1, 2$. Si utilizamos para contrastar la igualdad de medias el estadístico

$$\frac{X_1 - X_2}{\sqrt{cN_1L + cN_2L}}.$$

La potencia del test depende del cociente entre la media y la desviación estándar de $D = X_1 - X_2$[66] que viene dado por

[66] O error estándar.

$$\delta = \frac{cN_1L - cN_2L}{\sqrt{cN_1L + cN_2L}} \propto \sqrt{L}.$$

Es decir, δ es proporcional a la raiz cuadrada de la longitud del gen. Puede pensarse que un corrección simple como sería dividir el conteo por la longitud del gen corrige este efecto. Si aplicamos esta corrección no se resuelve el problema ya que entonces la diferencia sería $D' = \frac{X_1}{L} - \frac{X_2}{L}$. Como es bien conocido tenemos que la media y varianza de X_i/N_i son

$$E\frac{X_i}{L} = \frac{EX_i}{L} \text{ y } var\left(\frac{X_i}{L}\right) = \frac{var(X_i)}{L^2}$$

Por tanto el cociente entre la media y la desviación estándar de D' no se modifica y vale

$$\delta = \frac{cN_1L - cN_2L}{\sqrt{cN_1L + cN_2L}} \propto \sqrt{L}.$$

3.9.1 RPKM

[67] Reads per kilobase per million.

[67] El primer método de normalización se propuso en [105]. Supongamos que denotamos el conteo de interés por C. Es el número de lecturas alineadas sobre la característica de interés (exon, gen, etc.). El total de lecturas que podemos alinear es N (o tamaño de la librería). Y denotamos por L la longitud de la característica de interés en bp. Se define el RPKM como el siguiente cociente

$$RPKM = \frac{10^9 C}{NL}. \tag{3.4}$$

Esta cuantificación de la expresión del gen nos permite comparar genes entre sí dentro de una misma muestra o librería.

En el paquete [162, geneLenDataBase] tenemos las longitudes de los distintos transcritos para distintos organismos. En el siguiente código leemos los datos correspondientes a humanos (versión 19).

```
pacman::p_load("geneLenDataBase")
data(hg19.ensGene.LENGTH)
head(hg19.ensGene.LENGTH)
```

```
           Gene Transcript Length
1 ENSG00000118473 ENST00000237247 4024
2 ENSG00000118473 ENST00000371039 4102
3 ENSG00000118473 ENST00000424320 964
4 ENSG00000118473 ENST00000320161 4693
5 ENSG00000118473 ENST00000371036 7294
6 ENSG00000118473 ENST00000407289 7901
```

3.9.2 Método TMM: Media ajustada de M-valores

Es, quizás, el método más popular de normalización propuesto en [123]. Comentan los autores en [123, pág. 2] el siguiente ejemplo hipotético. Tenemos dos experimentos de secuenciación de RNA o muestras. Llamamos a estas dos muestras A y B. Supongamos dos conjuntos de genes disjuntos y con el mismo número de elementos: S_1 y S_2. El primer conjunto se expresa igualmente en A y B, es decir, tenemos el mismo número de transcritos en ambas muestras. Sin embargo, S_2 solamente se expresa en la muestra A y no en la muestra B. Además asumimos que no se expresa ningún otro gen. Si la profundidad de secuenciación es la misma en las dos muestras entonces el número de transcritos que observaremos de un gen de S_1 será la mitad en A que en B aunque sabemos que se expresa exactamente igual. En resumen, en la población (total de transcritos) el número total de transcritos de este gen es el mismo en ambas muestras. La probabilidad de observar uno de ellos depende de su frecuencia y del número de transcritos observados. Por ello, depende de que otros genes se expresen más o menos. En resumen, de la composición del RNA.

Sea X_{ij} (x_{ij}) el conteo aleatorio (observado) del gen i en la muestra j. Denotamos L_i la longitud del gen i y m_j es el total de lecturas de la

librería j (o tamaño de la librería j). En [123] se asume que la media de X_{ij} verifica

$$E\left[X_{ij}\right] = \frac{\mu_{ij}L_i}{c_j}m_j,$$

siendo $c_j = \sum_{i=1}^{N} \mu_{ij}L_i$. Notemos que c_j nos está representando el total de RNA en la muestra. No se asume ninguna distribución de probabilidad específica para X_{ij}. La producción total de RNA, c_j, no es conocida. Sin embargo, sí es fácil estimar el cociente de estos valores para dos muestras, es decir, estimar el cociente $f_j = c_j/c_{j'}$.

Elegimos una muestra como muestra de referencia. Por ejemplo, la muestra r denota a partir de ahora la muestra tomada (arbitrariamente) como de referencia.

Nos fijamos en la muestra j y vamos a determinar la constante por la que multiplicaremos los conteos originales. En lo que sigue tanto j como r son fijos y las cantidades definidas dependen de i que denota el gen. Se define

$$M_{ij}^{(r)} = \log_2 \frac{x_{ij}/m_j}{x_{ir}/m_r} = \log_2(x_{ij}/m_j) - \log_2(x_{ir}/m_r),$$

y

$$A_{ij}^{(r)} = \frac{1}{2}\Big(\log_2(x_{ij}/m_j) + \log_2(x_{ir}/m_r)\Big)$$

Se eliminan los valores extremos tanto de los $M_{ij}^{(r)}$ como de los $A_{ij}^{(r)}$.[68] En concreto eliminamos un porcentaje de los M_i más pequeños y el mismo porcentaje de los más grandes. Lo mismo hacemos para los valores A_i. También eliminamos aquellos índices i tales que $x_{ij} = 0$ o bien $x_{ir} = 0$. El conjunto de índices i restante lo denotamos por G^*. Finalmente, el factor de normalización sería

$$\log_2(TMM_j^{(r)}) = \frac{\sum_{i\in G^*} w_{ij}^{(r)} M_{ij}^{(r)}}{\sum_{i\in G^*} w_{ij}^{(r)}}$$

con

$$w_{ij}^{(r)} = \frac{m_j - x_{ij}}{m_j x_{ij}} + \frac{m_r - x_{ir}}{m_r x_{ir}}.$$

La muestra de referencia r es fija y lo que acabamos de calcular es el factor por el que multiplicamos los conteos originales de la muestra j. Este factor viene dado por $TMM_j^{(r)}$.[69]

Tanto este método como los que siguen están implementados en [38, edgeR].[70]

Utilizamos los datos `tamidata::PRJNA297664`.

```
data(PRJNA297664,package="tamidata")
```

Accedemos a la matriz de expresión con los conteos.

```
pacman::p_load(SummarizedExperiment)
counts0 = assay(PRJNA297664)
```

```
pacman::p_load(edgeR)
deg.n = DGEList(counts = counts0,
                group = rep(1,ncol(counts0)))
deg.n = calcNormFactors(deg.n, method = "TMM")
deg.n = estimateCommonDisp(deg.n)
counts1 = deg.n$pseudo.counts
```

[68] Recordemos que varía solamente i y mantenemos fijos j y r.

[69] Obviamente tenemos que $TMM_r^{(r)} = 1$ y esta muestra de referencia no se normaliza.

[70] El código relativo a métodos de normalización procede de [66, capítulo 4].

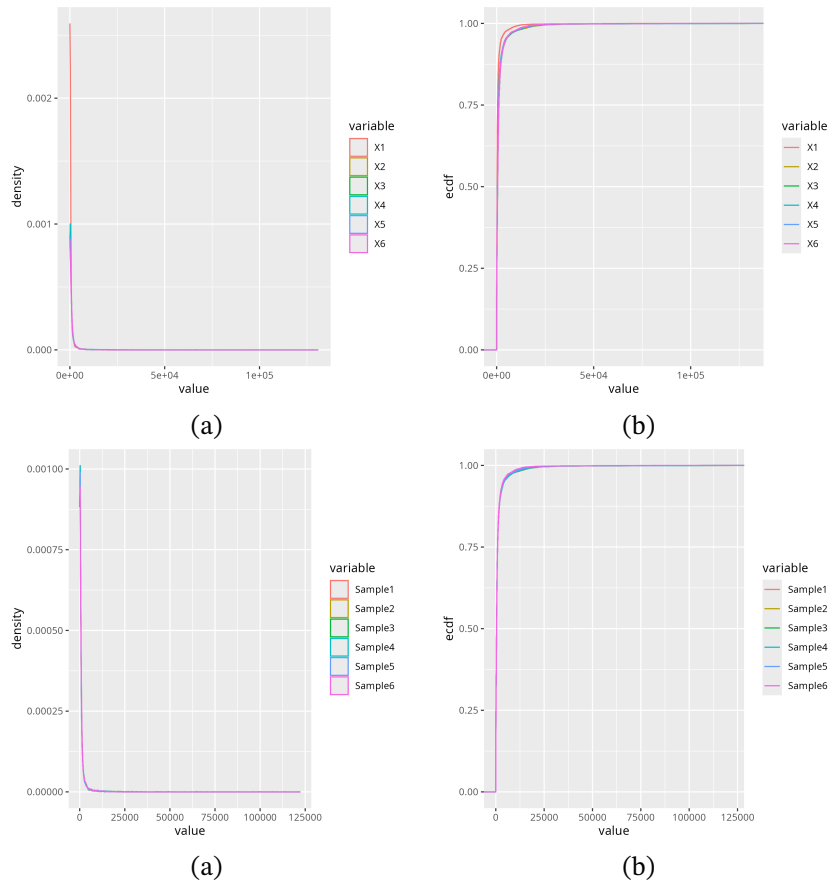


Figura 3.1: (a) Estimaciones de la función de densidad de los conteos para las distintas muestras de PRJNA297664 con los conteos originales. (b) Lo mismo que (a) con la función de distribución. En (c) y (d) tenemos los dibujos análogos utilizando los conteos después de aplicarles una normalización TMM.

En la figura 3.1(a) y (b) tenemos la función de densidad y de distribución estimadas de los conteos antes de la normalización para las distintas muestras. Los dibujos se generan con el siguiente código.

```
pacman::p_load("ggplot2")
counts0 = data.frame(counts0)
df0 = reshape2::melt(counts0)
p = ggplot(df0, aes(x=value, colour = variable)) +
      stat_ecdf()
ggsave(paste0(dirTamiFigures,"PRJNA297664_counts0_ecdf.png"),p)
p = ggplot(df0, aes(x=value, colour = variable)) +
      geom_density()
ggsave(paste0(dirTamiFigures,"PRJNA297664_counts0_density.png"),p)
counts1 = data.frame(counts1)
df0 = reshape2::melt(counts1)
p = ggplot(df0, aes(x=value, colour = variable)) +
      stat_ecdf()
ggsave(paste0(dirTamiFigures,"PRJNA297664_counts1_ecdf.png"),p)
p = ggplot(df0, aes(x=value, colour = variable)) + geom_density()
ggsave(paste0(dirTamiFigures,"PRJNA297664_counts1_density.png"),p)
```

3.9.3 Mediana de los cocientes

Se propuso en [10]. Se consideran los factores de normalización

$$s_j = mediana_{\{i:\tilde{m}_i \neq 0\}} \frac{x_{ij}}{\tilde{m}_i} \tag{3.5}$$

siendo $\tilde{m}_i = \left(\prod_{j=1}^{n} x_{ij} \right)^{\frac{1}{n}}$. La idea es normalizar para cada gen dividiendo el conteo por la correspondiente media geométrica de los distintos conteos para un mismo gen. Una vez que tenemos los datos normalizados intra gen buscamos una constante de normalización por muestra como la correspondiente mediana en la muestra. Estas constantes son utilizadas no para normalizar directamente el conteo sino para incorporarlo como factor que multiplica a la media del conteo en los métodos `DESeq` y `DESeq2`.

Utilizando los mismos datos `tamidata::PRJNA297664` el método anterior lo podemos aplicar con el siguiente código.

```
deg.n = DGEList(counts = counts0,
                group = rep(1,ncol(counts0)))
deg.n = calcNormFactors(deg.n, method = "RLE")
deg.n = estimateCommonDisp(deg.n)
counts2 = deg.n$pseudo.counts
```

3.9.4 Homogeneizando los percentiles

Este método forma parte del método propuesto en [125] y comentado en § 10.3. Indicamos aquí cómo aplicarlo a los datos `tamidata::`
↪ `PRJNA297664`. Como vemos en el código que sigue no se especifica método pues es el que aplica por defecto.

```
deg.n = DGEList(counts = counts0, group = rep(1,ncol(counts0)))
deg.n = estimateCommonDisp(deg.n)
counts3 = deg.n$pseudo.counts
```

Capítulo 4

Metilación

La metilación del DNA es una modificación de una base citosina para formar una metilcitosina. Esta modificación ocurre casi exclusivamente cuando la citosina está seguida por una guanina en la dirección 5' a 3'. Esto se denota como CpG. Básicamente pretendemos evaluar si una citosina está metilada o no. Una alta proporción de las regiones en donde se inicia la transcripción de un gen tienen elevados contenidos de **CpG** y que son conocidos como **islas CpG**. Si en una de estas islas CpG están localizadas en un promotor (zona de inicio de transcripción) y además hay una alta proporción de CpG metilados entonces la transcripción se dificulta. Si corresponde con genes que intervienen en procesos de regulación ligados a enfermedades como el cáncer esto produce un incremento en la probabilidad de desarrollar la enfermedad. Es interesante evaluar la metilación del genoma y, particularmente, en las regiones promotoras por su relación con el desarrollo de enfermedades.

Dos tipos de técnicas ómicas permiten esta evaluación: una basada en arrays y la otra basada en secuenciación. Distintas plataformas (arrays) han sido desarrolladas por Illumina.

Si hablamos de **arrays** hemos de tener en cuenta que se evalúan una enorme cantidad de CpGs. Realmente a nivel molecular es un fenómenos binario ya que una citosina está metilada o no pero estas tecnologías utilizan millones de células y por tanto la medida que nos proporcionan está relacionada con la proporción de células que están metiladas en la muestra biológica que estamos analizando. Nuestra información consiste en la posición que ocupa la base (cromosoma y posición dentro de cromosoma) y un valor que indica un grado de metilación (no exactamente una proporción).

Si se utilizan técnicas de secuenciación entonces el valor que se nos proporciona es distinto. Básicamente tendremos, en un locus donde tenemos una citosina, un número de lecturas alineadas que lo cubren. En estas lecturas cortas algunas tendrán la metilación y otras no. Lo que tenemos pues es una fracción de lecturas metiladas que cubren ese locus. Es interesante darse cuenta que en realidad podemos ver el total de lecturas alineadas que cubren el locus como un número de pruebas donde habrá éxitos (lecturas con metilación en el locus de interés) y fracasos (lecturas no metiladas en el locus de interés).

Tenemos estos datos de metilación a lo largo de miles y millones de loci y hemos de intentar asociar estos datos con variables fenotípicas (cáncer o no por ejemplo). Sustituímos un DNA microarray por otro tipo de información pero el problema no es distinto. Si utilizamos arrays te-

nemos un dato continuo (modelos lineales normales o similares) y con secuenciación estamos más próximos, estadísticamente hablando, a modelo lineal generalizado.

4.1 Datos y preprocesado

El código de esta sección es debido a Angela Riffo-Campos. Cargamos algunos paquetes necesarios.

```
pacman::p_load(minfi,SummarizedExperiment,
    ↪ IlluminaHumanMethylationEPICmanifest,
    ↪ IlluminaHumanMethylationEPICanno.ilm10b4.hg19)
##pacman::p_load(minfi,IlluminaHumanMethylationEPICmanifest,
## RColorBrewer)
##pacman::p_load(missMethyl,minfiData,DMRcate)
```

Bajamos los datos de GEO con número de acceso GSE149282.

```
setwd(dirTamiData)
gcel = GEOquery::getGEOSuppFiles("GSE149282")
```

El formato de los datos is IDAT. Utilizando la información que encontramos en GSE149282 hemos creado el fichero de texto `idats.csv` con los siguientes datos.

```
path = paste0(dirTamiData,"GSE149282_RAW/")
targetsBasename = read.csv(paste0(path,"idats.csv"),
                           head=TRUE)
head(targetsBasename)
```

```
                          File_name
1 GSM4495491_200811050117_R01C01_Grn.idat
2 GSM4495491_200811050117_R01C01_Red.idat
3 GSM4495492_200811050117_R02C01_Grn.idat
4 GSM4495492_200811050117_R02C01_Red.idat
5 GSM4495493_200811050117_R03C01_Grn.idat
6 GSM4495493_200811050117_R03C01_Red.idat
```

Notemos que `targetsBasename` es un `data.frame`.

```
class(targetsBasename)
```

```
[1] "data.frame"
```

Necesitamos que `targetsBasename` sea un `DataFrame` en lo que sigue.

```
targetsBasename = DataFrame(targetsBasename)
```

Construimos un vector con los nombres de los ficheros que contienen los datos.

```
targets = paste0(path,as.character(targetsBasename$File_name))
```

Leemos los datos.

```
rgset = minfi::read.metharray(targets, verbose = TRUE,
                              force=TRUE)
```

Asignames a `rgset` los metadatos que hemos leído previamente.

```
colData(rgset) = targetsBasename
```

```
Error in .requirePackage(package): unable to load required package '
    ↪ minfi'
```

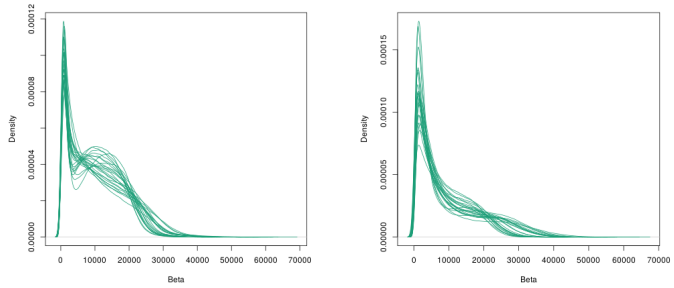


Figura 4.1: Estimadores de densidad

Podemos ver las intensidades en la banda del rojo,

```
head(minfi::getRed(rgset),n=1)
```

```
Error in (function (cond) : error al evaluar el argumento 'x' al
    ↪ seleccionar un método para la función 'head': no hay paquete
    ↪ llamado 'minfi'
```

y en la banda del verde.

```
head(minfi::getGreen(rgset),n=1)
```

```
Error in (function (cond) : error al evaluar el argumento 'x' al
    ↪ seleccionar un método para la función 'head': no hay paquete
    ↪ llamado 'minfi'
```

Preprocesamos los datos utilizando el paquete [**R-IlluminaHumanMethylationEPICmanifest**].

```
pacman::p_load(IlluminaHumanMethylationEPICmanifest)
mset = preprocessIllumina(rgset)
```

Hacemos la correspondencia con el genoma que necesita el paquete [**R-IlluminaHumanMethylationEPICa**

```
GSE149282 = mapToGenome(mset)
save(GSE149282,file=paste0(dirTamiData,"GSE149282.rda"))
```

4.1.1 Explorando los datos

Tenemos las señales metiladas y no metiladas con

```
head(getMeth(GSE149282),n=1)
head(getUnmeth(GSE149282),n=1)
```

Podemos visualizar un estimador de densidad de ambos conjuntos de valores.

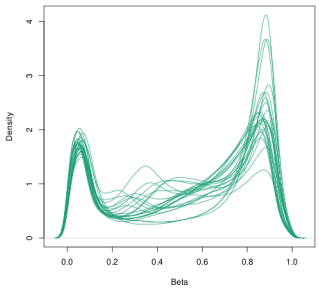
```
minfi::densityPlot(getMeth(GSE149282))
minfi::densityPlot(getUnmeth(GSE149282))
```

En la figura 4.1 tenemos ambos estimadores.

Si denotamos por M y por U los valores observamos para las señales metiladas y no metiladas entonces el valor β se define como

$$\beta = \frac{M}{U} \tag{4.1}$$

En figura 4.2 mostramos la densidad estimada de los valores β.




```
minfi::densityPlot(getBeta(GSE149282,type="Illumina"))
```

Los valores $\mathbb{M}$ se definen como

$$\mathbb{M} = \log_2 \frac{M}{U}. \tag{4.2}$$

La densidad estimada de $\mathbb{M}$ la tenemos en figura 4.3.

```
minfi::densityPlot(getM(GSE149282))
```

Normalizamos los datos.

```
GSE149282n = preprocessQuantile(GSE149282)
```

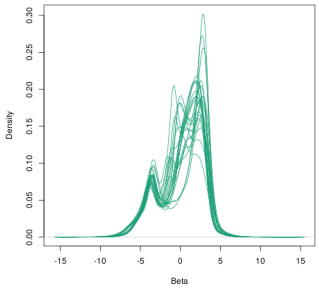


Figura 4.3: Estimadores de densidad

4.1.2 Filtrado

Eliminamos las sondas basándonos en el p-valor de detección.

```
detP = detectionP(rgset)
```

Nos aseguramos que las sondas están en la misma posición en `GSE149282n` y en `detP`.

```
detP = detP[match(featureNames(GSE149282n),rownames(detP)),]
```

Eliminamos las sondas con p-valores altos.

```
keep = rowSums(detP < 0.01) == ncol(GSE149282n)
table(keep)
GSE149282nFlt = GSE149282n[keep,]
```

Eliminamos sondas con SNPs en la región CpG.

```
GSE149282nFlt = dropLociWithSnps(GSE149282nFlt)
```

Podemos obtener los valores de $\mathbb{M}$ y β.

```
mVals = getM(GSE149282nFlt)
```

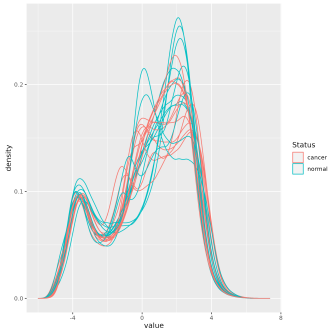
```
Error in getM(GSE149282nFlt): no se pudo encontrar la función "getM"
```

```
bVals = getBeta(GSE149282nFlt)
```

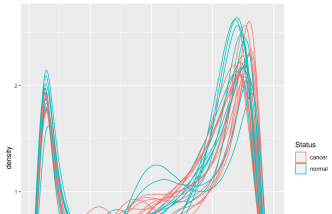
```
Error in getBeta(GSE149282nFlt): no se pudo encontrar la función "
    ↪ getBeta"
```

Las densidades estimadas aparecen en la figura 4.4.

```
pacman::p_load(ggplot2)
df = reshape2::melt(mVals)
df = cbind(df,
           Status=rep(colData(GSE149282nFlt)[,"Status"],
                      each=nrow(mVals)))
head(df)
names(df)=c("cg","sample","value","Status")
df$Status = factor(df$Status)
p = ggplot(df,aes(x=value,group=sample,color=Status))+geom_density()
ggsave(paste0(dirTamiFigures,"GSE149282nFltM.png"),p)
```




```
df = reshape2::melt(bVals)
df = cbind(df,Status=rep(colData(GSE149282nFlt)[,"Status"],each=nrow(mVals)))
head(df)
names(df)=c("cg","sample","value","Status")
df$Status = factor(df$Status)
p = ggplot(df,aes(x=value,group=sample,color=Status))+
    geom_density()
ggsave(paste0(dirTamiFigures,"GSE149282nFltB.png"),p)
```



Parte II

Análisis marginal

Capítulo 5

Expresión diferencial marginal

5.1 Introducción

Para cada uno de los genes (características ómicas) tenemos un valor numérico (expresión) que nos indica abundancia de la característica en la que estamos interesados. Tenemos este valor observado para distintas muestras. De cada una de las muestras tenemos a su vez variables que las describen (tiempos, tamaños celulares, temperatura, etc) a las llamaremos en lo que sigue **variables fenotípicas, covariables o metadatos**. Estas variables pueden ser categóricas, numéricas o tiempos de supervivencia censurados. Si la covariable tiene dos categorías entonces nos define dos grupos (por ejemplo, control y tratamiento). El caso en que queremos comparar dos grupos es el más frecuente y por ello siempre le daremos una mayor atención.

El problema que se conoce como **expresión diferencial** se puede traducir a saber si hay algún tipo de asociación entre las expresiones observadas y los valores de la variable fenotípica. Si la variable fenotípica es binaria nos define dos categorías y la pregunta de la asociación covariable-expresión se puede formular como: ¿Hay diferencias entre la expresión de genes entre dos grupos considerados? Dicho de otro modo, la expresión es distinta bajo los tratamientos que estamos considerando.

En un primer momento vamos a adoptar la aproximación gen-a-gen (o marginalmente para cada característica). Buscamos genes que se expresan diferencialmente sin atender a las interacciones que puedan existir entre los distintos genes, que las hay y las consideraremos más adelante.

5.2 Algo de notación

En lo que sigue denotaremos la matriz con los datos de expresión de los distintos genes como $\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$ donde el valor y_{ij} nos da el nivel de expresión **observado** del gen i-ésimo en la muestra j-ésima. Asociada a la muestra j tenemos las variables fenotípicas $\boldsymbol{x}_j \in \mathbb{R}^p$. Por ejemplo, si estamos comparando los niveles de expresión en dos grupos (un grupo control y un grupo de enfermos por ejemplo) entonces x_j tomará el valor 1 si está en el primer grupo (control) y valor 2 si está

en el segundo grupo (valor 2). Como es de uso habitual denotaremos

$$y_{\cdot j} = \sum_{i=1}^{N} y_{ij}; \;\; y_{i\cdot} = \sum_{j=1}^{n} y_{ij},$$

$$\bar{y}_{\cdot j} = \sum_{i=1}^{N} \frac{y_{ij}}{N}; \;\; \bar{y}_{i\cdot} = \sum_{j=1}^{n} \frac{y_{ij}}{n}.$$

En la notación previa hemos indicado los valores de expresión o la variable fenotípica utilizando letras en minúsculas como es habitual en Probabilidad. Sin embargo cuando consideremos los valores que observaremos **antes** de realizar la experimentación tendremos los valores aleatorios que denotaremos con la misma letra pero en mayúscula. Se tiene que $\boldsymbol{Y} = [Y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$ es la matriz de expresión aleatoria antes de observar los datos y $\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$ es la misma matriz después de observarlos.

5.3 Selección no específica o filtrado independiente

El material de esta sección se ha obtenido de [67, capítulo 5] y [59]. Como material complementario es muy interesante [23].

La mayor parte de los genes no se expresan de un modo diferenciado entre condiciones. Y hay muchos.[1] Más genes evaluados *simultáneamente* suponen una dificultad mayor en el tratamiento estadístico posterior. Por ello si podemos de un modo simple *quitar* genes que no parecen tener actividad pues mucho mejor.[2]

Es costumbre realizar una preselección, un filtrado que no utilice información de la covariable y. Utilizamos el perfil de expresión del gen pero no información sobre las muestras, la posible clasificación de muestras en distintos grupos. A este filtrado se le da en la literatura la denominación de *filtrado independiente* o bien **selección no específica**.

Veamos un posible método. Consideramos el gen i-ésimo y tomamos una medida de localizacion como la mediana. Denotamos este valor por u_i. Determinamos un cierto percentil de estos valores u_i. Por ejemplo, si denotamos por p_1 el orden del percentil entonces $q_{p_1}(u)$ sería el correspondiente percentil de orden p_1 de los valores u_i. Mantendremos los genes tales que su valor u_i sea mayor que $q_p(u)$. Ahora consideramos una medida de dispersión como el coeficiente intercuartílico. Del mismo modo, v_i sería el coeficiente intercuartílico del i-ésimo gen y el correspondiente percentil sería $q_{p_2}(v)$ donde el orden p_2 no tiene porqué coincidir con p_1. Nos quedamos con los genes en el siguiente conjunto

$$\{i : i = 1, \ldots, N; u_i \geq q_{p_1}(u); v_i \geq q_{p_2}(v)\},$$

es decir, aquellos que tienen un nivel de expresión alto y una variabilidad alta. En otras palabras, genes que se expresan y que lo hacen de un modo variable en las distintas muestras.

[1] A veces uno piensa que demasiados.

[2] Aunque esto puede tener muchos riesgos como descartar genes que actúan conjuntamente con otros y marginalmente no tengan una actividad apreciable. Es pues algo peligroso eliminar genes sin tener muy claro como lo hacemos.

En el método que acabamos de explicar podemos sustituir la mediana como localización y el rango intercuartílico como dispersión por la media y la desviación estándar.

Otro procedimiento que se ha propuesto y utilizado con frecuencia es que un gen se exprese con claridad en algunas muestras, lo que podemos llamar el método k-sobre-A. La idea es simple, un gen lo conservamos si en al menos k muestras tiene un nivel de expresión por encima de un valor mínimo A.

En lo que sigue vemos cómo utilizar estas ideas con R/Bioconductor. En § 5.3.1 se propone una opción muy sencilla para preseleccionar genes utilizando simplemente la función *apply*. En § 5.3.2 utilizamos el paquete [59, genefilter]. Finalmente en § 5.3.3 utilizamos la función `genefilter` ↪ `::nsFilter()`.

Vamos a abordar el problema con los datos `ALL::ALL`. En primer lugar aplicamos la selección de muestras de § 2.7.2 que podemos ver en § A.4.

5.3.1 Utilizando apply

Vamos a implementar un par de procedimientos de selección independiente con la función `apply`.

Un procedimiento muy básico puede ser calcular para el rango intercuartílico del perfil de expresión de cada gen.

```
load(paste0(dirTamiData,"bcrneg.rda"))
bcrneg.iqr = apply(exprs(bcrneg),1,IQR)
```

Y luego podemos determinar el cuantil (por ejemplo, el percentil 0.5 o mediana) de los rangos intercuartílicos. Si estás por encima de este percentil conservamos el gen. De lo contrario, no lo consideramos en lo que sigue.

```
sel.iqr = (bcrneg.iqr > quantile(bcrneg.iqr,0.5))
```

Una idea como la que acabamos de aplicar para seleccionar genes tiene el inconveniente de que si un grupo tiene pocos datos entonces aunque hubiera una expresión diferencial la variabilidad total no tiene porqué ser muy grande.

Ahora filtramos atendiendo a la mediana del perfil de expresión.

```
bcrneg.median = apply(exprs(bcrneg),1,median)
sel.median = (bcrneg.median > quantile(bcrneg.median,0.5))
```

Y nos quedamos con los genes que verifican ambas condiciones. Nuestro ExpressionSet original es **bcrneg**, ¿cómo nos quedamos con estos genes y todas las muestras? Con el siguiente código.

```
bcrneg1 = bcrneg[sel.iqr & sel.median,]
```

MEDIA Y DESVIACIÓN ESTÁNDAR

Sustituimos mediana y rango intercuartílico por media y desviación estándar respectivamente.

Es claro que podríamos sustituir la desviación estándar por alguna otra medida de variabilidad, por ejemplo, el rango intercuartílico. El método anterior se podría implementar con el siguiente código.

```
bcrneg.sd = apply(exprs(bcrneg),1,sd)
sel.sd = (bcrneg.sd > quantile(bcrneg.sd,0.5))
bcrneg.mean = apply(exprs(bcrneg),1,mean)
```

```
sel.mean = (bcrneg.mean > quantile(bcrneg.mean,0.5))
bcrneg2 = bcrneg[sel.sd & sel.mean,]
```

k sobre A

Otra opción natural sería fijar un nivel de actividad mínima para un gen y quedarnos con aquellos que superen este nivel mínimo de actividad. Consideremos el siguiente criterio: si el nivel de expresión mínimo es c y tenemos n muestras podemos pedir que un gen determinado se considere activo si en al menos k muestras del total de n su nivel de expresión supere este nivel mínimo de actividad.

¿Cómo hacerlo con R? Empezamos fijando el nivel mínimo c. ¿Y cómo? Podemos ver los percentiles de todas las expresiones (todos los genes y todas las muestras).

```
quantile(exprs(bcrneg))
```

```
     0% 25% 50% 75% 100%
1.984919 4.117796 5.468801 6.832439 14.044896
```

Nos quedamos con la mediana para c.

```
c = 5.468801
```

Determinamos qué niveles de expresión lo superan.

```
overc = exprs(bcrneg) > c
```

Podemos ver los resultados para la fila 433.

```
overc[433,]
```

Determinamos el número de muestras por fila que superan el valor c. Notemos que aplicamos una suma. Cuando sumamos un vector lógico (con valores TRUE y FALSE) entonces el valor TRUE se interpreta como el número 1 y el valor FALSE se interpreta como 0.

```
count.c = apply(overc,1,sum)
```

Por ejemplo, las primeras filas son

```
head(count.c)
```

```
 1000_at 1001_at 1002_f_at 1003_s_at 1004_at
    79 3 0 77 76
 1005_at
    79
```

Y finalmente determinamos si estamos por encima de 5 muestras.

```
sel.c = count.c ≥5
```

Y veamos cuántos genes conservamos.

```
table(sel.c)
```

```
sel.c
FALSE TRUE
 4736 7889
```

Una vez elegido un procedimiento de los comentados eliminamos aquellos genes que no pasan el criterio de selección. Y con los genes que nos quedan seguiríamos estudiando si hay o no expresión diferencial. Observemos que no hemos utilizado para nada los grupos que pretendemos comparar.

5.3.2 Utilizando genefilter

En la anterior sección utilizamos funciones básicas de R para realizar la selección no específica. Podemos utilizar un paquete diseñado para esto. El paquete genefilter [59] nos permite implementar distintos procedimientos de filtrado independiente de un modo sencillo.

Seguimos utilizando los datos del ExpressionSet **bcrneg**. Supongamos un procedimiento de filtrado no específico tan simple como el siguiente: mantendremos aquellos genes tal que su nivel de expresión es al menos de 200 en al menos 5 de las muestras. Para hacer esto con el paquete [59, genefilter] hemos de realizar tres pasos:

1. Crear la función que implementa cada criterio de filtrado.
2. Combinar todos los criterios de filtrado en una única función.
3. Aplicar la función de filtrado final a la matriz de expresión.

El siguiente código implementa todo el procedimiento.

```
library(Biobase)
library(genefilter)
f1 = kOverA(5, 5.468801)
ffun = filterfun(f1)
wh1 = genefilter(exprs(bcrneg), ffun)
```

Finalmente podemos ver cuántos genes permanecen en nuestro estudio. Son aquellos que tiene el valor de wh1 igual a TRUE.

```
table(wh1)
```

```
wh1
FALSE TRUE
 4736 7889
```

Vamos a incorporar los otros dos criterios comentados previamente. Necesitamos una función que nos diga si la desviación estándar supera el valor c.

Primero vemos el modelo que nos ofrece la función `genefilter::kOverA()`.

```
kOverA = function (k, A = 100, na.rm = TRUE)
{
    function(x) {
        if (na.rm)
            x = x[!is.na(x)]
        sum(x > A) ≥k
    }
}
```

Y nos definimos algo similar utilizando la desviación estándar.

```
sdOverc = function (c, na.rm = TRUE)
{
    function(x) {
        if (na.rm)
            x = x[!is.na(x)]
        sd(x) ≥c
    }
  }
```

Y ahora otra función que evalúe si el rango intercuartílico supera un cierto valor.

```
iqrOverc = function (c, na.rm = TRUE)
{
    function(x) {
        if (na.rm)
            x = x[!is.na(x)]
        IQR(x) ≥c
    }
  }
```

```
f1 = kOverA(5, 200)
f2 = sdOverc(150)
f3 = iqrOverc(72)
ffun = filterfun(f1,f2,f3)
wh123 = genefilter(exprs(bcrneg), ffun)
```

¿Qué genes verifican este criterio de preselección?

```
table(wh123)
```

```
wh123
FALSE
12625
```

5.3.3 Utilizando `nsFilter()`

Supongamos que queremos aplicar la siguiente selección nos vamos a quedar con aquellas sondas tales que el rango intercuartílico (para todas las muestras) es mayor que la mediana de los rangos intercuartílicos. Luego vamos a necesitar conocer la anotación de los genes. Si esto es así podemos filtrar aquellos genes de los cuales desconocemos su anotación. Para ello necesitamos conocer primero el paquete de anotación que utilizan nuestros datos.[3]

```
annotation(ALL)
```

```
[1] "hgu95av2"
```

Por tanto hemos de cargar este paquete.

```
library(hgu95av2.db)
```

Realizamos el filtrado correspondiente.[4]

```
bcrneg.filt1 = nsFilter(bcrneg,var.func=IQR,var.cutoff=0.5,
    require.GOBP=TRUE)
```

La función que utilizamos para medir variabilidad podemos ir modificándola. Por ejemplo, podemos sustituir `var.func()` por la varianza o por la desviación estándar. Así podríamos realizar

```
bcrneg.filt2 = nsFilter(bcrneg,var.func=sd,var.cutoff=0.5,
    require.GOBP=TRUE)
```

¿Cómo podemos combinar ambas selección? Utilizando la función `genefilter` ↪ `::nsFilter()` tenemos un `Biobase::ExpressionSet`. Una manera simple puede ser la siguiente.

[3] En http://www.bioconductor.org/packages/release/data/annotation/ tenemos un listado de paquetes de anotación.

[4] Observemos que ahora no pasamos la matriz de expresión sino el ExpressionSet mismo.

```
sel = intersect(featureNames(bcrneg.filt1),
    featureNames(bcrneg.filt2))
```

Y nuestro `Biobase::ExpressionSet` filtrado sería

```
bcrneg1 = bcrneg[sel,]
```

Y ya podemos seguir trabajando con los genes que hemos filtrado.

Los peligros de la selección no específica

Hemos presentado como habitual la combinación de una selección no específica o independiente de genes con un análisis posterior marginal de los genes que no han sido seleccionados utilizando procedimientos de corrección por comparaciones múltiples. Este procedimiento, aún habitual en la literatura, tiene sus riesgos. Un trabajo de gran interés a consultar sobre este problema es [23].

5.4 Fold-change

[5] Queremos comparar dos grupos. Denotamos los valores de expresión originales con x_{ij} (respectivamente y_{ij}) para la i-ésima característica en la j-ésima muestra del primer grupo (respectivamente del segundo grupo). A los logaritmos (en base 2 o log2) de los valores originales los denotamos por $u_{ij} = \log_2(x_{ij})$ y $v_{ij} = \log_2(y_{ij})$. Para cada gen, tendremos una expresión media para la i-ésima característica en cada uno de los dos grupos $\bar{x}_{i\cdot} = \sum_{j=1}^{n_1} \frac{x_{ij}}{n_1}$ para la media en el primer grupo. Similarmente definimos $\bar{y}_{i\cdot}$, $\bar{u}_{i\cdot}$, $\bar{v}_{i\cdot}$. ¿Qué se entiende por **fold-change**? Dos son las interpretaciones de este valor. La primera ([140]) lo define como

$$FC_i^{(1)} = \frac{\bar{x}_{i\cdot}}{\bar{y}_{i\cdot}}. \tag{5.1}$$

Lo definimos como el cociente de las medias de las expresiones en la escala original. La segunda definición (que no es equivalente a la primera) es

$$FC_i^{(2)} = \bar{u}_{i\cdot} - \bar{v}_{i\cdot}, \tag{5.2}$$

tenemos la diferencia de las medias de los logaritmos en base 2 de las expresiones. Un procedimiento[71] que se ha utilizado frecuentemente en la literatura de microarrays consiste en tomar el log2 del fold-change en la definición 5.1, el valor

[71] Que no es estadístico y que por tanto no vamos a considerar en este manual.

$$\log_2 FC_i^{(1)} = \log_2 \left(\frac{\bar{x}_{i\cdot}}{\bar{y}_{i\cdot}} \right).$$

Si el módulo del cociente anterior es mayor que una constante positiva c ($c > 0$) entonces diríamos que el gen i se sobre expresa en el grupo 1 en relación con el grupo 2 ya que

$$\log_2 \left(\frac{\bar{x}_{i\cdot}}{\bar{y}_{i\cdot}} \right) \geq c.$$

[5] Lo que comentamos en esta sección me ha costado de entender una enfermedad. Lo primero el porqué de utilizarlo y luego que hay una indefinición en el término en la literatura. Vamos a intentar aclarar qué significa antes de que se me olvide. Porque mucho interés en usarlo no tengo.

y se sigue que

$$\frac{\bar{x}_{i\cdot}}{\bar{y}_{i\cdot}} \geq 2^c.$$

Si $\log_2\left(\frac{\bar{x}_1}{\bar{x}_2}\right) < -c$ entonces tendremos que

$$\frac{\bar{x}_{i\cdot}}{\bar{y}_{i\cdot}} \leq -2^c,$$

o bien que

$$\frac{\bar{y}_{i\cdot}}{\bar{x}_{i\cdot}} \geq 2^c.$$

En este segundo caso el gen i se sobre expresa en el grupo 2 en relación con el primer grupo. El término que se utiliza es sobre regulación en el primer caso y de infra regulación en el segundo.[72] Se utiliza con mucha frecuencia. Es sencillo, entendible y fácil de usar. Pero **no** es una buena opción. Lo fundamental, *no se tiene en cuenta la variabilidad de las medias* que estamos comparando. Una opción más correcta la vemos posteriormente en este tema y es la utilización de un test de la t para comparar las medias de las dos poblaciones que estamos comparando. O versiones modificadas del t-test clásico como la propuestas en [140] o en [129] en donde esencialmente se modifica la estimación del error estándar.

[72] Up (down) regulation.

Por último una interpretación en mi opinión inadecuada de la expresión log fold-change es como nombre de los coeficientes en los ajustes lineal o lineal generalizado en procedimientos implementados en algunos paquetes. Por ejemplo, en el método limma [129] implementado en [128]. Asumen que trabajan en escala logarítmica y eso hace que bajo alguna circunstancia se pueda asimilar a alguna de las definiciones previas. No es adecuado en absoluto.

5.5 Expresión diferencial marginal

Trabajamos con los datos `tamidata::gse21942`.

```
library(Biobase)
data(gse21942,package="tamidata")
```

Pretendemos comparar la expresión observada en las distintas sondas para enfermos (esclerosis múltiple) y para sanos. La variable fenotípica que nos indica si la muestra corresponde a enfermo o sano es

```
x0 = pData(gse21942)[,"FactorValue..DISEASE.STATE."]
table(x0) ## Frecuencias de cada categoría
```

```
x0
         healthy multiple sclerosis
             15 12
```

Elegimos (arbitrariamente) el gen que nos aparece en la fila **1345** de la matriz de expresión. ¿Qué información de anotación tenemos?

```
probeRow = 1345
fData(gse21942)[probeRow,]
```

```
        PROBEID ENTREZID ENSEMBL
1938 1554564_a_at 121665 ENSG00000157837
```

Guardamos los niveles de expresión.

```
y0 = exprs(gse21942)[probeRow,]
```

En la figura 5.1(a) mostramos los datos para las distintas muestras (con distintos colores y caracteres distintos).

```
pacman::p_load(ggplot2)
df=data.frame(id = 1:ncol(gse21942),expression = y0,type = x0)
p = ggplot(df,aes(x=type,y=expression,color=type)) +
    geom_jitter(position=position_jitter(0.2))
ggsave(paste0(dirTamiFigures,"DDR1THRA_a.png"),p)
```

Sabemos que hay dos grupos de pacientes. Podemos representar dos diagramas de cajas que nos den una comparación sencilla de los niveles de expresión para los dos grupos. Lo tenemos en figura 5.1(b).

```
df=data.frame(id = 1:ncol(gse21942),expression = y0,type = x0)
p = ggplot(df,aes(x= id,y = expression,color = type)) + geom_boxplot()
ggsave(paste0(dirTamiFigures,"DDR1THRA_b.png"),p)
```

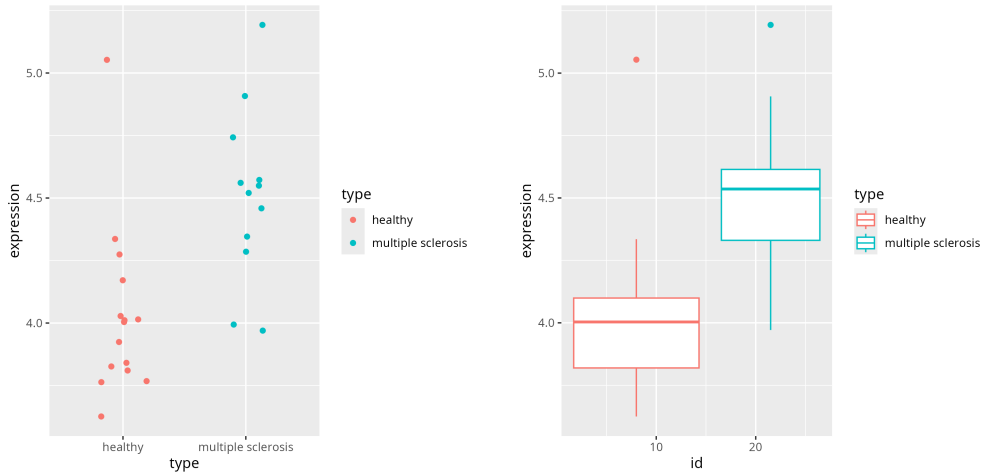


Figura 5.1: a) Perfil de expresión para el gen en fila 1345 utilizando casos y controles. b) Diagrama de cajas para el mismo gen en los dos grupos.

Este es el problema planteado: dos grupos de valores indicando los niveles de expresión bajo dos condiciones y la pregunta fundamental a responder es: ¿son ambos grupos de valores, ambos grupos de niveles de expresión, similares entre sí o difieren claramente uno de otro?

Tenemos datos relativos a niveles de expresión bajo dos condiciones experimentales. En nuestro caso, la condición experimental nos indica si hay esclerosis múltiple o no. ¿Son las expresiones parecidas en ambas condiciones? Nos encontramos con el problema de la comparación de dos muestras. Tenemos un problema de contraste de hipótesis.[6] La hipótesis nula sería que ambas muestras han sido obtenidas de una misma población. Que las diferencias que podamos observar se deben a la variabilidad presente en esta única población.

Tenemos distintas opciones para contrastar estas hipótesis, desde la opción parámetricas que nos obliga a asumir hipótesis sobre las poblaciones que comparamos mientras que también podemos utilizar procedimientos que no piden más que saber que las muestras son aleatorias (datos obtenidos independientemente de las mismas poblaciones). Veamos su uso con los datos. Apliquemos cada una de las tres opciones.

[6] En nmr-§ 6 se trata el problema del contraste de hipótesis de un modo introductorio.

5.5.1 t-test para dos muestras

Hemos de formular un modelo estocástico y, utilizando el modelo, contrastar una hipótesis.[7] De un modo simple: denotamos por Y_1 el nivel de expresión aleatorio que observamos bajo la primera condición (representa un dato observado en la primera población a comparar) y por Y_2 lo mismo pero con la segunda condición (segunda población que comparamos). Asumimos que ambas variables siguen una distribución normal con medias distintas y varianzas iguales. Es decir, asumimos que $Y_i \sim N(\mu_i, \sigma^2)$ para $i = 1, 2$. También asumimos que tenemos una muestra aleatoria de Y_i $(Y_{i1}, \ldots, Y_{in_1})$ y otra de Y_2 $(Y_{21}, \ldots, Y_{2n_2})$. Nos planteamos el contraste de hipótesis:

$$\begin{aligned} H_0: &\quad \mu_1 = \mu_2, \\ H_1: &\quad \mu_1 \neq \mu_2. \end{aligned}$$

El estadístico del contraste es

$$T = \frac{\bar{Y}_1 - \bar{Y}_2}{S_p\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}}, \tag{5.3}$$

siendo $\bar{Y}_i = \sum_{j=1}^{n_i} Y_{ij}/n_i$ para $i = 1, 2$ y

$$S_p^2 = \frac{(n_1 - 1)S_1^2 + (n_2 - 1)S_2^2}{n_1 + n_2 - 2}.$$

Bajo la hipótesis nula de que las medias son la misma el estadístico T definido en (**??**) sigue una distribución t de Student con $n_1 + n_2 - 2$ grados de libertad: $T \sim t_{n_1+n_2-2}$. Rechazamos la hipótesis nula cuando el módulo de T es grande. Por tanto, el p-valor correspondiente a un valor observado t_0 de T viene dado por $p = P(|T| > t_0|H_0)$.

Utilizamos los datos `tamidata::gse21942`.

```
data(gse21942,package="tamidata")
```

`stats::t.test()`

Podemos contrastar si el gen en la fila 1345 tiene una expresión significativamente distinta entre las condiciones que nos definen dos grupos: sanos y enfermos.

```
(tprobeRow = t.test(exprs(gse21942)[probeRow,]~
                    pData(gse21942)[,"FactorValue..DISEASE.STATE."],
                var.equal=TRUE))
```

```
        Two Sample t-test

data: exprs(gse21942)[probeRow, ] by pData(gse21942)[, "FactorValue..
    ↪ DISEASE.STATE."]
t = -3.5777, df = 25, p-value = 0.001452
alternative hypothesis: true difference in means between group healthy
    ↪  and group multiple sclerosis is not equal to 0
95 percent confidence interval:
 -0.7543060 -0.2031418
sample estimates:
          mean in group healthy
                       4.029890
mean in group multiple sclerosis
                       4.508614
```

[7] La comparación de dos poblaciones asumiendo normalidad se estudia en nmr-§ 7.

Podemos ver que el valor del estadístico es -3.5777012 y el p-valor es 0.0014522. Con un nivel de significación $\alpha = 0.05$ no rechazaríamos la hipótesis nula de igualdad de las medias. En la figura 5.2 ilustramos la densidad utilizada con el estadístico y el significado del p-valor correspondiente.

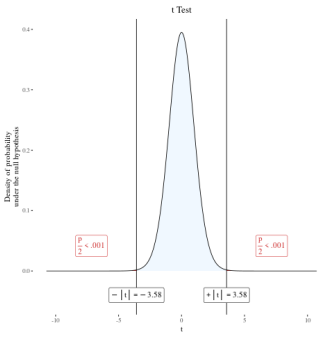


Figura 5.2: Comparamos las dos condiciones para la sonda en fila 1345 de tamidata::gse21942 utilizando un test de la t con varianzas iguales.

`genefilter::rowttests()`

Vamos a repetir el análisis para cada una de las sondas utilizando `genefilter::rowttests()`.

```
tt.gse21942 = genefilter::rowttests(gse21942,
                                    pData(gse21942)[,"FactorValue..DISEASE.STATE."])
```

Se está realizando un t-test para cada fila de la matriz de expresión. Podemos ver las primeras filas.

```
head(tt.gse21942,n=2)
```

```
        statistic dm p.value
1007_s_at -3.077949 -0.2071239 0.005003033
1053_at 3.369555 0.2163679 0.002445187
```

Cada fila corresponde a un gen. La primera columna (`statistic`) nos muestra el estadístico t del contraste de hipótesis para la igualdad de las medias que asume varianzas iguales nmr-(7.5). La segunda columna (`dm` ↪) nos muestra la diferencia de medias y la última columna (`p.value` ↪) nos da el p-valor del contraste. Comprobemos que obtenemos para el gen previamente analizado con `stats::t.test()` obtenemos los mismos resultados.

```
tt.gse21942[probeRow,]
```

```
           statistic dm p.value
1554564_a_at -3.577701 -0.4787239 0.00145221
```

Para cada gen tenemos pues un p-valor. ¿Podemos seguir aplicando el criterio habitual de rechazar la hipótesis nula cuando el p-valor es inferior al nivel de significación α previamente elegido? Si lo hacemos en este ejemplo, ¿cuándo tests mostrarían una expresión diferencial entre ambos grupos?

```
table(tt.gse21942[,"p.value"] < 0.05)
```

```

FALSE TRUE
15011 6347
```

Como vemos tenemos demasiados genes que muestran expresión diferencial. Y no parece muy razonable esto. ¿Con qué problema nos estamos encontrando? El nivel de significación α es una cota superior del error tipo I cuando realizamos **un solo test**. Pero no estamos aplicando un solo test. En concreto con estos datos acabamos de realizar miles de tests (tantos como filas en la matriz de expresión) que además son dependientes entre si. Estamos rechazando muchas hipótesis nulas (no diferencia entre grupos para cada gen) cuando no deberíamos de hacerlo.

Sin embargo, cuando realizamos muchos test: ¿qué es el error tipo I? Ya no es claro cómo cuantificamos los errores que cometemos y, de hecho, tenemos que definir nuevas tasas de error para cuantificar y, sobre todo, controlar los errores que cometemos cuando realizamos *simultáneamente* muchos tests. Lo tratamos más adelante.

5.5.2 Test no paramétrico

No tendríamos porqué asumir normalidad y homogeneidad de las varianzas. Podemos limitarnos a asumir que tenemos observaciones independientes obtenidas de dos poblaciones distintas y compararlas utilizando un test no paramétrico. Una opción común es el test de Kolmogorov-Smirnov (nmr-§ 7.6, [61, sección 6.3, pág. 239]).

```
(ksprobeRow = ks.test(exprs(gse21942)[probeRow,]~
                      pData(gse21942)[,"FactorValue..DISEASE.STATE."]))
```

```
        Exact two-sample Kolmogorov-Smirnov test

data: exprs(gse21942)[probeRow, ] by pData(gse21942)[, "FactorValue..
    ↪ DISEASE.STATE."]
D = 0.7, p-value = 0.001305
alternative hypothesis: two-sided
```

También podríamos calcular los p-valores del test de Kolmogorov-Smirnov para cada una de las filas de la matriz de expresión.

```
ks.gse21942 = sapply(1:nrow(gse21942),function(i)
    ks.test(exprs(gse21942)[i,]~
                pData(gse21942)[,"FactorValue..DISEASE.STATE."])$p.value
    )
```

¿Cuántas sondas son significativas?

```
table(ks.gse21942 ≤0.05)
```

```
FALSE TRUE
15916 5442
```

5.5.3 Un test de permutación de Fisher-Pitman

En § 5.5.1 hemos asumido un modelo paramétrico. Obviamente habría que evaluar estas hipótesis. Si no lo hacemos puede que se verifiquen y podamos aplicar el test. Sin duda, cuando lo aplicamos a miles de características (sondas en nuestro caso) se verificarán algunas veces y otras no. Es por tanto dudoso asumir **siempre** estas hipótesis. El procedimiento no paramétrico propuesto en § 5.5.2 no tiene este problema. Sin embargo, suelen ser menos potentes (probabilidad de rechazar la hipótesis nula cuando es falsa). Otra opción a utilizar sería un test de permutación [93, capítulo 1]. En esta opción tenemos los datos a comparar denotados con $\{y_{11}, \ldots, y_{1n_1}\}$ en el primer grupo e $\{y_{21}, \ldots, y_{2n_2}\}$ en el segundo grupo. Elegimos un estadístico que nos sirva para comparar ambos grupos de datos y que denotamos por T. Supongamos que el valor calculado con las muestras originales lo denotamos por t_0. Por ejemplo, podemos elegir (aunque no sería obligatorio) el estadístico del contraste de la t de Student. Tenemos un total de $n_1 + n_2$ datos si juntamos los dos grupos: $S_y = \{y_{11}, \ldots, y_{1,n_1}, y_{21}, \ldots, y_{2,n_2}\}$. Tenemos dos opciones. La primera opción es extraer sin reemplazamiento n_1 datos de S_y y considerar que han sido extraidos de la primera población que queremos comparar. El resto de valores asumimos que vienen de la segunda población. Tenemos pues ahora los conjuntos $S_1^* = \{y_{11}^*, \ldots, y_{1,n_1}^*\}$ y $S_2^* = \{y_{21}^*, \ldots, y_{2,n_1}^*\}$. El estadístico T evaluado en esta nueva muestra podemos denotarlo con

T_1. Esto podemos repetirlo un total de $\binom{n_1+n_2}{n_1}$ formas distintas. Repetimos el proceso de selección aleatoria un total de B veces y los sucesivos valores de T que vamos obteniendo los denotamos por $t_1, \ldots, t_B$. Bajo la hipótesis nula de que ambas poblaciones que comparamos son la misma cualquier ordenación de $(t_0, t_1, \ldots, t_B)$ es igualmente probable. Supongamos que valor grande de T se corresponde con la hipótesis alternativa de que ambas poblaciones no son la misma. Entonces tendremos un p-valor de aleatorización dado por

$$p_a = \frac{|\{b : t_b > t_0, b = 1, \ldots, B\}|}{B+1}. \tag{5.4}$$

Si valores pequeños de T se asocian con la hipótesis alternativa entonces el p-valor anterior sería

$$p_a = \frac{|\{b : t_b < t_0, b = 1, \ldots, B\}|}{B+1}. \tag{5.5}$$

Si finalmente valores grandes o pequeños de T corresponden entonces la hipótesis alternativa entonces el p-valor correspondería con

$$p_a = \frac{|\{b : |t_b| > |t_0|, b = 1, \ldots, B\}|}{B+1}. \tag{5.6}$$

Del total de datos S_y hemos realizado extracciones sin reeemplazamiento. Por ello hablamos de **distribución de permutación**.

En el código que sigue el estadístico es la diferencia de medianas.

```
pacman::p_load(coin)
(perprobeRow = coin::oneway_test(exprs(gse21942)[probeRow,]~
                                 pData(gse21942)[,"FactorValue..DISEASE.STATE."]))
```

```
        Asymptotic Two-Sample Fisher-Pitman
        Permutation Test

data: exprs(gse21942)[probeRow, ] by
         pData(gse21942)[, "FactorValue..DISEASE.STATE."] (healthy,
              ↪ multiple sclerosis)
Z = -2.9672, p-value = 0.003005
alternative hypothesis: true mu is not equal to 0
```

En particular el p-valor lo tenemos con

```
pvalue(perprobeRow)
```

```
[1] 0.003005365
```

Otra vez podemos repetir el análisis para todas las filas.

```
per.gse21942 = sapply(1:nrow(gse21942),function(i)
    pvalue(coin::oneway_test(exprs(gse21942)[i,]~
                 pData(gse21942)[,"FactorValue..DISEASE.STATE."])))
```

Y como en los casos anteriores podemos ver cuántos son declarados significativos.

```
table(per.gse21942<.05)
```

```
FALSE TRUE
15118 6240
```

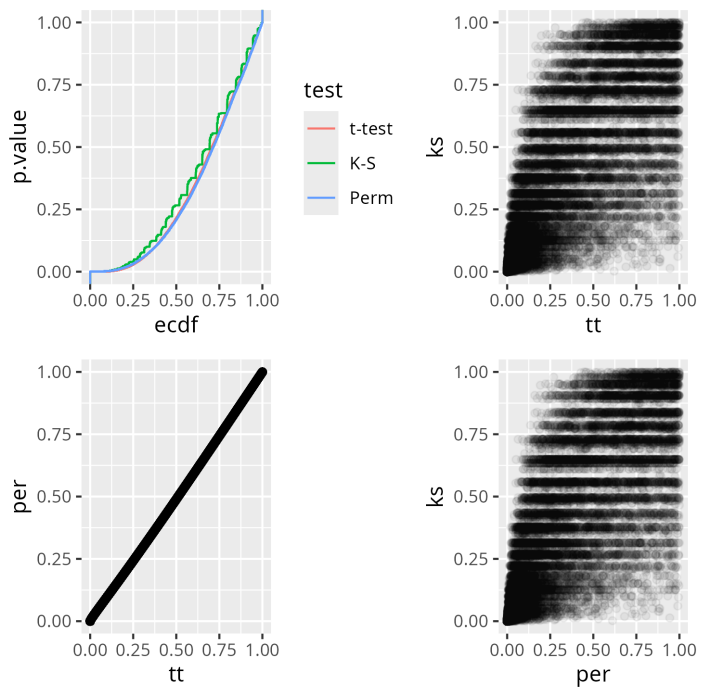


Figura 5.3: Arriba-izquierda: las funciones de distribución acumuladas para cada uno de los métodos. Arriba-derecha: Los p-valores de Kolmogorov-Smirnov frente a los del test de la t. Abajo-izquierda: p-valores de permutación frente a test de la t. Abajo-derecha: p-valores de permutación frente a los de Kolmogorov-Smirnov.

5.5.4 Comparando los tres procedimientos

Vamos a comparar las tres opciones que hemos propuesto en las tres secciones previas § 5.5.1, § 5.5.2 y § 5.5.3. En la figura 5.3 mostramos las comparaciones entre cada par de métodos así como sus funciones de distribución acumuladas. Destaca que los resultados del test de la t son muy simulares a los obtenidos por el método de permutación. Y ambos difieren en cuanto a los resultados obtenidos mediante el test de Kolmogorov-Smirnov.

```
pacman::p_load(ggplot2,patchwork) ## patchwork combina gráficos
df1 = data.frame(test = factor(rep(1:3,each=nrow(gse21942)),
                               labels=c("t−test","K−S","Perm")),
                 p.value = c(tt.gse21942$p.value,ks.gse21942,per.gse21942))
p1 = ggplot(df1,aes(y=p.value,color=test))+stat_ecdf()
df2 = data.frame(tt = tt.gse21942$p.value,ks = ks.gse21942,per = per.gse21942)
p2 = ggplot(df2,aes(x = tt, y=ks))+geom_point(alpha=1/20)
p3 = ggplot(df2,aes(x = tt, y=per))+geom_point()
p4 = ggplot(df2,aes(x = per, y=ks))+geom_point(alpha=1/20)
```

Supongamos que declaramos significativos cuando el p-valor esté por debajo de 0.05 según cada uno de los tres procedimientos considerados. Vamos a utilizar un diagrama de Venn para comparar gráficamente las **sondas** significativas.

```
pacman::p_load(ggVennDiagram)
x = list(tt = fData(gse21942)$PROBEID[df2$tt ≤.05],
         ks = fData(gse21942)$PROBEID[df2$ks ≤.05],
```

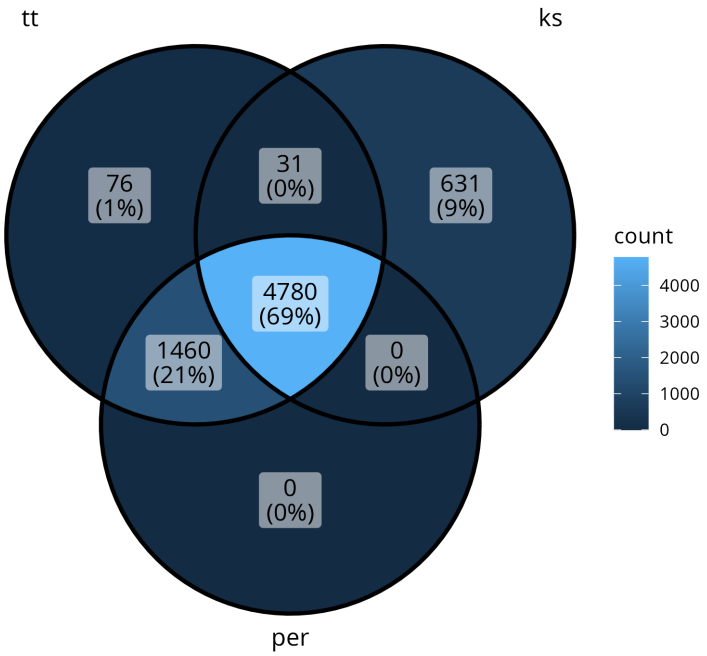


Figura 5.4: Diagrama de Venn comparando las sondas significativas (sin eliminar multiplicidad entre sonda y gen) para los tres procedimientos considerados.

```
        per = fData(gse21942)$PROBEID[df2$per ≤.05])
p = ggVennDiagram(x)
```

En la figura 5.4 tenemos el diagrama de Venn. Vemos una gran coincidencia entre los tres procedimientos estadísticos utilizados. Vemos también que hay los dós métodos (como era esperable por las comparaciones previas de p-valores) que más coinciden son el test de la t y el método de permutaciones de Fisher-Pitman.

5.6 Ejercicios

Filtrado independiente

Ex. 7 — Con los datos `multtest::golub` y utilizando las funciones `base::apply()`, `stats::sd()`, `stats::IQR()` se pide:

1. Determinar para cada gen (esto es, las expresiones de una fila dada) el rango intercuartílico.
2. Una vez hemos calculado el rango intercuartílico para gen calcular el percentil de orden 0.50 (o mediana).
3. Determinar las filas correspondientes a los genes cuyo rango intercuartílico supera el percentil que hemos calculado en el apartado anterior.
4. Determinar aquellas filas donde al menos 10 muestras de las 38 superan una expresión de 0.

5. Seleccionar en la matriz de expresión golub las filas que verifican los criterios dados en los puntos 3 y 4.

** **Ex. 8 —** Utilizando el paquete [59, genefilter] se pide diseñar un filtrado no específico para los datos `multtest::golub` y que seleccione los genes atendiendo a los siguientes criterios:

1. Determinamos para cada gen el coeficiente de variación.
2. Determinamos el percentil 0.90 del los coeficientes de variación para todos los genes.
3. El coeficiente de variación ha de superar el percentil 0.9 de los coeficientes de variación observados.
4. Repetimos los puntos anteriores sustituyendo el coeficiente de variación por el nivel de expresión medio.
5. Conservamos los genes que verifican los dos criterios de selección.

** **Ex. 9 —** Realizar una selección no específica de los datos [83, ALL] utilizando el paquete [59, genefilter]. Los criterios de selección son los siguientes:

1. La mediana de los niveles de expresión del gen ha de superar el percentil 0.9 de las medianas observadas para todos los genes.
2. El rango intercuartílico de los niveles de expresión del gen ha de superar el percentil 0.9 de los rangos intercuartílicos observados para todos los genes.

Ex. 10 — Utilizando los datos `tamidata::gse21942` y la variable fenotípica `FactorValue..DISEASE.STATE.` vamos a aplicar un test de Kolmogorov-Smirnov (nmr-§ 7.6) para comparar los grupos de sanos y enfermos. Se pide:

1. Comparar los dos grupos utilizando `stats::ks.test()` para la sonda en la fila probeRow de la matriz de expresión.
2. Realizar la la comparación para todas las sondas. Se recomienda utilizar `sapply()`.

Expresión diferencial marginal

* **Ex. 11 —** Utilizando los datos `tamidata::gse1397` y el factor `pData(`
↪ `gse1397[,"type"]` realizar un análisis de expresión diferencial marginal utilizando el test de la t, el test de Kolmogorov-Smirnov así como el test de Fisher-Pitman. Comparar los resultados obtenidos mediante un diagrama de Ven. Determinamos significativos cuando los p–valores originales estén por debajo del valor 0.01.

Capítulo 6

Modelos lineales

En este tema pretendemos repasar conceptos básicos de regresión lineal múltiple. Es lo básico que necesitamos para entender los datos en que la respuesta es numérica y tenemos variables predictoras (variables fenotípicas en este contexto) que pretendemos ver cómo influyen en la expresión del gen (variable respuesta de un modo genérico).

Lo que veamos en este capítulo es de aplicación cuando trabajamos microarrays de DNA, metilación, microarrays de proteínas.

Utilizamos para ilustrar los datos `tamidata2::gse25171`.

```
pacman::p_load(Biobase)
data(gse25171,package="tamidata2")
```

Veamos las variables fenotípicas de las que disponemos.

```
head(pData(gse25171),n=2)
```

```
              time time2 Pi replication
GSM618324.CEL.gz 0 Short Treatment 1
GSM618325.CEL.gz 0 Short Control 2
```

Vamos a plantearnos cómo pueden influir el tiempo de observación de la muestra (**`time`**) y la presencia o no de fósforo en la expresión del gen (`Pi`).

Elegimos la sonda en fila 15639. Construimos un **`data.frame`** con la información necesaria en donde recogemos los dos predictores (**`time`** y `Pi`) y la variable respuesta que sería la fila de la matriz de expresión correspondiente a la sonda considerada.

```
sel0 = 15639
df0 = data.frame(pData(gse25171)[,c("time","Pi")],
                 expression=exprs(gse25171)[sel0,])
```

6.1 Sobre lo que vamos a tratar

6.1.1 Problemas y datos

¿Qué problemas queremos resolver? ¿Qué información tenemos para resolverlos? Empezamos por la información. Tenemos información (numérica, categórica) sobre una serie de muestras u observaciones. Pueden tomar valores fijados por nosotros o bien valores observados (realizaciones de variables aleatorias). No todas las variables de las que disponemos tienen el mismo interés. Habitualmente hay una variable **importante**:

expresión de un gen, número de lecturas alineadas. A esta variable de interés la denotamos como y_i para la i-ésima observación y la llamaremos **variable respuesta** (en denominaciones más clásicas, variable dependiente). El resto de variables del estudio son las variables predictoras o variables independientes o (menos frecuente) **regresores** o **inputs**. Habitualmente en nuestro caso serán las variables fenotípicas que nos describen las muestras. Para la i-ésima observación tenemos p variables predictoras que recogemos en el vector columna $\boldsymbol{x}_i \in \mathbb{R}^p$ donde

$$\boldsymbol{x}_i = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ip} \end{bmatrix}.$$

Denotamos el vector traspuesto como $\boldsymbol{x}_i^T = (x_{i1}, \ldots, x_{ip})$. Nuestra información consiste en los pares $(\boldsymbol{x}_i, y_i)$ con $i = 1, \ldots, n$.

¿Qué problema queremos resolver? Una respuesta fácil es decir que queremos conocer el valor de la variable respuesta utilizando las variables predictoras. ¿Solamente queremos conocer el valor de la respuesta? Quizás estamos pensando en un futuro en donde conozcamos las variables predictoras y nos interese saber cuál será el valor de la variable respuesta correspondiente. Sin embargo, conocer el valor de la respuesta tiene distintas interpretaciones posibles: ¿El valor exacto de la respuesta? ¿La media de la respuesta? ¿Un valor numérico que aproxime cada una de estas cantidades o bien un intervalo que las contenga?

Los valores observados y_i consideraremos que son realizaciones de variables aleatorias Y_i.[73] Realmente en lo que sigue modelizaremos el comportamiento aleatorio de la variable Y_i condicionada a los valores observados $\boldsymbol{x}_i$. Es decir, nuestro interés estará en *la distribución condicionada*: tenemos el vector aleatorio $\boldsymbol{X}_i$ (donde $\boldsymbol{x}_i$ son los valores observados) y la variable aletoria Y_i observadas conjuntamente. Entonces el vector $(\boldsymbol{X}_i^T, Y_i)^T$ tendrá densidad conjunta[74] $f(y_i, \boldsymbol{x}_i)$ y podemos considerar la densidad condicionada $f(y_i|\boldsymbol{x}_i)$ (o la probabilidad condicionada P_i). La media condicionada que nos interesa es, cuando tenemos una distribución (absolutamente) continua respecto de la medida de Lebesgue, la siguiente

$$\mu_i = E[Y_i|\boldsymbol{x}_i] = \int_{-\infty}^{+\infty} y_i f(y_i|\boldsymbol{x}_i) dy_i. \tag{6.1}$$

Si consideramos una distribución discreta será

$$\mu_i = E[Y_i|\boldsymbol{x}_i] = \sum_{y_i} y_i f(y_i|\boldsymbol{x}_i). \tag{6.2}$$

Y si consideramos una variable que no es ni discreta ni continua y denotamos por P_i la probabilidad condicional entonces

$$\mu_i = E[Y_i|\boldsymbol{x}_i] = \int_{-\infty}^{+\infty} y_i P_i(dy_i). \tag{6.3}$$

En lo que sigue vamos a asumir que las correspondientes distribuciones condicionadas son independientes entre si: las Y_i serán condicionalmente independientes.

Nuestro interés fundamental (pero no único) estará en conocer las medias condicionadas $\mu_i = E[Y_i|\boldsymbol{x}_i]$. La variable respuesta podrá ser cuantitativa (continua o discreta) o cualitativa (posiblemente ordinal). Las variables predictoras pueden ser numéricas o categóricas.

[73] Denotamos con mayúsculas las variables aleatorias y con minúsculas los valores observados.

[74] Entendemos densidad en su sentido más genérico incluyendo distribuciones continuas, discretas y distribuciones que no son ni continuas ni discretas.

6.1.2 Modelos sobre la media

En la sección anterior una de las opciones que nos planteamos de conocer la respuesta es conocer la media de la respuesta aleatoria. Tenemos unas variables aleatorias. Queremos conocer esa respuesta media cuando están dadas estas variables predictoras. En definitiva pretendemos conocer la media de la respuesta aleatoria **condicionada** a los valores de las variables predictoras. Si denotamos por Y la respuesta aleatoria y por $\boldsymbol{x}$ las variables predictoras[1] nuestro interés es conocer $E[Y|\boldsymbol{x}]$. Si consideramos la i-ésima respuesta tendremos $E[Y_i|\boldsymbol{x}_i]$. Para simplificar esta notación denotaremos simplemente $\mu_i = E[Y_i|\boldsymbol{x}_i]$ sin indicar explícitamente las variables predictoras.

6.1.3 Dependencia lineal

¿Qué tipos de dependencia vamos a considerar? En la mayor parte de los casos dependencias de tipo lineal. En lo que sigue vemos cómo expresar dependencias de la media condicionada μ_i respecto de los predictoras cuando estos son números o categóricos o numéricos y categóricos. Suponemos dos predictores $\boldsymbol{x} = (x_1, x_2)^T$ tales que x_1 es numérico y el segundo es una variable categórica binaria codificada con 1 y 0.[75] ¿Cómo modelizamos la dependencia de la media condicionada respecto de x_1? Una dependencia lineal vendría dada como

[75] Indicando presencia o ausencia de un atributo.

$$\mu_i = \beta_0 + \beta_1 x_{i1}. \tag{6.4}$$

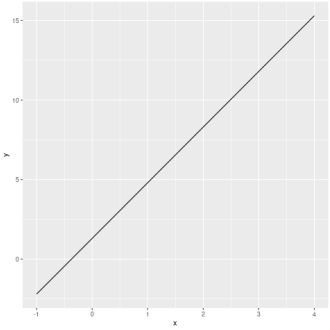


Figura 6.1: Modelo para la media según modelo en ecuación 6.4.

¿Y la dependencia de las μ_i respecto de la variable binaria? Obviamente simplemente tenemos dos valores. Un modo simple es

$$\mu_i = \beta_0 + \beta_2 x_{i2}. \tag{6.5}$$

Si lo hacemos así tenemos que cuando $x_{i2} = 0$ entonces $\mu_i = \beta_0$ mientras que cuando $x_{i2} = 1$ tendremos $\mu_i = \beta_0 + \beta_2$.

¿Y las dos variables predictoras conjuntamente consideradas? Quizás el modelo más sencillo sería:

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2}. \tag{6.6}$$

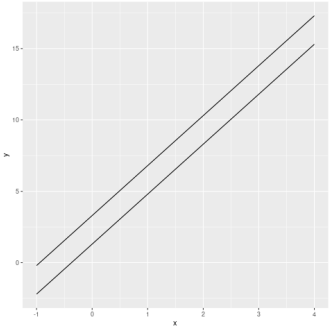


Figura 6.2: Modelo para la media en ecuación 6.6.

Si consideramos este modelo tendremos que estamos realmente especificando dos modelos para la media. Cuando $x_{i2} = 0$ entonces $\mu_i = \beta_0 + \beta_1 x_{i1}$. Cuando $x_{i2} = 1$ entonces $\mu_i = \beta_0 + \beta_2 + \beta_1 x_{i1}$. Realmente estamos modificando la ordenada en el origen de la línea recta. En la figura 6.2 mostramos ambas líneas. Tenemos pues dos líneas paralelas.

¿Como podemos expresar la dependencia de ambas covariables? Otra vez recurrimos a la opción de expresarlo de un modo lineal.

$$\mu_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \beta_3 x_{i1} x_{i2} \tag{6.7}$$

Tendremos dos líneas que expresan la dependencia. Cuando $x_{i2} = 0$ tenemos

$$\mu_i = \beta_0 + \beta_1 x_{i1}. \tag{6.8}$$

Cuando $x_{i2} = 1$ tenemos

$$\mu_i = (\beta_0 + \beta_2) + (\beta_1 + \beta_3) x_{i1}. \tag{6.9}$$

[1] Observemos que denotamos y denotaremos el valor aleatorio en mayúscula mientras que las variables predictoras las consideramos dadas, observadas, no aleatorias y por lo tanto las denotamos en minúscula.

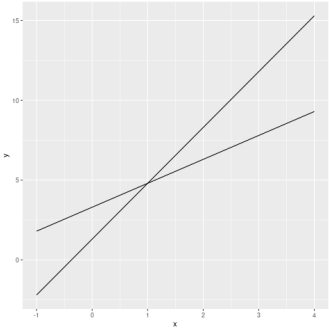


Figura 6.3: Modelo para la media propuesto en ecuación 6.9.

Vemos que se modifica tanto la ordenada en origen como la pendiente de la recta. Hay, lo que luego llamaremos, una interacción entre las dos variables predictoras.

Cuando consideramos una variable predictora categórica con más de dos categorías entonces es habitual codificarla utilizando variables tontas.[76] Si la variable predictora categórica tiene I categorías entonces se elige una categoría de referencia, por ejemplo la primera,[77] entonces las variables binarias asociadas a la variable original x serían: $v_1 = 1$ si $x = 2$ y cero en otro caso; $v_2 = 1$ si $x = 3$ y cero en otro caso; ...; $v_{I-1} = 1$ si $x = I$ y cero en otro caso. Obviamente cuando todas las variables v son nulas estamos en la primera categoría. Si solamente tenemos la variable categórica como predictora entonces la media sería función lineal de x del siguiente modo:

$$\mu_i = \beta_0 + \beta_1 v_1 + \ldots + \beta_{I-1} v_{I-1}.$$

Si como variables predictoras tenemos distintas variables algunas numéricas y otras categóricas (con dos o más de dos categorías) tendremos el mismo modo de modelizar que acabamos de ver simplemente añadiendo más términos.

Por ejemplo, supongamos una numérica x y una categórica v con I categorías. Construimos las variables tontas siendo I la de referencia. Podemos considerar modelos como

$$\mu_i = \beta_0 + \beta_1 x_i, \tag{6.10}$$

que lo expresamos como $y \sim x$. Un modelo que contiene solamente a v sería el dado previamente

$$\mu_i = \beta_0 + \beta_1 v_{i1} + \ldots + \beta_{I-1} v_{i,I-1}. \tag{6.11}$$

y lo expresamos como $y \sim v$. Un modelo que contempla ambas variables puede ser

$$\mu_i = \beta_0 + \beta_1 x_i + \beta_2 v_{i1} + \ldots + \beta_I v_{i,I-1} \tag{6.12}$$

Este modelo lo podemos abreviar (y así se le indicará a R) como $y \sim x+v$. Un modelo más completo que contempla la posible interacción sería

$$\begin{aligned}\mu_i = \beta_0 + \beta_1 x_i + \beta_2 v_{i1} + \ldots + \beta_I v_{i,I-1} + \\ \beta_{I+1} x_i v_{i1} + \ldots + \beta_{2I-1} x_i v_{i,(I-1)}\end{aligned} \tag{6.13}$$

Esto lo indicaremos como $y \sim x * v$ o bien como $y \sim x + v + x : v$.

Supongamos dos variables categóricas u y v con I y J categorías respectivamente. Tendríamos el modelo $y \sim u$ y el modelo $y \sim v$ en donde solo se considera la influencia o efecto de cada una de ellas aislada o marginalmente. Un modelo más completo sería

$$\begin{aligned}\mu_i = \beta_0 + \\ \beta_1 u_{i1} + \ldots + \beta_I u_{i,I-1} + \\ \beta_{I+1} v_{i1} + \ldots + \beta_{2I-1} v_{i,(I-1)} + \\ \beta_{I+1} u_{i1} v_{i1} + \ldots + \beta_{3I-2} u_{i1} v_{i,(I-1)} + \ldots + \\ \beta_{I+1} u_{i(I-1)} v_{i1} + \ldots + \beta_{(I-1)^2+2(I-1)+1} u_{i(I-1)} v_{i,(I-1)}\end{aligned} \tag{6.14}$$

Esto se indicaría como $y \sim u * v$.

En la tabla 6.1 se indica cómo se suele denominar al modelo resultante. Se utiliza en esta tabla la notación propuesta en [152] y que utilizaremos extensamente en el curso. Es lo que se conoce como `formula` en R.

[76] Dummy variables.

[77] Cada software por defecto elige una y podemos modificarla. En R la categoría de referencia es la primera.

Tabla 6.1: Especificaciones simplificadas de los modelos.

Modelo	Notación en R
Regresión lineal simple	$y \sim x$
Regresión lineal múltiple	$y \sim x_1 + \ldots + x_p$
Anova de una vía	$y \sim u$
Anova de dos vías sin interacción	$y \sim u + v$
Anova de dos vías con interacción	$y \sim u + v + u : v$
	$y \sim u * v$

6.1.4 Efectos

Cuando se habla de efectos en modelos lineales nos referimos a los parámetros. El *efecto* que producen en la variable respuesta es a traves de este coeficiente. ¿Cómo interpretamos los coeficientes en un modelo lineal? El caso más simple sería un modelo con una sola covariable o variable predictora: $\mu_i = \beta_0 + \beta_1 x_{i1}$. Parece natural y simple decir algo como: un incremento unitario en la variable x se traduce en un incremento unitario de la media. Formalmente es correcta la afirmación. Sin embargo, en términos de interpretación la cosa no es tan correcta. Si pudiéramos realizar un experimento y sobre la misma unidad experimental modificar el valor de x y ver su efecto entonces sí que sería correcto. Esto no es lo habitual. Por ello, un interpretación mejor podría ser la siguiente: Consideramos dos subpoblaciones formadas por los individuos donde el valor de la covariable es x y la formada por los individuos donde la covariable toma el valor $x + 1$. La diferencia de medias entre ambas subpoblaciones es el coeficiente β_1. Una cosa es el formalismo matemático y otra cosa es la interpretación estadística.[78]

[78] Este manual está lleno de comentarios de este tipo.

Supongamos que tenemos más de una covariable. Nuestro modelo ahora es $\mu_i = \beta_0 + \beta_1 x_{i1} + \ldots + \beta_{p-1} x_{i,p-1}$. ¿Cómo interpretamos el valor de β_1, el efecto de la variable x_{i1} sobre la variable respuesta? La afirmación siguiente es matemáticamente correcta: si mantenemos todas las demás covariables constantes e incrementamos en una unidad la covariable x_{i1} entonces la media cambia en β_1 unidades. Correcto matemáticamente lo es. Pero, ¿es posible? En ocasiones no podemos mantener constantes todas las covariables y modificar el valor de x_{i1}. ¿Por qué? Porque puede haber dependencias entre ellas. Cambiar el valor de x_{i1} supone cambiar el valor de las otras si trabajamos con datos observacionales, no controlados. Incluso pueden darse combinaciones de las covariables imposibles. Pensemos que una covariable puede ser el sexo de la persona y valores de otras covariables solamente se pueden dar para un sexo y no para otro. Otra vez: ¿cómo interpretamos β_1? Consideramos otra vez dos subpoblaciones con valores x_{i1} y $x_{i1} + 1$ para la primera covariable. Y suponemos que, en ambas subpoblaciones, el valor de $\beta_2 x_{i2} + \ldots + \beta_1 x_{ip}$ es el mismo. La diferencia de medias de la respuesta en ambas subpoblaciones viene dada por β_1. Dicho de una manera más técnica, β_1 es la diferencia de medias cuando modificamos en una unidad x_{i1} **ajustando** por el resto de las covariables.

6.1.5 Matriz modelo

Tenemos el vector de medias a estimar

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_n \end{bmatrix}.$$

Podemos considerar la matriz que, en cada fila, tenga los predictores correspondientes a la i-ésima observación

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \dots & x_{1p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \dots & x_{np} \end{bmatrix}.$$

Esta matriz recibe el nombre de **matriz modelo**.[79] De hecho, los ejemplos que hemos comentado previamente todos verifican que

[79] A veces, en diseño de experimentos se habla de **matriz de diseño**.

$$\mu_i = \sum_{j=1}^{p} \beta_j x_{ij},$$

por tanto, podemos expresar esta dependencia como

$$\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta},$$

siendo el vector de coeficientes

$$\boldsymbol{\beta} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_p \end{bmatrix}.$$

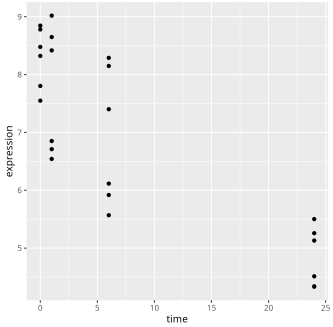
Pretendemos estimar $\boldsymbol{\mu} = E\boldsymbol{Y}$. Tenemos unas covariables o predictores. ¿En dónde estamos jugando? ¿Qué conjunto de posibles valores para $\boldsymbol{\mu}$ tenemos si asumimos un modelo lineal? Puesto que asumimos $\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}$ entonces el conjunto vendría dado por

$$C(\boldsymbol{X}) = \{\boldsymbol{\eta} : \boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta}, \boldsymbol{\beta} \in \mathbb{R}^p\},$$

o lo que es lo mismo, el espacio vectorial generado por las columnas de la matriz de modelo $\boldsymbol{X}$. A este espacio vectorial lo podemos llamar **espacio modelo**.

6.2 Regresión lineal simple

En un primer momento consideramos solamente la variable predictora (fenotípica en nuestro contexto) `time`. En figura 6.4 tenemos el tiempo en abscisas y la expresión observada en la sonda previamente considerada.




```
pacman::p_load(ggplot2)
p = ggplot(df0,aes(x=time,y=expression))+geom_point()
ggsave(paste0(dirTamiFigures,"gse25171_261892_at.png"),p)
```

Figura 6.4: Una sonda de tamida-

6.2.1 Recta de mínimos cuadrados

Nuestros datos son (x_i, y_i) con $i = 1, \ldots, n$ y pretendemos estudiar la posible dependencia de los valores y respecto de los valores x. Una posible dependencia, sin duda, la más simple es considerar que la respuesta es función lineal de la predictora,

$$y_i = \beta_0 + \beta_1 x_i,$$

con $i = 1, \ldots, n$. Obviamente no es posible. No hay solución para las ecuaciones anteriores. Podemos sustituir la idea de resolver las ecuaciones con la de encontrar una **buena** solución aproximada,

$$y_i \approx \beta_0 + \beta_1 x_i,$$

para $i = 1, \ldots, n$. Una posibilidad para determinar unos **buenos** valores para β_0 y β_1 es considerar la siguiente suma de cuadrados S_t

$$S_t = \sum_{i=1}^{n} (y_i - \beta_0 - \beta_1 x_i)^2, \tag{6.15}$$

y considerar los valores de β_0 y β_1 que la minimizan. En modest18-§ 2.2 podemos ver que los estimadores mínimo cuadráticos vienen dados por

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n} x_i y_i - (\sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i)/n}{\sum_{i=1}^{n} x_i^2 - (\sum_{i=1}^{n} x_i)^2/n}, \tag{6.16}$$

y

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}. \tag{6.17}$$

Ya tenemos una buena aproximación que nos permite relacionar la variable predictora con la variable respuesta. Sería la **recta de mínimos cuadrados**:

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x.$$

Consecuencia inmediata de la ecuación (6.17), la recta de mínimos cuadrados se puede escribir como

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x = \bar{y} + \hat{\beta}_1 (x - \bar{x}). \tag{6.18}$$

Y, en particular, a partir de (6.18) se sigue que pasa por el punto $(\bar{x}, \bar{y})$. Si denotamos, como es usual,

$$S_{xy} = \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}), \tag{6.19}$$

y

$$S_{xx} = \sum_{i=1}^{n} (x_i - \bar{x})^2, \;\; S_{yy} = \sum_{i=1}^{n} (y_i - \bar{y})^2, \tag{6.20}$$

entonces

$$\hat{\beta}_1 = \frac{S_{xy}}{S_{xx}}. \tag{6.21}$$

Ejemplo 6.1. *¿Cómo obtenemos con R estos estimadores de la ordenada en el origen y de la pendiente de la curva?*

```
(fit0 = lm(expression ~time, data = df0))
```

```
Call:
lm(formula = expression ~time, data = df0)

Coefficients:
(Intercept) time
    7.9684 -0.1331
```

Los coeficientes de la recta de mínimos cuadrados son

```
coef(fit0)
```

```
(Intercept) time
  7.9684362 -0.1330703
```

Podemos representar la recta de mínimos cuadrados con el siguiente código y aparece en la figura 6.5.

```
p = ggplot(df0,aes(x=time,y=expression))+
    geom_point() + geom_smooth(method='lm', se = FALSE)
```

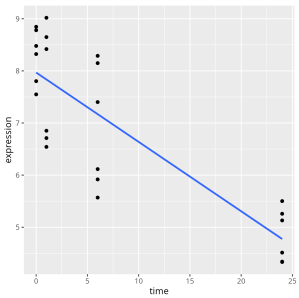


Figura 6.5: Datos de figura 6.5 superponiendo la recta de mínimos cuadrados.

Nos planteábamos en un principio encontrar unas buenas aproximaciones para y_i utilizando x_i. Lo natural sería tomar el siguiente valor la **predicción** o **valor ajustado** para y_i:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_i.$$

Las predicciones, $\hat{y}_i$ las tenemos con (`utils::head()` muestra las primeras),

```
head(predict(fit0))
```

```
GSM618324.CEL.gz GSM618325.CEL.gz
       7.968436 7.968436
GSM618326.CEL.gz GSM618327.CEL.gz
       7.835366 7.835366
GSM618328.CEL.gz GSM618329.CEL.gz
       7.170014 7.170014
```

La diferencia entre valor observado, y_i, y su predicción, $\hat{y}_i$, se conoce como **residuo**. Esencialmente una estimación del error,

$$e_i = y_i - \hat{y}_i.$$

Se prueba que, cuando el modelo tiene una constante, entonces la suma de todos los residuos es nula,

$$\sum_{i=1}^{n} e_i = 0.$$

Los residuos los podemos calcular con

```
head(resid(fit0))
```

```
GSM618324.CEL.gz GSM618325.CEL.gz
      0.3545741 0.8777147
GSM618326.CEL.gz GSM618327.CEL.gz
     -0.9836174 0.5831095
GSM618328.CEL.gz GSM618329.CEL.gz
     -1.2520196 0.9791937
```

6.2.2 Sumas de cuadrados

¿Cuánto de la variación que hay en la variable respuesta ha sido explicada por la regresión? Para responder la pregunta quizás lo mejor sería preguntarnos ¿qué entendemos por la variación de la variable respuesta? Se puede interpretar de dos formas: la primera sería simplemente como

$$\sum_{i=1}^{n} y_i^2,$$

o bien, como variación respecto de la media que vendría dada por

$$\sum_{i=1}^{n} (y_i - \bar{y})^2.$$

De momento supongamos que utilizamos la segunda versión, variación respecto de la media. En lo que sigue vamos a descomponer esta suma de cuadrados en sumandos que tengan una interpretación sencilla. Se tiene

$$y_i - \bar{y} = (y_i - \hat{y}_i) + (\hat{y}_i - \bar{y}). \tag{6.22}$$

Notemos que

$$\sum_{i=1}^{n} \frac{\hat{y}_i}{n} = \sum_{i=1}^{n} \frac{\hat{\beta}_0 + \hat{\beta}_1 x_i}{n} = \hat{\beta}_0 + \hat{\beta}_1 \bar{x} = \bar{y}. \tag{6.23}$$

A partir de (6.22), elevamos al cuadrado y sumamos y tenemos (ver modest18-2.17) que

$$\sum_{i=1}^{n} (y_i - \bar{y})^2 = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2, \tag{6.24}$$

También tenemos

$$\sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2 = \sum_{i=1}^{n} \hat{\beta}_1^2 (x_i - \bar{x})^2 = \frac{S_{xy}^2}{S_{xx}}. \tag{6.25}$$

Consideremos las siguientes sumas de cuadrados.

Suma de cuadrados total

$$SS(total) = \sum_{i=1}^{n} (y_i - \bar{y})^2,$$

Suma de cuadrados debida a la regresión

$$SS(regression) = \sum_{i=1}^{n} (\hat{y}_i - \bar{y})^2,$$

Suma de cuadrados residual

$$SS(residual) = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2.$$

Se tiene la siguiente igualdad

$$SS(Total) = SS(Regression) + SS(Residual). \tag{6.26}$$

Las tres sumas de cuadrados de la ecuación (6.26) suelen disponerse en forma de tabla que recibe el nombre de tabla de análisis de la varianza. La mostramos en la tabla 6.2. Más adelante veremos esta misma tabla y la interpretaremos con detalle. Ahora quizás darse cuenta que estamos considerando un cociente que cuantifica hasta qué punto la suma de cuadrados de la regresión es grande en relación a la suma de cuadrados residual. Este cociente, considerando los grados de libertad, tiene una distribución de probabilidad que es una F de Fisher. En lo que sigue probamos que esto es efectivamente así. Ahora, cuanto mayor es el valor del estadístico (`F value`) o equivalentemente cuanto menor es el área a la derecha del estadístico (`Pr(>F)`) mejor es el ajuste que hemos obtenido.

Tabla 6.2: Tabla de análisis de la varianza asociada al ajuste de regresión.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
$Regression$	1	$SS(Regression)$	$SS(Regression)/1$	$\frac{SS(Regression)/1}{SS(Residual)/(n-2)}$	
$Residuals$	$n-2$	$SS(Residual)$	$SS(Residual)/(n-2)$		

Ejemplo 6.2. *Veamos la tabla de análisis de la varianza.*

```
fit1 = aov(expression ~time, data = df0)
summary(fit1)
```

```
          Df Sum Sq Mean Sq F value Pr(>F)
time 1 39.60 39.60 50.73 3.84e-07
Residuals 22 17.18 0.78

time ***
Residuals
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Tabla 6.3: Tabla 6.2 observada para la sonda.

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
time	1	39.60	39.60	50.73	0.0000
Residuals	22	17.18	0.78		

6.2.3 Coeficiente de determinación R^2

Es una medida de la calidad del ajuste. Se define como

$$R^2 = \frac{\sum_{i=1}^n (\hat{y}_i - \bar{y})^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = \frac{SS(Regression)}{SS(Total)}. \tag{6.27}$$

Si tenemos en cuenta la ecuación (6.26) entonces

$$R^2 = 1 - \frac{\sum_{i=1}^n (y_i - \hat{y}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2} = 1 - \frac{SS(Residual)}{SS(Total)}. \tag{6.28}$$

A partir de (6.28) se ve fácilmente que un mejor ajuste de la recta de mínimos cuadrados, esto es, cuanto más cerca estén los puntos de la recta

mayor será el valor del coeficiente de determinación R^2. El coeficiente de determinación (solamente) en el caso de regresión lineal simple tiene una interpretación muy simple e interesante. Notemos que $\sum_{i=1}^n(\hat{y}_i - \bar{y})^2 = \hat{\beta}_1^2 \sum_{i=1}^n (x_i - \bar{x})^2$ y $\hat{\beta}_1 = S_{xy}/S_{xx}$. Por tanto,

$$R^2 = \frac{S_{xy}^2}{S_{xx}S_{yy}}, \tag{6.29}$$

que no es más que el cuadrado del coeficiente de correlación entre la predictora y la respuesta.[80][2]

[80] Cuando el modelo tenga más variables predictoras veremos que es en general el coeficiente de correlación entre las predicccciones y la variable respuesta.

```
fit0.s = summary(fit0)
fit0.s$r.squared
```

```
[1] 0.6974934
```

6.2.4 Modelo

En lo que hemos hecho hasta ahora nos hemos limitado a obtener una buena aproximación lineal de las respuesta a partir del predictor. No se puede ir mucho más allá sin un modelo probabilístico que nos permita valorar lo que estamos haciendo.

Definición 6.1 (Regresión lineal simple)**.**

1. $Y_i = \beta_0 + \beta_1 x_i + \epsilon_i$ *para* $i = 1, \ldots, n$.
2. $\epsilon_i \sim N(0, \sigma^2)$.
3. *Los errores* ϵ_i *son independientes.*

En definitiva estamos asumiendo que la distribución condicionada de Y_i al predictor x_i es normal con media $\beta_0 + \beta_1 x_i$ y varianza σ^2,

$$Y_i | x_i \sim N(\beta_0 + \beta_1 x_i, \sigma^2),$$

y que los distintos Y_i son independientes entre sí.

Es conveniente y útil considerar una representación matricial del modelo de regresión lineal simple. Tenemos el **vector de respuestas aleatorias**, $\boldsymbol{Y}$; la **matriz de modelo**, $\boldsymbol{X}$; el **vector de coeficientes**, $\boldsymbol{\beta}$ y el **vector de errores aleatorios**, $\boldsymbol{\epsilon}$ dados por

$$\boldsymbol{Y} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_n \end{bmatrix}; \quad \boldsymbol{X} = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}; \quad \boldsymbol{\beta} = \begin{bmatrix} \beta_0 \\ \beta_1 \end{bmatrix}; \quad \boldsymbol{\epsilon} = \begin{bmatrix} \epsilon_1 \\ \vdots \\ \epsilon_n \end{bmatrix}$$

Y el modelo de regresión lineal simple se puede formular de un modo más compacto como

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}. \tag{6.30}$$

Los distintos errores aleatorios son independientes y con distribución común $N(0, \sigma^2)$ entonces el vector de errores aleatorios sigue una distribución normal multivariante con vector de medias nulo y matriz de covarianzas $\sigma^2 \boldsymbol{I}_n$ (modest18-§ 12.5)

$$\boldsymbol{\epsilon} \sim N_n(\boldsymbol{0}_n, \sigma^2 \boldsymbol{I}_n). \tag{6.31}$$

[81]

[81] $\mathbf{0}_n$ es el vector con n ceros mientras que $\boldsymbol{I}_n$ es la matriz identidad $n \times n$.

Ejemplo 6.3. *En el ajuste propuesto en el ejemplo podemos obtener la matriz de modelo con*

```
head(model.matrix(fit0))
```

```
               (Intercept) time
GSM618324.CEL.gz 1 0
GSM618325.CEL.gz 1 0
GSM618326.CEL.gz 1 1
GSM618327.CEL.gz 1 1
GSM618328.CEL.gz 1 6
GSM618329.CEL.gz 1 6
```

6.2.5 Verosimilitud

¿Cuál es la función de verosimitud? Asumiendo el modelo formulado en § 6.2.4 la función de verosimilitud viene dada por

$$L(\beta_0, \beta_1, \sigma^2; y_1, \ldots, y_n) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2\sigma^2}(y_i - \beta_0 - \beta_1 x_i)^2} = \frac{1}{(2\pi)^{n/2}\sigma^n} e^{-\frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2} \quad (6.32)$$

Como sabemos es más habitual y práctico trabajar con el logaritmo natural de la verosimilitud o **logverosimilitud**. En este caso la función de logverosimilitud viene dada por

$$\ell(\beta_0, \beta_1, \sigma^2; y_1, \ldots, y_n) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \beta_0 - \beta_1 x_i)^2. \quad (6.33)$$

Por ser una transformación monótona los valores máximos para los parámetros son los mismos en la función de verosimilitud y en la de logverosimilitud. Estos estimadores son los estimadores máximo verosímiles. Si observamos la expresión de la función de logverosimilitud en (6.33) es equivalente maximizar esta función a minimizar la suma de cuadrados. De otro modo los estimadores máximo verosímiles de los coeficientes corresponden con los estimadores mínimo cuadráticos obtenidos previamente.

6.3 Análisis de la varianza con un factor fijo

Vamos a estudiar qué es el análisis de la varianza con un solo factor (fijo). Lo hacemos sin incluir demostraciones que veremos en lo que sigue. Estudiamos de un modo sencillo la situación en que tenemos una variable respuesta cuantitativa y un único predictor de carácter categórico (factor experimental).

[2]Recordemos que el coeficiente de correlación muestral para $(x_i, y_i) \in \mathbb{R}^2$ con $i = 1, \ldots, n$ se define como $r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$. Con la notación que estamos utilizando $r_{xy} = \frac{S_{xy}}{\sqrt{S_{xx}}\sqrt{S_{yy}}}$.

6.3.1 Análisis de la varianza de una vía

Con frecuencia tenemos una variable de interés Y y pretendemos estudiar su posible dependencia de un factor experimental. Por ejemplo, la variable respuesta puede cuantificar el resultado de un tratamiento médico y pretendemos comparar sus valores para distintos tratamientos. El experimentador tiene un **factor** de interés con distintos niveles y se pretende evaluar la dependencia de la variable respuesta de este factor experimental. Si el factor tiene dos niveles entonces podemos comparar las medias mediante un test de la t o t-test. Con más de dos niveles necesitamos otros procedimientos. Habitualmente, pero no siempre, el factor corresponderá a una variable de control fijada por el experimentador. En este tema nos ocupamos (de un modo muy simple) de lo que se conoce como **experimentos con un solo factor completamente aleatorizado**.

Ejemplo 6.4. *Como ejemplo a analizar en esta sección vamos a seguir con los datos* `tamidata2::gse25171`*, en particular, la sonda* `261892_at`*. Podemos ver en las variables fenotípicas que tenemos definida la variable* `time2` *en la que el tiempo (*`time`*) es discretizado en dos valores (*`Short` ↪ *y* `Medium`*). También tenemos la variable categórica que nos indica la presencia o no de fosfatos (*`Pi`*). Vamos a construir una variable categórica (*`time2Pi`*) que recoge las cuatro combinaciones posibles de las dos variables binarias.*

```
time2Pi = vector("list",ncol(gse25171))
for(i in seq_along(time2Pi))
   time2Pi[[i]] = paste0(pData(gse25171)[,"time2"][i],
                         pData(gse25171)[,"Pi"][i])
time2Pi = factor(unlist(time2Pi))
```

Podemos ver los distintos valores que puede tomar la variable `time2Pi`*.*

```
levels(time2Pi)
```

```
[1] "MediumControl" "MediumTreatment"
[3] "ShortControl" "ShortTreatment"
```

Construimos un `data.frame` `df1` *con la variable categórica que acabamos de construir.*

```
sel0 = which("261892_at"==fData(gse25171)[,"PROBEID"])
df1 = data.frame(time2Pi,expression=exprs(gse25171)[sel0,])
```

La variable respuesta es `expression` *y nuestra predictora es* `time2Pi`*.*

```
summary(df1[,"time2Pi"])
```

```
 MediumControl MediumTreatment ShortControl
             6 6 6
 ShortTreatment
             6
```

Tenemos los cuatro grupos equilibrados. Pretendemos evaluar si los valores de la variable respuesta dependen del tratamiento. Quizás, para empezar, no viene mal hacer un diagrama de cajas comparando los valores de la variable respuesta en los grupos definidos por la variable predictora. En la figura 6.6 aparece el diagrama de cajas.

```
p = ggplot(df1,aes(x=time2Pi,y=expression)) + geom_boxplot()
```

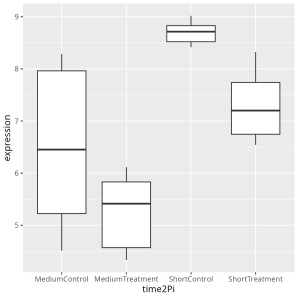


Figura 6.6: Diagrama de cajas mostrando la expresión en la sonda considerando el factor

6.3.2 Comparando grupos

Supongamos que tenemos I condiciones distintas y en cada una de ellas n_i muestras de modo que $\sum_{i=1}^{I} n_i = n$, el total de muestras de las que disponemos. Supongamos que Y_{ij} denota la respuesta aleatoria en la j-ésima muestra de la i-ésima condición. Suponemos que los valores Y_{ij} con $j = 1, \ldots, n_i$ son independientes y con la misma distribución. El modelo de análisis de la varianza de una vía es

$$Y_{ij} = \mu_i + \epsilon_{ij}, \tag{6.34}$$

donde se asume que $\epsilon_{ij} \sim N(0, \sigma^2)$ y son independientes entre si para los distintos grupos y dentro de cada grupo.

Una formulación alternativa y habitual del modelo (6.34) es

$$Y_{ij} = \beta_0 + \beta_i + \epsilon_{ij}, \tag{6.35}$$

donde estamos expresando la media μ_i en el grupo i-ésimo de observaciones como

$$\mu_i = E[Y_{ij}] = \beta_0 + \beta_i. \tag{6.36}$$

Es claro que en la formulación (6.36) tenemos I medias μ_i pero $I + 1$ parámetros β. En particular notemos que podemos sumar una cantidad δ a β_0 y restar esa misma cantidad a cada uno de los β_i con $i = 1, \ldots, I$ y las ecuaciones se mantienen. Esto es, tenemos un problema de identificabilidad de los parámetros. Por ello hemos de asumir una ecuación más. Lo habitual es considerar una categoría de referencia (por ejemplo pero no obligatoriamente la primera) y asumir que el parámetro es nulo. Por ejemplo, asumir que $\beta_1 = 0$.

Realmente estamos asumiendo en el modelo (6.36) que $Y_{ij} \sim N(\beta_0 + \beta_i, \sigma^2)$. La interpretación de los distintos parámetros es la siguiente:

1. β_0 es la media en el grupo de referencia.
2. β_i sería la diferencia de la media del grupo i respecto de la media del grupo de referencia.
3. ϵ_{ij} sería el error aleatorio de la observación j-ésima del grupo i-ésimo respecto de la media de la variable respuesta en el grupo, $\beta_0 + \beta_i$.

Se trata de evaluar si hay **diferencias entre grupos**. Bajo el modelo que acabamos de formular se traduce en la siguiente hipótesis nula,

$$H_0 : \beta_1 = \beta_2 = \ldots = \beta_I = 0,$$

frente a que algúno los β_i con $i \geq 2$ sea no nulo. Recordemos que asumimos $\beta_1 = 0$ por defecto.

¿Cómo podemos contrastar la hipótesis nula anterior? Si y_{ij} es la j-ésima muestra observada bajo la condición i ($i = 1, \ldots, I$ y $j = 1, \ldots, n_i$) entonces las medias muestrales para cada grupo serán

$$\bar{y}_{i\cdot} = \sum_{j=1}^{n_i} \frac{y_{ij}}{n_i},$$

y la media de todas las observaciones o media total será

$$\bar{y}_{\cdot\cdot} = \sum_{i=1}^{I} \sum_{j=1}^{n_i} \frac{y_{ij}}{n}.$$

Definimos la *suma de cuadrados intra* o **del error** como

$$SS(Within) = \sum_{i=1}^{I}\sum_{j=1}^{n_i}(y_{ij} - \bar{y}_{i\cdot})^2,$$

y la *suma de cuadrados entre* como

$$SS(Between) = \sum_{i=1}^{I} n_i(\bar{y}_{i\cdot} - \bar{y}_{\cdot\cdot})^2.$$

El estadístico para contrastar esta hipótesis nula es

$$F = \frac{SS(Between)/(I-1)}{SS(Within)/(n-I)}.$$

Estas sumas de cuadrados se suelen disponer en forma de tabla.

Tabla 6.4: Tabla de análisis de la varianza.

Source	SS	df	MS	F	p
Between	$SS(Between)$	$I-1$	$SS(Between)/(I-1)$	$F = \frac{SS(Between)/(I-1)}{SS(Within)/(n-I)}$	$P(>F)$
Within	$SS(Within)$	$n-I$	$SS(Within)/(n-I)$		
Total	$SS(Between) + SS(Within)$				

Bajo la hipótesis nula de que todas las medias son la misma (y puesto que asumimos una misma varianza) tendríamos una distribución común bajo todas las condiciones. Asumiendo la hipótesis nula el estadístico F se distribuye como un F con $I-1$ y $n-I$ grados de libertad,

$$F \sim F_{I-1,n-I}.$$

Es claro que, bajo la hipótesis alternativa, los valores de F tenderán a ser **grandes** o mayores que los esperables bajo la hipótesis nula. En resumen, la **región crítica** (donde rechazamos la hipótesis nula) será un intervalo de la forma $[c, +\infty)$. Si tomamos como valor c el valor observado tendremos el p-valor.

Ejemplo 6.5. *Consideramos como variable respuesta* `expression` *y como factor experimental (predictora) la variable* `time2Pi`.

```
fit2 = aov(expression ~time2Pi,data=df1)
summary(fit2)
```

```
          Df Sum Sq Mean Sq F value Pr(>F)
time2Pi 3 37.52 12.505 12.98 6.22e-05
Residuals 20 19.26 0.963

time2Pi ***
Residuals
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como vemos resulta significativa la diferencia entre medias con un nivel de significación de 0.05.

Ejemplo 6.6. *El modelo que acabamos de considerar con una categoría de referencia se puede ajustar del siguiente modo.*

```
fit3 = lm(expression ~time2Pi,data=df1)
```

Los coeficientes ajustados son

```
coef(fit3)
```

```
          (Intercept) time2PiMediumTreatment
            6.4975788 -1.2418791
   time2PiShortControl time2PiShortTreatment
            2.2010323 0.7990972
```

Veamos qué ofrece el resumen del modelo.

```
summary(fit3)
```

```
Call:
lm(formula = expression ~time2Pi, data = df1)

Residuals:
     Min 1Q Median 3Q Max
-1.98463 -0.62805 0.04225 0.54559 1.79155

Coefficients:
                     Estimate Std. Error
(Intercept) 6.4976 0.4007
time2PiMediumTreatment -1.2419 0.5666
time2PiShortControl 2.2010 0.5666
time2PiShortTreatment 0.7991 0.5666
                     t value Pr(>|t|)
(Intercept) 16.217 5.66e-13 ***
time2PiMediumTreatment -2.192 0.040407 *
time2PiShortControl 3.884 0.000922 ***
time2PiShortTreatment 1.410 0.173828
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9814 on 20 degrees of freedom
Multiple R^2: 0.6607, Adjusted R^2: 0.6098
F-statistic: 12.98 on 3 and 20 DF, p-value: 6.219e-05
```

Es distinto.

6.4 Mínimos cuadrados

Esta sección[3] considera la situación en que pretendemos predecir una variable respuesta continua y tenemos más de un predictor.

6.4.1 El problema

Disponemos de los datos $(\boldsymbol{x}_i, y_i)$ con $i = 1, \ldots, n$ siendo $\boldsymbol{y} = (y_1, \ldots, y_n)^T$, las respuestas observadas; $\boldsymbol{x}_i = (x_{i1}, \ldots, x_{ip})^T$, los predictores correspondientes a la i-ésima observación. La matriz de modelo que recoge los valores de los predictores:

$$\boldsymbol{X} = \begin{bmatrix} x_{11} & \ldots & x_{1p} \\ \vdots & \vdots & \vdots \\ x_{n1} & \ldots & x_{np} \end{bmatrix}$$

En las secciones anteriores hemos visto modelos (regresión lineal simple y análisis de la varianza de un factor fijo) en donde, para el vector

[3] Una versión extendida de esta sección la tenemos en [6, capítulo 2] y en modest18-[**modest18-chapter:MinimosCuadrados**].

aleatorio de respuestas $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$, su vector de medias $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T$ (con medias $\mu_i = E[Y_i|\boldsymbol{x}_i]$) verifica

$$\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta}.$$

¿Cómo estimamos $\boldsymbol{\beta}$?

6.4.2 Planteamiento

Se trata de estimar $\boldsymbol{\beta}$. Estamos relacionando el vector de medias con los predictores. No conocemos $\boldsymbol{\mu}$. La idea es sustituir la media μ_i por el valor observado y_i. ¿Podemos resolver el siguiente sistema?

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta}.$$

Este sistema no tiene solución. Es un sistema sobre determinado en el cual tenemos más ecuaciones que incógnitas. Sustituimos la idea de resolver el sistema por la de buscar una buena solución. Una opción clásica son los mínimos cuadrados en donde pretendemos que

$$\|\boldsymbol{y} - \hat{\boldsymbol{\mu}}\|^2$$

sea mínimo donde $\hat{\boldsymbol{\mu}} = \boldsymbol{X}\hat{\beta}$. Obviamente lo que estamos haciendo es sustituir el vector de medias desconocido por los valores observados.

$$\|\boldsymbol{y} - \hat{\boldsymbol{\mu}}\|^2 = \sum_{i=1}^{n}(y_i - \hat{\mu}_i)^2 = \sum_{i=1}^{n}(y_i - \sum_{j=1}^{p}\hat{\beta}_j x_{ij})^2 \tag{6.37}$$

Consideramos la función

$$S_t(\boldsymbol{\beta}) = \sum_{i=1}^{n}(y_i - \mu_i)^2 = \sum_{i=1}^{n}\left(y_i - \sum_{j=1}^{p}\beta_j x_{ij}\right)^2.$$

Los estimadores mínimo cuadráticos minimizan la función S_t. Podemos ver (modest18-4.5 que se ha de verificar la ecuación

$$\boldsymbol{X}^T\boldsymbol{y} = \boldsymbol{X}^T\boldsymbol{X}\hat{\boldsymbol{\beta}}, \tag{6.38}$$

lo que asumiendo rango completo de la matriz de modelo nos da como estimadores mínimo cuadráticos

$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}. \tag{6.39}$$

6.4.3 Predicciones y la matriz H

Tenemos que

$$\boldsymbol{\mu} = \boldsymbol{X}\boldsymbol{\beta},$$

por tanto, las medias estimadas las obtenemos con

$$\hat{\boldsymbol{\mu}} = \boldsymbol{X}\hat{\boldsymbol{\beta}} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}.$$

Si denotamos

$$\boldsymbol{H} = \boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T, \tag{6.40}$$

entonces

$$\hat{\boldsymbol{\mu}} = \boldsymbol{H}\boldsymbol{y}.$$

La matriz $\boldsymbol{H}$ recibe el nombre de matriz sombrero (hat matrix) o matriz de influencia.

Ejemplo 6.7. *Analizamos los datos* `df0` *donde la variable respuesta es* `expression` *las predictoras son* `time` *y* `Pi`*. Realizamos el ajuste.*

```
fit4 = lm(expression~ time + Pi,data=df0)
```

Podemos ver que en el `data.frame` *solamente tenemos las variables que estamos utilizando. Por ello, es suficiente especificar quién es la respuesta e indicar con un punto que utilizamos las demás como predictoras. Ambos códigos son equivalentes.*

```
fit4 = lm(expression~.,data=df0)
```

La matriz modelo es (mostramos primeras columnas).

```
head(model.matrix(fit4),n=3)
```

```
                (Intercept) time PiTreatment
GSM618324.CEL.gz 1 0 1
GSM618325.CEL.gz 1 0 0
GSM618326.CEL.gz 1 1 1
```

Alternativamente podemos acceder a `model`*.*

```
fit4$model
```

¿Cómo podemos obtener la matriz $\boldsymbol{H}$*?*

```
X = model.matrix(fit4) ## Usamos stats::model.matrix
H = X %*% solve(t(X) %*% X) %*% t(X) ## Cálculo explícito
```

donde `%*%` *es el producto matricial,* `base::solve()` *nos devuelve la inversa de la matriz y* `base::t()` *es la matriz traspuesta. Los residuos los obtenemos con*

```
head(resid(fit4))
```

```
GSM618324.CEL.gz GSM618325.CEL.gz
     1.01552765 0.21676116
GSM618326.CEL.gz GSM618327.CEL.gz
    -0.32266386 -0.07784405
GSM618328.CEL.gz GSM618329.CEL.gz
    -0.59106601 0.31824015
```

6.4.4 Estimando la variación

Hasta ahora no nos hemos ocupado de la varianza del error. Asumimos errores independientes y equidistribuidos con una varianza constante. Abreviadamente, el modelo lineal que consideramos incorporando el error aleatorio es

$$\boldsymbol{Y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon},$$

con $var(\boldsymbol{\epsilon}) = \sigma^2\boldsymbol{I}$. En esta sección proponemos el estimador (insesgado) de σ^2 habitualmente utilizado.

Estimando la varianza

El resultado fundamental es el siguiente.

Proposición 6.1.

$$E\left[\frac{\boldsymbol{Y}^T(\boldsymbol{I}-\boldsymbol{H})\boldsymbol{Y}}{n-p}\right] = E\sum_{i=1}^{n}\frac{(Y_i-\hat{\mu}_i)^2}{n-p} = \sigma^2. \tag{6.41}$$

Por tanto, un estimador insesgado de la varianza del error aleatorio sería

$$S^2 = \sum_{i=1}^{n} \frac{(Y_i - \hat{\mu}_i)^2}{n-p}, \tag{6.42}$$

esto es, la suma de los cuadrados de los residuos dividida por el número de observaciones menos el número de parámetros. S^2 recibe el nombre de **cuadrado medio del error** o **cuadrado medio residual**. Su raiz cuadrada, S, es el **error estándar residual**. Por la proposición 6.1, S^2 es un estimador insesgado de σ^2 mientras que S no es un estimador insesgado de σ.

Ejemplo 6.8. *Si consideramos el modelo lineal solamente con un término constante:* $\boldsymbol{X} = \mathbf{1}_n$, $\hat{\mu}_i = \bar{Y}$ *y el estimador de* (6.42) *sería*

$$S^2 = \sum_{i=1}^{n} \frac{(Y_i - \bar{Y})^2}{n-1},$$

que es insesgado.

Ejemplo 6.9. *El error estándar residual lo tenemos con*

```
summary(fit4)$sigma
```

```
[1] 0.5644856
```

Sumas de cuadrados

Definición 6.2. *La* ***suma de cuadrados total*** *(o* ***suma de cuadrados corregida****) se define como*

$$SS(Total) = \sum_{i=1}^{n} y_i^2 - n\bar{y}^2 = \sum_{i=1}^{n} (y_i - \bar{y})^2.$$

La ***suma de cuadrados de la regresión*** *o* ***suma de cuadrados del modelo*** *es*

$$SS(Regression) = \sum_{i=1}^{n} (\hat{\mu}_i - \bar{y})^2.$$

La ***suma de cuadrados residual*** *o* ***suma de cuadrados del error*** *es*

$$SS(Residual) = \sum_{i=1}^{n} (y_i - \hat{\mu}_i)^2.$$

Proposición 6.2.

$$SS(Total) = SS(Regression) + SS(Residual). \tag{6.43}$$

En ocasiones a $\sum_{i=1}^{n} y_i^2$ se le llama **suma de cuadrados total** pero es más frecuente la notación que utilizamos aquí.

Coeficiente de determinación y correlación múltiple

Obviamente un ajuste es tanto mejor cuando menor es la suma de cuadrados residual y mayor la suma de cuadrados de la regresión. Una medida razonable de cuantificar la bondad del ajuste es considerar el siguiente cociente conocido como **coeficiente de determinación**. [82]

[82] Lo veremos etiquetado con **R-squared**.

$$R^2 = \frac{SS(Regression)}{SS(Total)} = \frac{SS(Total) - SS(Residual)}{SS(Total)} = \frac{\sum_{i=1}^n (y_i - \bar{y})^2 - \sum_{i=1}^n (y_i - \hat{\mu}_i)^2}{\sum_{i=1}^n (y_i - \bar{y})^2}. \quad (6.44)$$

También es razonable cuantificar la calidad del ajuste viendo si los valores que intentamos predecir están correlados con las predicciones que hemos obtenido, esto es, considerar el coeficiente de correlación muestral

$$corr(\boldsymbol{y}, \hat{\boldsymbol{\mu}}) = \frac{\sum_{i=1}^n (y_i - \bar{y})(\hat{\mu}_i - \bar{\hat{\mu}})}{\sqrt{\sum_{i=1}^n (y_i - \bar{y})^2 \sum_{i=1}^n (\hat{\mu}_i - \bar{\hat{\mu}})^2}}. \quad (6.45)$$

Un valor extremo de la correlación muestral indicaría una clara asociación y por lo tanto un buen ajuste.

Proposición 6.3.

$$corr(\boldsymbol{y}, \hat{\boldsymbol{\mu}}) = +\sqrt{R^2}.$$

La raiz cuadrada positiva de R^2 recibe el nombre de **correlación múltiple**. Tenemos $0 \leq R \leq 1$. Con una sola variable predictora entonces R es el coeficiente de correlación entre esta variable predictora y la variable respuesta como vimos en § 6.2.3.

Por la propia definición del coeficiente de determinación más variables en el modelo supone un incremento de R^2. Una cuantificación muy similar pero que no necesariamente verifica esto es la R^2 ajustada definida como

$$R^2_{adjusted} = 1 - \frac{SS(Residual)/(n-p)}{SS(Total)/(n-1)} = 1 - \frac{n-1}{n-p}(1 - R^2). \quad (6.46)$$

No hay grandes diferencias en su uso con la R^2.

6.5 Modelos lineales normales

En esta sección[4] se asume un modelo estocástico para el vector de respuestas, $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^T$. Asumiremos que sigue una distribución normal multivariante, $\boldsymbol{Y} \sim N_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, con vector de medias $\boldsymbol{\mu}$ y matriz de covarianzas $\boldsymbol{\Sigma}$.

6.5.1 Modelo y verosimilitud

El modelo lineal normal asume que $\boldsymbol{Y} \sim N_n(\boldsymbol{\mu}, \sigma^2 \boldsymbol{I}_n)$, siendo $\mu_i = \boldsymbol{x}_i^T \boldsymbol{\beta}$ de donde la función de verosimilitud es

$$L(\boldsymbol{\beta}, \sigma^2) = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma^n} \exp\{-\frac{1}{\sigma^2}(\mathbf{y} - \boldsymbol{\mu})^T(\mathbf{y} - \boldsymbol{\mu})\} = \frac{1}{(2\pi)^{\frac{n}{2}} \sigma^n} \exp\{-\frac{1}{\sigma^2}\sum_{i=1}^n (y_i - \boldsymbol{x}_i^T \boldsymbol{\beta})^2\}. \quad (6.47)$$

[4] Una versión extendida de esta sección la tenemos en [6, capítulo 3] y en modest18-§ 5.

La función de logverosimilitud es

$$\ell(\boldsymbol{\beta}, \sigma^2) = -\frac{n}{2}\ln(2\pi) - n\ln(\sigma) - \frac{1}{\sigma^2}\sum_{i=1}^{n}(y_i - \boldsymbol{x}_i^T\boldsymbol{\beta})^2. \tag{6.48}$$

En § 6.4 hemos visto que $\hat{\boldsymbol{\beta}} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{Y}$, $\hat{\boldsymbol{\mu}} = \boldsymbol{H}\boldsymbol{Y}$ y $\hat{\boldsymbol{e}} = (\boldsymbol{I} - \boldsymbol{H})\boldsymbol{Y}$. En resumen, los estimadores mínimo cuadráticos de los coeficientes, las estimaciones de las medias o predicciones y los residuos son transformaciones lineales del vector (aleatorio) de observaciones. Y por ello todos ellos tienen distribución normal multivariante. Sin embargo, sus matrices de varianzas no son proporcionales a la matriz identidad. Por tanto, los distintos estimadores de los coeficientes no son independientes entre sí. Lo mismo podemos decir de las estimadores de las medias. Los estimadores de las distintas medias no son independientes. Tampoco son independientes entre sí los distintos residuos.

6.5.2 Constrastes para modelos lineales normales

Anova de una vía

El modelo[5] es

$$Y_{ij} = \beta_0 + \beta_i + \epsilon_{ij}$$

siendo $\epsilon_{ij} \sim N(0, \sigma^2)$ e independientes entre sí. Asumimos la restricción $\beta_1 = 0$. Pretendemos contrastar: $H_0 : \mu_1 = \cdots = \mu_I$ con ($\mu_i = \beta_0 + \beta_i = E[Y_{ij}]$) frente a la alternativa de que al menos dos medias sean diferentes. Este contraste sería equivalente a $H_0 : \beta_1 = \cdots = \beta_I = 0$.

Tenemos pues que el estadístico para contrastar la hipótesis nula sigue la siguiente distribución nula

$$F = \frac{\sum_{i=1}^{I} n_i(\bar{Y}_{i\cdot} - \bar{Y}_{\cdot\cdot})^2/(I-1)}{\sum_{i=1}^{I}\sum_{j=1}^{n_i}(Y_{ij} - \bar{Y}_{i\cdot})^2/(n-I)} \sim F_{I-1,n-I}.$$

Modelos anidados

Tenemos dos modelos, M_0 y M_1, de modo que M_0 está anidado en M_1. Si el modelo M_0 es cierto tenemos que

$$F = \frac{(SS(Residual)_{M_0} - SS(Residual)_{M_1})/(r_1 - r_0)}{(SS(Residual)_{M_1})/(n - r_1)} \sim F_{r_1 - r_0, n - r_1} \tag{6.49}$$

Claramente rechazamos la hipótesis nula de que el modelo M_0 es asumible corresponde con valores grandes del estadístico en ecuación 6.49.

Consideremos un caso particular de lo considerado en la sección anterior. En concreto que todos los coeficientes correspondientes a todas las variables predictoras son iguales a cero. Por tanto, el modelo M_0 corresponde al modelo solamente con la constante. El modelo M_1 tiene todas las variables predictoras que estamos considerando. Un estudio más detallado lo tenemos en modest18-§ 5.4.2.

Contrastes para los coeficientes

Supongamos que nos planteamos la hipótesis nula de que $H_0 : \beta_j = 0$ vs $H_1 : \beta_j \neq 0$, es decir, que un coeficiente individual es nulo. Tenemos

[5] Un estudio más detallado en modest18-§ 5.4.1

dos modelos anidados, uno sería el modelo con todos los predictores y el otro el modelo en que eliminamos el j-ésimo predictor. Se verifica (lo damos sin prueba) que

$$F = \frac{(SS(Residual)_{M_0} - SS(Residual)_{M_1})/1}{SS(Residual)_{M_1}/(n-p)} = \frac{\hat{\beta}_j^2}{(SE_j)^2}, \qquad (6.50)$$

siendo SE_j el error estándar de $\hat{\beta}_j$. Recordemos que $var(\boldsymbol{\beta}) = \sigma^2(\boldsymbol{X}^T\boldsymbol{X})^{-1}$. Por tanto, $SE_j = s\sqrt{u_{jj}}$ siendo u_{jj} el elemento en posición (j,j) de la matriz $(\boldsymbol{X}^T\boldsymbol{X})^{-1}$. El estadístico en ecuación 6.50 sigue una distribución $F_{1,n-p}$. En la ecuación 6.50 tenemos el cuadrado de $\frac{\hat{\beta}_j}{SE_j}$ que sigue por lo tanto una distribución t de Student con $n-p$ grados de libertad,

$$\frac{\hat{\beta}_j}{SE_j} \sim t_{n-p}.$$

Comparando modelos

Volvemos a ajustar el modelo `fit4`.

```
fit4 = lm(expression~ time + Pi,data=df0)
```

Los contrastes para los coeficientes así como el contraste global de si todos los predictores podemos considerar que son nulos lo tenemos con un **`summary`** del ajuste que nos devuelve la función **`lm`**.

```
summary(fit4)
```

```
Call:
lm(formula = expression ~time + Pi, data = df0)

Residuals:
   Min 1Q Median 3Q Max
-0.9399 -0.4025 0.1085 0.2609 1.1446

Coefficients:
           Estimate Std. Error t value Pr(>|t|)
(Intercept) 8.62939 0.18738 46.053 < 2e-16
time -0.13307 0.01194 -11.148 2.79e-10
PiTreatment -1.32191 0.23045 -5.736 1.08e-05

(Intercept) ***
time ***
PiTreatment ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.5645 on 21 degrees of freedom
Multiple R^2: 0.8821, Adjusted R^2: 0.8709
F-statistic: 78.6 on 2 and 21 DF, p-value: 1.774e-10
```

En la salida anterior podemos ver el test global en que todas las variables predictoras son no significativas, esto es, que todas los coeficientes de todas las predictoras son nulos. Este test aparece en la última línea del resumen. Para cada variable podemos ver junto a su estimación, su error estándar, el correspondiente cociente con distribución t de Student y el p-valor.

6.5.3 Intervalos de confianza y de predicción

En esta sección nos ocupamos de obtener intervalos de confianza para los coeficientes, la media de la respuesta para unos predictores dados y el propio valor aleatorio de la respuesta.

Proposición 6.4. *Se tiene que*

$$\frac{n-p}{\sigma^2}S^2 = \frac{1}{\sigma^2}\sum_{i=1}^{n} e_i^2 \sim \chi^2_{n-p}.$$

Intervalo para un coeficiente

Consideramos la hipótesis nula $H_0 : \beta_j = \beta_{j0}$ frente a que $H_1 : \beta_j \neq \beta_{j0}$ y el estadístico

$$t = \frac{\hat{\beta}_j - \beta_{j0}}{SE_j}. \tag{6.51}$$

El intervalo de confianza al nivel $1-\alpha$ vendría dado por los valores β_{j0} para los cuales no rechazamos la hipótesis nula $H_0 : \beta_j = \beta_{j0}$ y vendría dado por

$$\hat{\beta}_j \pm t_{n-p,1-\alpha/2} SE_j.$$

Ejemplo 6.10. *Veamos cómo obtenerlos utilizando la función genérica* `confint` *aplicada a un objeto de clase* `lm` *Los intervalos de confianza, con nivel de confianza* $1-\alpha = .95$*, vistos en esta sección los obtenemos con*

```
confint(fit4,conf.level=0.95)
```

```
               2.5 % 97.5 %
(Intercept) 8.2397130 9.0190666
time -0.1578931 -0.1082475
PiTreatment -1.8011547 -0.8426596
```

Intervalo de confianza para la media

Supongamos que tomamos unos predictores dados $\boldsymbol{x}_0$ (vector columna) y queremos el intervalo de confianza para la media condicionada $\mu_0 = E[Y|\boldsymbol{x}_0] = \boldsymbol{x}_0^T\boldsymbol{\beta}$. Otra vez lo construimos a partir de un t-test. La predicción correspondiente a los nuevos predictores sería $\hat{\mu}_0 = \boldsymbol{x}_0^T\hat{\boldsymbol{\beta}}$. El intervalo de confianza para la media condicionada $\boldsymbol{x}_0^T\boldsymbol{\beta}$ al nivel $1-\alpha$ sería

$$\boldsymbol{x}_0^T\hat{\boldsymbol{\beta}} \pm t_{n-p,1-\alpha/2}\, s\sqrt{\boldsymbol{x}_0^T(\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{x}_0}.$$

Intervalo de predicción

Consideramos unos predictores $\boldsymbol{x}_0$ y pretendemos construir un intervalo que contenga a la variable Y condicionada a los predictores $\boldsymbol{x}_0$ con una confianza dada. En resumen, un intervalo de predicción para una observación futura. Obviamente el intervalo será mayor. Es más complejo estimar la observación que estimar la media de la observación de lo que nos hemos ocupado esencialmente hasta ahora. Nuestro modelo asumido es

$$Y = \boldsymbol{x}_0^T\boldsymbol{\beta} + \epsilon \; con \; \epsilon \sim N(0, \sigma^2).$$

Ese valor aleatorio (futuro) tendrá como predicción $\hat{\mu}_0 = \boldsymbol{x}_0^T\hat{\boldsymbol{\beta}}$, esto es, nuestra predicción es el valor ajustado para la media. Lo que no será igual

es el intervalo que lo contenga. Tendremos un residuo (aleatorio) $e = Y - \hat{\mu}$ y se satisface

$$Y = \boldsymbol{x}_0^T \hat{\boldsymbol{\beta}} + e.$$

Tenemos como intervalo de predicción para la futura observación Y con predictores $\boldsymbol{x}_0$ el siguiente

$$\boldsymbol{x}_0^T \hat{\boldsymbol{\beta}} \pm t_{n-p,1-\alpha/2} s \sqrt{1 + \boldsymbol{x}_0^T (\boldsymbol{X}^T \boldsymbol{X})^{-1} \boldsymbol{x}_0}.$$

Ejemplos de intervalos para media y observación

Ejemplo 6.11. *Con los datos* `df0` *consideramos cómo obtener las predicciones y los intervalos de confianza para las medias y para las predicciones. Utilizamos los propios datos que se han utilizado para ajustar el modelo. Con la función genérica* `predict` *obtenemos las predicciones así como los intervalos de confianza para las medias de las predicciones (*`predict` *con la opción* `interval="confidence"`*) y los intervalos de confianza para las observaciones (*`predict` *con la opción* `interval"="prediction`*).*

Primero obtengamos los valores ajustados o predicciones.

```
head(predict(fit4))
```

```
GSM618324.CEL.gz GSM618325.CEL.gz
       7.307483 8.629390
GSM618326.CEL.gz GSM618327.CEL.gz
       7.174412 8.496319
GSM618328.CEL.gz GSM618329.CEL.gz
       6.509061 7.830968
```

Los intervalos de confianza para la media se obtienen con

```
head(predict(fit4,interval = "confidence"))
```

```
                   fit lwr upr
GSM618324.CEL.gz 7.307483 6.917806 7.697159
GSM618325.CEL.gz 8.629390 8.239713 9.019067
GSM618326.CEL.gz 7.174412 6.796373 7.552451
GSM618327.CEL.gz 8.496319 8.118280 8.874358
GSM618328.CEL.gz 6.509061 6.167409 6.850713
GSM618329.CEL.gz 7.830968 7.489316 8.172620
```

Los intervalos de confianza para las observaciones (en este caso sobre los propios datos) se obtienen con

```
head(predict(fit4,interval = "prediction"))
```

```
Warning in predict.lm(fit4, interval = "prediction"): predictions on
    ↪ current data refer to _future_ responses
```

```
                   fit lwr upr
GSM618324.CEL.gz 7.307483 6.070584 8.544381
GSM618325.CEL.gz 8.629390 7.392491 9.866288
GSM618326.CEL.gz 7.174412 5.941131 8.407694
GSM618327.CEL.gz 8.496319 7.263038 9.729601
GSM618328.CEL.gz 6.509061 5.286442 7.731679
GSM618329.CEL.gz 7.830968 6.608350 9.053586
```

Es interesante el comentario. Nos indica que se refieren a ***futuras observaciones*** *que pudieran tener los mismos predictores que estamos usando.*

6.6 Muchos modelos

Hemos elegido trabajar con una variable respuesta elegida de un modo arbitrario. En los ejemplos considerados la variable respuesta es una sonda. Tenemos interés en todas las sondas. Cada una de ellas nos proporciona una variable respuesta distinta. Esto es tenemos muchos modelos para ajustar y para cada uno de ellos contrastes a realizar. Es claro que los predictores serán variables fenotípicas compartidas por todas las respuestas. Por tanto los distintos modelos comparten los predictores y difieren en la variable respuesta. Veamos algunas opciones para hacerlo.

GSEAlm

Podemos utilizar el paquete [110].

```
pacman::p_load(GSEAlm)
data(gse25171,package="tamidata2")
fits1 = GSEAlm::lmPerGene(gse25171,~ time + Pi)
```

La matriz de modelo común a todos los ajustes la tenemos con

```
head(fits1$x)
```

```
              (Intercept) time PiTreatment
GSM618324.CEL.gz 1 0 1
GSM618325.CEL.gz 1 0 0
GSM618326.CEL.gz 1 1 1
GSM618327.CEL.gz 1 1 0
GSM618328.CEL.gz 1 6 1
GSM618329.CEL.gz 1 6 0
```

La matriz de proyección sobre el espacio modelo o matriz hat $\boldsymbol{H}$ con

```
fits1$Hmat
```

Cada uno de los ajustes correspondiendo con cada una de las filas de la matriz de expresión nos proporciona un vector de coeficientes que podemos obtener con

```
fits1$coefficients
```

Cada columna de la matriz anterior corresponde con cada fila de la matriz de expresión. Por ejemplo para la primera sonda tenemos los coeficientes.

```
fits1$coefficients[,1]
```

```
 (Intercept) time PiTreatment
 5.086290213 -0.004315513 -0.116650336
```

¿Cómo podemos obtener los p-valores de si el coeficiente asociado al tiempo es nulo o bien si el coeficiente asociado a `Pi` es nulo? Notemos que los estadísticos que nos dan los coeficientes divididos por sus errores estándar los tenemos en `fits1$tstat`. Por ejemplo, para la primera sonda lo tenemos en la primera columna.

```
fits1$tstat[,1]
```

```
(Intercept) time PiTreatment
  77.915652 -1.037790 -1.452962
```

Pero sabemos que sigue una distribución que es una t de Student con n_p grados de libertad siendo p el número de predictores y n el número de datos (columnas de la matriz de expresión). Por tanto los p-valores los podemos calcular utilizando la función de distribución acumulada de la correspondiente t de Student. En nuestro caso los correspondientes p-valores que corresponden al test de coeficiente nulo serían

```
ncol(gse25171) # n
```

```
Samples
    24
```

```
length(fits1$coefficients[,1]) # p
```

```
[1] 3
```

```
nu = ncol(gse25171) - length(fits1$coefficients[,1])
1-pt(fits1$tstat[,1],df=nu)
```

```
(Intercept) time PiTreatment
  0.0000000 0.8444147 0.9194926
```

Si queremos hacerlo para todos los coeficientes de todos los ajustes podemos usar el siguiente código en donde cada columna nos corresponde con un ajuste.

```
p.values = 1-pt(fits1$tstat,df=nu)
```

limma

Podemos utilizar para ajustar todos los modelos el paquete [128, limma].

```
design = model.matrix(~ pData(gse25171)[,"time"] + pData(gse25171)[,"Pi"])
fits2 = limma::lmFit(gse25171,design)
```

Los coeficientes los tenemos con

```
head(fits2$coefficients,n=2)
```

```
          (Intercept) pData(gse25171)[, "time"]
244919_at 5.086290 -0.004315513
244920_s_at 8.371483 -0.004977996
          pData(gse25171)[, "Pi"]Treatment
244919_at -0.11665034
244920_s_at -0.08573999
```

En este caso cada fila corresponde con un ajuste de modo que los coeficientes del ajuste para la primera sonda son

```
fits2$coefficients[1,]
```

```
                  (Intercept)
                  5.086290213
      pData(gse25171)[, "time"]
                 -0.004315513
pData(gse25171)[, "Pi"]Treatment
                 -0.116650336
```

Más detalles de esta opción las podemos ver en § 9.2.2.

6.7 rowFtests

En todo lo anterior hemos considerado la situación más habitual. Tenemos las muestras clasificadas en dos grupos o condiciones y pretendemos determinar aquellos genes que se expresan de un modo distinto entre ambas condiciones, que se expresan diferencialmente. Sin embargo, podemos tener más de dos grupos y pretender realizar un análisis similar. Esto nos conduce al análisis de la varianza de una vía. En modest18-§ 3 tenemos una presentación del modelo que utilizamos. No diría que es una opción muy recomendable en este tipo de datos. Quizás comparaciones dos a dos son más interpretables que no una interpretación a nivel de todas las medias que comparamos.

Ejemplo 6.12. *Lo ilustramos la expresión bajo diferentes condiciones de un gen de los datos* `tamidata::gse20986`.

```
data(gse20986,package="tamidata")
```

Seleccionamos un gen cualquiera.

```
y = exprs(gse20986)[678,]
```

Realizamos un análisis de la varianza.

```
summary(aov(y ~pData(gse20986)[,"tissue"]))
```

```
                          Df Sum Sq Mean Sq
pData(gse20986)[, "tissue"] 3 0.005309 0.001770
Residuals 8 0.020212 0.002527
                          F value Pr(>F)
pData(gse20986)[, "tissue"] 0.7 0.578
Residuals
```

De acuerdo con el p-valor observado no rechazaríamos la hipótesis nula a ninguno de los niveles de significación habituales (0.05 *ó* 0.01*).*

En el siguiente ejemplo analizamos, buscando diferencias entre cualquier par de grupos (tejidos en el ejemplo).

Ejemplo 6.13 (GSE20986)**.** *Vamos a realizar un análisis de expresión diferencial de los datos gse20986. Cargamos paquetes necesarios*

```
pacman::p_load(multtest,genefilter)
```

Tenemos cuatro grupos. Tenemos interés en la comparación entre todos los grupos (tejidos) al mismo tiempo. En definitiva buscamos aquellos genes que muestran una expresión diferencial entre al menos dos de los grupos. Un posible planteamiento es el modelo de análisis de la varianza y la comparación entre los grupos se reformula como una comparación entre valores medios.[6] *Esto supone una comparación simultánea de las medias de los cuatro grupos y esto se realiza con un análisis de la varianza. Calculamos los p-valores con la función* `genefilter::rowFtests()`.

```
gse20986.aov =
    rowFtests(gse20986, pData(gse20986)[,"tissue"])
```

Por ejemplo, vemos los dos primeros valores.

```
head(gse20986.aov,n=2)
```

```
         statistic p.value
1007_s_at 9.782456 0.004715281
1053_at 14.144595 0.001455658
```

[6] No es la única opción. Podríamos usar una opción no paramétrica.

Capítulo 7

Modelos lineales generalizados

En § 6 se ha asumido una distribución normal para la respuesta. Podemos llamar a esto la **componente aleatoria** del modelo. Para ser exactos se ha asumido que la distribución condicionada de Y_i a los predictores (variables fenotípicas en el contexto ómico), $\boldsymbol{x}_i$, asumiendo que $Y_i|\boldsymbol{x}_i \sim N(\boldsymbol{x}_i^T\boldsymbol{\beta}, \sigma^2)$. Estamos **eligiendo** las variables predictoras y **asumiendo** que la dependencia de la variable respuesta respecto de estas variables predictoras es a través de la media de la variable,

$$\mu_i = E[Y_i|\boldsymbol{x}_i] = \boldsymbol{x}_i^T\boldsymbol{\beta}.$$

Es la **componente sistemática** del modelo. Estamos **identificando** la media condicionada de la respuesta con la combinación lineal de los predictores. Si denotamos por g la función identidad, $g(\mu_i) = \mu_i$, entonces **enlazamos** (transformamos) la media μ_i con la componente sistemática mediante la función (**de enlace**) identidad.

En este tema estudiamos los **modelos lineales generalizados** que abreviaremos en **GLM** (generalized lineal model). [83] En esencia, la idea es poder trabajar con otras distribuciones de probabilidad para la variable respuesta sin limitarnos a asumir una distribución normal. En concreto generalizaremos a lo que se conoce como **familia de dispersión exponencial**. Esto incluirá distribuciones discretas como la binomial, Poisson o binomial negativa (asumiendo un parámetro de dispersión conocido). Con la distribución binomial modelizamos respuestas binarias en donde, dados unos predictores, observamos un éxito o fracaso (presencia o no de un atributo). Con las distribuciones Poisson o binomial negativa (con la restricción indicada) podremos modelizar datos de conteo (como números de lecturas alineadas sobre un gen en RNA-Seq). También podemos asumir distribuciones continuas como la gamma (también asumiendo un valor conocido para la dispersión) que nos permite modelizar datos no negativos con distribuciones claramente no normales.

[83] Sin duda, un nombre mal elegido por Wedderburn y Nelder porque se confunde con modelo lineal general que consiste en admitir una matriz de covarianzas genérica en lugar de un múltiplo de la matriz identidad. A estas alturas no parece tener remedio. Cosas peores se ven.

La componente sistemática la mantendremos tal cual, esto es, las variables predictoras intervienen conjuntamente mediante una combinación lineal de las mismas.

La función de enlace que nos lleva la media de la variable respuesta a la componente sistemática vamos a generalizarla de modo que no nos limitamos a la función identidad.

7.1 Componentes de un modelo lineal generalizado

Empezamos con una presentación formal y genérica de estos modelos. Un modelo lineal generalizado (GLM de un modo abreviado) consta de las siguientes componentes:

Componente aleatoria Identifica la variable respuesta Y y su distribución de probabilidad.

Componente sistemática Especifica las variables explicativas (independientes, predictoras) utilizadas en la función predictora lineal.

[84] Link function.

Función de enlace [84] Especifica la función de EY que la expresa como una combinación lineal de las variables predictoras.

En lo que sigue vamos a ver qué es cada una de estas componentes y estudiaremos los ejemplos más importantes desde el punto de vista de las aplicaciones.

La componente aleatoria de un GLM consiste de una variable aleatoria Y con observaciones independientes $(Y_1, \ldots, Y_n)$. Suponemos la distribución de Y es de la *familia de dispersión exponencial* cuya forma genérica sería

$$f(y_i; \theta_i, \phi) = e^{\frac{y_i\theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi)}. \tag{7.1}$$

En esta función de densidad el parámetro θ_i recibe el nombre de **parámetro natural** y ϕ es el **parámetro de dispersión**. Un caso de particular interés ocurre cuando $a(\phi) = 1$ y $c(y_i, \phi) = c(y_i)$. Esta subfamilia recibe el nombre de **familia exponencial natural** con

$$f(y_i; \theta_i) = h(y_i) \exp\left[y_i\theta_i - b(\theta_i)\right]. \tag{7.2}$$

La función $a(\phi)$ puede ser cualquiera aunque por razones prácticas que veremos después suele adoptar la forma

$$a(\phi) = \frac{\phi}{\omega}, \tag{7.3}$$

siendo ω un valor conocido y ϕ desconocido. El parámetro ϕ[85] es constante y no depende de las distintas observaciones. El valor ω es un peso **conocido** que sí dependerá de la observación.

[85] No es muy común hoy en día pero en [96] y otros artículos a veces se denota el parámetro de dispersión como σ^2. Utilizaremos ϕ en este manual.

Denotamos $\ell_i = \ln f(y_i; \theta_i, \phi)$ la logverosimilitud para la observación y_i. La logverosimilitud total es $l = \sum_{i=1}^n \ell_i$. Si suponemos que estamos en la familia de dispersión exponencial tendremos

$$\ell_i = [y_i\theta_i - b(\theta_i)]/a(\phi) + c(y_i, \phi).$$

Ejemplos

Veamos algunos ejemplos importantes de distribuciones que pertenecen a la familia de dispersión exponencial.

Ejemplo 7.1 (Distribución binomial)**.** *Suponemos que tenemos n_i pruebas Bernoulli que comparten unos mismos predictores $\boldsymbol{x}_i$. Vamos a considerar como respuesta aleatoria Y_i, en lugar del número de éxitos (como es*

habitual), la proporción de éxitos. Esto significa que el número total de éxitos vendrá dado por n_iY_i. Si denotamos por π_i la probabilidad de éxito común a las n_i pruebas Bernoulli entonces $n_iY_i \sim Bi(n_i, \pi_i)$ con $EY_i = \pi_i$. Consideramos

$$\theta_i = \ln \frac{\pi_i}{1-\pi_i}$$

esto es, definimos θ_i como el logit de la probabilidad de éxito. Fácilmente se comprueba que la transformación inversa es

$$\pi_i = \frac{e^{\theta_i}}{1+e^{\theta_i}},$$

y que $\ln(1-\pi_i) = -\ln(1+e^{\theta_i}$. Podemos expresar la función de probabilidad de la proporción muestral como

$$f(y_i;\pi_i,n_i) = \binom{n_i}{n_iy_i}\pi_i^{n_iy_i}(1-\pi_i)^{n_i-n_iy_i} = \exp\left[\frac{y_i\theta_i - \ln[1+\exp(\theta_i)]}{1/n_i} + \ln\binom{n_i}{n_iy_i}\right], \quad (7.4)$$

siendo $b(\theta_i) = \ln[1+\exp(\theta_i)]$, $a(\phi) = 1/n_i$ y $c(y_i,\phi) = \ln\binom{n_i}{n_iy_i}$. El parámetro natural es $\theta_i = \ln\frac{\pi_i}{1-\pi_i}$, el logit de π_i. Notemos que $a(\phi) = 1/n_i$ es conocido y tiene la forma de un parámetro ϕ (en este caso la unidad) dividida por un cantidad (un peso) conocido (en este caso el número de observaciones que comparten los predictores $\boldsymbol{x}_i$).

Ejemplo 7.2 (Distribución Poisson)**.** *La función de densidad de la distribución Poisson viene dada por*

$$f(y_i;\mu_i) = \frac{e^{-\mu_i}\mu_i^{y_i}}{y_i!} = \exp y_i \ln\mu_i - \mu_i - \ln(y_i!). \quad (7.5)$$

Si tomamos $\theta_i = \ln\mu_i$, $b(\theta_i) = \exp\theta_i$, $a(\phi) = 1$, $c(y_i,\phi) = -\ln(y_i!)$.

Ejemplo 7.3 (Distribución binomial negativa con dispersión conocida)**.** *La función de probabilidad de la binomial negativa tiene la expresión:*

$$f(y|\phi,p) = \frac{\Gamma(y+\phi)}{\Gamma(y+1)\Gamma(\phi)}\left(\frac{\mu}{\phi+\mu}\right)^y\left(\frac{\phi}{\mu+\phi}\right)^\phi,$$

*para $y = 0,1,\ldots$. Si suponemos que el parámetro ϕ es **conocido** es un elemento de la familia exponencial natural (modest18-3).*

Ejemplo 7.4 (Distribución normal)**.** *Vamos a ver el caso de la normal que hemos tratado previamente desde otro punto de vista. La normal pertenece a la familia de dispersión exponencial.*

$$f(y_i;\mu_i,\sigma^2) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left[-\frac{(y_i-\mu_i)^2}{2\sigma^2}\right] = \exp\left[\frac{y_i\mu_i - \frac{1}{2}\mu_i^2}{\sigma^2} - \frac{1}{2}\ln(2\pi\sigma^2) - \frac{y_i^2}{2\sigma^2}.\right] \quad (7.6)$$

Vemos que estamos en la familia de dispersión exponencial donde

$$\theta_i = \mu_i,\ b(\theta_i) = \frac{1}{2}\mu_i^2 = \frac{1}{2}\theta_i^2,\ a(\phi) = \sigma^2,\ c(y_i,\phi) = -\frac{1}{2}\ln(2\pi\sigma^2) - \frac{y_i^2}{2\sigma^2}.$$

Notemos que $a(\phi) = \sigma^2$ tiene la forma de un parámetro $\phi = \sigma^2$ dividido por una cantidad que en este caso es la unidad.

La componente sistemática de un modelo lineal generalizado es el vector $(\eta_1, \ldots, \eta_n)$ donde cada uno de los η_i es la combinación lineal de los predictores correspondientes a la i-ésima observación, es decir,

$$\eta_i = \sum_{j=1}^{n} \beta_j x_{ij} = \boldsymbol{x}_i' \boldsymbol{\beta},$$

con $i = 1, \ldots, N$ donde x_{ij} es el valor del j-ésimo predictor en el i-ésimo individuo. La combinación lineal $\sum_j \beta_j x_{ij}$ es el *predictor lineal*. Como es habitual, se suele considerar que uno de los predictores x_{ij} vale uno para todos los i de modo que consideramos el término independiente o constante.

La función de enlace (link function) g relaciona las componentes aleatoria y sistemática,

$$g(\mu_i) = \eta_i = \boldsymbol{x}_i' \boldsymbol{\beta}.$$

Tendremos que

$$\mu_i = g^{-1}(\eta_i) = g^{-1}(\boldsymbol{x}_i' \boldsymbol{\beta}).$$

A la función g^{-1} en ocasiones se le denomina la función respuesta.

Hay una función de enlace que tiene un interés especial, es aquella que nos transforma la media μ_i en el parámetro natural y recibe el nombre de **enlace canónico** (canonical link). Con esta función de enlace tendremos

$$\theta_i = \boldsymbol{x}_i^T \boldsymbol{\beta}.$$

Notemos que en el caso de la normal $\theta_i = \mu_i$ y allí igualamos la media al predictor lineal. De algún modo θ_i es el parámetro que sustituye a la media de la normal de una forma natural.

Con los ejemplos antes comentados tenemos que el enlace canónico es la función logit para la binomial, el logaritmo natural para la Poisson y la función identidad para la normal.

7.2 Verosimilitud, ajuste y distribución asintótica de los GLMs

Con modelos lineales generalizados vamos a estimar el vector de coeficientes maximizando la verosimilitud. Sus estimadores van a ser los máximo verosímiles del mismo modo que en modelos lineales normales.

7.2.1 Verosimilitud

La función de verosimilitud viene dada por

$$L(\boldsymbol{\beta}) = \prod_{i=1}^{n} \exp\left\{ \frac{y_i \theta_i - b(\theta_i)}{a(\phi)} + c(y_i, \phi) \right\} \tag{7.7}$$

de donde obtenemos la función de logverosimilitud como

$$\ell(\boldsymbol{\beta}) = \sum_{i=1}^{n} \frac{y_i \theta_i - b(\theta_i)}{a(\phi)} + \sum_{i=1}^{n} c(y_i, \phi). \tag{7.8}$$

[86] Estimating equations.

Las ecuaciones de verosimilitud (en la literatura se habla también de ecuaciones de estimación[86]) las obtenemos considerando las derivadas parciales e igualando a cero, en definitiva buscando el punto donde se anula el vector gradiente.

$$\frac{\partial \ell(\boldsymbol{\beta})}{\partial \beta_j} = \sum_{i=1}^{n} \frac{\partial \ell_i(\boldsymbol{\beta})}{\partial \beta_j}. \tag{7.9}$$

Los parámetros en un modelo lineal generalizado los estimamos utilizando los estimadores máximo verosímiles, esto es, maximimizando la logverosimilitud, $\ell(\boldsymbol{\beta})$.

7.2.2 Distribución asintótica de los estimadores

La distribución conjunta asintótica o distribución con grandes muestras de los estimadores máximo verosímiles de los coeficientes viene dada por el siguiente resultado.

Teorema 7.1. *Asintóticamente $\hat{\boldsymbol{\beta}}$,*

$$\hat{\boldsymbol{\beta}} \sim N_p(\boldsymbol{\beta}, (\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1}),$$

siendo $\boldsymbol{W}$ una matriz diagonal con las entradas

$$w_i = \left(\frac{\partial \mu_i}{\partial \eta_i}\right)^2 \frac{1}{var(Y_i)}.$$

Teniendo en cuenta el teorema 7.1 podemos estimar la matriz de covarianzas asintótica con

$$\widehat{var(\hat{\boldsymbol{\beta}})} = (\boldsymbol{X}^T \hat{\boldsymbol{W}} \boldsymbol{X})^{-1} \tag{7.10}$$

donde $\hat{\boldsymbol{W}}$ es la matriz $\boldsymbol{W}$ evaluada en $\hat{\boldsymbol{\beta}}$.

7.2.3 Estimación del predictor lineal y las medias

Una vez tenemos la distribución asintótica de $\hat{\boldsymbol{\beta}}$ podemos plantearnos estimar la componente sistemática que viene dada por

$$\boldsymbol{\eta} = \boldsymbol{X}\boldsymbol{\beta} \tag{7.11}$$

por lo que la estimamos con

$$\hat{\boldsymbol{\eta}} = \boldsymbol{X}\hat{\boldsymbol{\beta}}. \tag{7.12}$$

La matriz de covarianzas de estos estimadores sería

$$var(\hat{\boldsymbol{\eta}}) = \boldsymbol{X} var(\hat{\boldsymbol{\beta}}) \boldsymbol{X}^T. \tag{7.13}$$

Puesto que podemos aproximar la matriz de covarianzas de $\hat{\boldsymbol{\beta}}$ con $(\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1}$ que, a su vez, aproximamos con $(\boldsymbol{X}^T \hat{\boldsymbol{W}} \boldsymbol{X})^{-1}$.

¿Cómo estimamos el vector de medias y la matriz de covarianza de los estimadores? Obviamente $g(\mu_i) = \eta_i$ de donde $g(\hat{\mu}_i) = \hat{\eta}_i$. En resumen las medias estimadas serían $\hat{\mu}_i = g^{-1}(\hat{\eta}_i)$. Utilizando el método delta en el caso multivariante podemos aproximar la matriz de covarianzas de $\hat{\boldsymbol{\mu}}$ con

$$var(\hat{\boldsymbol{\mu}}) \approx \boldsymbol{D} var(\hat{\boldsymbol{\eta}}) \boldsymbol{D} \approx \boldsymbol{D}\boldsymbol{X}(\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X})^{-1} \boldsymbol{X}^T \boldsymbol{D}. \tag{7.14}$$

Obviamente son aproximaciones. Para construir un intervalo de confianza para μ_i lo que se hace es construirlo para η_i y luego aplicar la transformación g^{-1} a los extremos y de este modo no aplicamos la aproximación del método delta.

¿Qué entendemos por distribución asintótica en este contexto? Tenemos n observaciones. Aquí no entendemos por asintótico que el valor de n tienda a infinito. Aquí se considera un número fijo y para cada componente se incrementa el número de observaciones que dan lugar al valor aleatorio de esa componente. En el caso de binomial trabajamos con la media. Se entiende que el valor de n_i crece. En el caso de Poisson que la media crece en cada componente. Estos casos suelen tener $a(\phi) = \phi/\omega_i$ con un ω_i creciendo.

7.3 Bondad de ajuste

Estamos usando un modelo estocástico. Un modelo estocástico siempre es una visión simplificada del proceso que produce nuestros datos y_i. Utilizando este modelo tendremos unas predicciones o valores ajustados que denotamos por $\hat{\mu}_i$ (o $\hat{y}_i$). La diferencia entre ambos valores es lo que nos da la medida en que el modelo puede ser considerado aproximadamente válido. En esta sección se evalúa la bondad del ajuste. Hasta qué punto el modelo produce predicciones que son razonablemente compatibles con los datos observados.

7.3.1 Desviación

Consideramos un GLM con observaciones $\boldsymbol{y} = (y_1, \ldots, y_n)$ Sea $\ell(\boldsymbol{\mu}; \boldsymbol{y})$ la logverosimilitud expresada como función del vector de medias $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$. La máxima logverosimilitud será $\ell(\hat{\boldsymbol{\mu}}; \boldsymbol{y})$. Consideremos ahora el modelo en que no utilizamos para nada los predictores y estimamos la media μ_i con la propia observación y_i, esto es, hacemos $\mu_i = y_i$. La logverosimilitud en este caso será $\ell(\boldsymbol{y}; \boldsymbol{y})$. A este modelo con un parámetro por observación se le llama **modelo saturado**. Tenemos un ajuste perfecto completamente inútil. La diferencia entre el valor observado, y_i, y el valor estimado para su media (y para la observación misma),y_i, sería nulo. ¿Y de qué sirve cuando tengamos otros predictores en donde no conocemos la media? No sirve nada más que como modelo base. Los demás modelos los compararemos con este modelo saturado. Son el máximo alcanzable de la logverosimilitud, un valor de referencia.

Si $\hat{\boldsymbol{\mu}}$ es el estimador máximo verosímil bajo el modelo que estamos considerando entonces el estadístico del cociente de verosimilitudes contrastando H_0 frente a un modelo más general sería

$$-2\ln \frac{\text{máxima verosilimitud bajo el modelo}}{\text{máxima verosimilitud bajo el modelo saturado}} = -2[\ell(\hat{\boldsymbol{\mu}}; \boldsymbol{y})) - \ell(\boldsymbol{y}; \boldsymbol{y})]. \quad (7.15)$$

Si $\hat{\theta}_i$ denota el estimador máximo verosímil de θ_i y $\tilde{\theta}_i$ denota el estimador de θ_i cuando tenemos el modelo saturado entonces la expresión del

cociente sería

$$-2[\ell(\hat{\boldsymbol{\mu}};\boldsymbol{y})) - \ell(\boldsymbol{y};\boldsymbol{y})] = 2\sum_{i=1}^{n}\frac{y_i\tilde{\theta}_i - b(\tilde{\theta}_i)}{a(\phi)} - 2\sum_{i=1}^{n}\frac{y_i\hat{\theta}_i - b(\hat{\theta}_i)}{a(\phi)}. \quad (7.16)$$

En el caso particular y frecuente en que $a(\phi) = \frac{\phi}{\omega_i}$ entonces

$$2\frac{1}{\phi}\sum_{i=1}^{n}\omega_i[y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)] = \frac{D(y;\hat{\boldsymbol{\mu}})}{\phi}. \quad (7.17)$$

y recibe el nombre de **desviación escalada** mientras que $D(y;\hat{\boldsymbol{\mu}})$ recibe el nombre de **desviación** del modelo actual.

Definición 7.1 (Desviación)**.**

$$D(y;\hat{\boldsymbol{\mu}}) = 2\sum_{i=1}^{n}\omega_i[y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)]. \quad (7.18)$$

Definición 7.2 (Desviación escalada)**.**

$$D^*(y;\hat{\boldsymbol{\mu}}) = \frac{D(y;\hat{\boldsymbol{\mu}})}{\phi}. \quad (7.19)$$

Es obvio que $\ell(\hat{\boldsymbol{\mu}};\boldsymbol{y})) \leq \ell(\boldsymbol{y};\boldsymbol{y}))$ por lo que $D(y;\hat{\boldsymbol{\mu}}) \geq 0$.

7.3.2 Residuos

Supongamos un modelo con una función varianza $V(\mu)$ (recordemos que $var(Y) = a(\phi)b''(\theta_i) = a(\phi)V(\mu)$). Una forma natural de evaluar la bondad de ajuste del modelo es considerar la diferencia entre la observación y su predicción (de media u observación que son la misma). Un primer ejemplo es el residuo de Pearson.

Definición 7.3 (Residuo de Pearson)**.** *Siendo la función varianza $\nu(\mu)$ definimos el **residuo de Pearson** para la observación y_i como*

$$e_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\nu(\hat{\mu}_i)}}. \quad (7.20)$$

Ejemplo 7.5 (GLM binomial)**.** *Recordemos que y_i denota la proporción de éxitos en n_i pruebas. Tenemos $V(\pi_i) = \pi_i(1-\pi_i)$ y por lo tanto el residuo de Pearson será*

$$e_i = \frac{y_i - \hat{\pi}_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)}}. \quad (7.21)$$

Ejemplo 7.6 (GLM Poisson)**.** *En este caso $V(\mu) = \mu$ y por tanto el residuo de Pearson viene dado por*

$$e_i = \frac{y_i - \hat{\mu}_i}{\sqrt{\hat{\mu}_i}}. \quad (7.22)$$

Consideremos la desviación

$$D(\boldsymbol{y};\hat{\boldsymbol{\mu}}) = \sum_{i=1}^{n} d_i,$$

donde

$$d_i = 2\omega_i[y_i(\tilde{\theta}_i - \hat{\theta}_i) - b(\tilde{\theta}_i) + b(\hat{\theta}_i)].$$

Definición 7.4 (Desviación residual). *Se define la* ***desviación residual*** *o* ***residuo de la desviación*** *como*

$$\sqrt{d_i} \times signo(y_i - \hat{\mu}_i).$$

Por la definición que hemos dado desviación residual tenemos que la suma de los cuadrados de la desviación residual es igual a la desviación.

7.4 Regresión logística

[1] Nuestra variable respuesta es binaria: presencia o ausencia de un atributo. Nuestra variable respuesta la reducimos a dos posibles resultados. A uno de ellos arbitrariamente lo podemos llamar éxito (codificarlo como 1 es razonable) y el al otro resultado lo llamamos fracaso (0).

7.4.1 Datos

Leemos los datos. Son datos en donde tenemos SNPs, esto es, polimorfismos que afectan a un solo par de bases en el genoma. Son datos anonimizados. Leemos y arreglamos un poco los datos. La variable respuesta será `DM2` que nos indica si la persona tiene diabetes mellitus de tipo 2 o no. Las variables predictoras a considerar son las demás.

```
finput = system.file("extdata","SNPs_RM.csv",
                     package="tamidata3")
df = read.table(file = finput,header = TRUE,sep=",")
sel = c(1:5,8,10)
for(i in sel) df[,i] = as.factor(df[,i])
df[,6] = (df[,6] == "YES")*1.
df = df[complete.cases(df),]
names(df)[6:10] = c("DM2","age","gender","weight",
                    "smoking")
```

Podemos ver las primeras filas de los datos.

```
head(df,n=2)
```

```
  SNP1 SNP2 SNP3 SNP4 SNP5 DM2 age gender weight
2 AG AA AG GT TT 0 46 Female 72
6 GG AG AG GG CT 0 32 Male 69
   smoking
2 ex-smoker
6 ex-smoker
```

Los SNPs aparecen en las primeras columnas. Hemos de transformar estas variables para convertirlas en posibles predictores de acuerdo con distintos modelos genéticos. La siguiente función lo hace.

```
#' Transformation of the SNP to a genetic model
#' @description
#' Transformation of the SNP to a genetic model
#' @param x SNPs
#' @param type Model to be used
#' @param sep Separator
#' @export
  snp2model = function(x,type=c("codominant","dominant",
                                "recessive"),
                       sep=""){
      type = match.arg(type)
```

[1] Esta sección utiliza fundamentalmente el capítulo 5 de [6] y el capítulo 5 de [4]. El código utilizado para los ejemplos de [4] en parte procede de [136].

```
        x1 = substr(x,1,1)
        x2 = substr(x,2,2)
        a = table(c(x1,x2))
        recessive = names(which.min(a))
        dominant = names(which.max(a))
        x1 = (x1 == recessive)*1
        x2 = (x2 == recessive)*1
        if(type == "codominant") rs = x1+x2
        if(type == "dominant") rs = (x1+x2 == 0)*1.
        if(type == "recessive") rs = (x1+x2 == 2)*1.
        rs
    }
```

Construimos tres `data.frame`s donde los SNPs son codificados según cada uno de los modelos considerados, el mismo para todos ellos. Esto no tiene porqué ser así y podríamos considerar un modelo distinto para cada uno de ellos.

```
snpscol = 1:5
dfc = dfd = dfr = df
dfc[,snpscol] = apply(df[,snpscol],2,snp2model,
                      type="codominant")
dfd[,snpscol] = apply(df[,snpscol],2,snp2model,
                      type="dominant")
dfr[,snpscol] = apply(df[,snpscol],2,snp2model,
                      type="recessive")
```

Estos datos los usamos en lo que sigue para ilustrar.

7.4.2 Función de enlace

Para un conjunto de predictores dados x_i tendremos n_i pruebas Bernoulli que darán lugar a una observación binomial (bien como número de éxitos bien como una proporción de éxitos). Denotaremos las respuestas $y_i, \ldots, y_N$ y supondremos que denotan las proporciones de éxito, esto es, $n_i Y_i \sim Bi(n_i, \pi_i)$ para $i = 1, \ldots, N$. Cada i denota ahora una situación en la que repetimos n_i pruebas. Normalmente los predictores serán categóricos. En el caso de algún predictor continuo se ha de asumir que repetimos este valor en todas las pruebas. Por tanto:

$$EY_i = \mu_i = \pi_i.$$

El vector $(n_1, \ldots, n_N)$ denota los tamaños muestrales. Tenemos n_i pruebas asociadas a un mismo vector de predictores $\boldsymbol{x}_i$. El total de observaciones es $n = \sum_{i=1}^{N} n_i$.

Podemos considerar los datos *agrupados* y *no agrupados*. ¿Qué significa esto? Podemos considerar los datos originales en donde repetimos las covariables $\boldsymbol{x}_i$ y consideramos como respuesta uno o cero, esto es, la respuesta binaria. Tenemos una distribución Bernoulli de la respuesta. Obviamente, en este contexto, cuando hablamos de resultado asintótico nos referimos a que el tamaño $N = n$ tiende a infinito.

En el segundo caso, consideramos los datos agrupados. Esto corresponde habitualmente a la situación en que **todos** los predictores son categóricos. Si tenemos algún predictor que sea numérico (continuo) raramente tendremos más de una observación. En esta situación el valor de N es fijo y lo que crece es n_i para todos los i. El tamaño de las muestras en cada situación crece pero no el número de situaciones distintas que consideramos. A esto se le llama **small dispersion asymptotics**. Obviamente logramos disminuir la varianza de la proporción de éxitos en cada situación sin incrementar el número de situaciones distintas.

Ambas situaciones dan los mismos estimadores de los coeficientes y sus errores estándar son los mismos. No así la desviación. Es más cómodo trabajar con datos agrupados.

7.4.3 Modelos con variable latente

Supongamos que nuestra variable respuesta, Y^*, es una variable que no podemos observar y que verifica

$$Y_i^* = \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i.$$

Suponemos que los errores aleatorios ϵ_i son independientes con media nula y función de distribución común F. Supongamos que hay un punto de corte desconocido τ tal que la variable que realmente estamos observando es $Y_i = 0$ si $Y_i^* \leq \tau$ e $Y_i = 1$ si $Y_i^* > \tau$. Estamos considerando que lo que vemos es una versión discretizada en dos valores de una variable continua subyacente o latente que no podemos observar realmente. La función de probabilidad de la variable binaria observada se puede expresar en términos de la función de distribución F y del punto de corte desconocido τ.

$$P(Y_i = 1) = \pi_i = P(Y_i^* > \tau) = P\left(\sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i > \tau\right) =$$
$$1 - \left(\epsilon_i \leq \tau - \sum_{j=1}^{p} \beta_j x_{ij}\right) = 1 - F\left(\tau - \sum_{j=1}^{p} \beta_j x_{ij}\right). \quad (7.23)$$

Los datos nunca nos van a permitir conocer τ. Podemos pues considerar de un modo arbitrario que τ es nulo. De modo que

$$P(Y_i = 1) = \pi_i = 1 - F\left(-\sum_{j=1}^{p} \beta_j x_{ij}\right). \quad (7.24)$$

Si todos los parámetros son multiplicados por un mismo valor el modelo sería equivalente a dividir por ese mismo valor las predictoras. Realmente hablamos de un modelo equivalente en lo esencial. Podemos asumir también varianza dada. En los modelos habituales se suele considerar una distribución simétrica respecto del origen: $F(x) = 1 - F(-x)$. Nos queda

$$P(Y_i = 1) = \pi_i = F\left(\sum_{j=1}^{p} \beta_j x_{ij}\right). \quad (7.25)$$

En consecuencia se tiene que

$$F^{-1}(P(Y_i = 1)) = F^{-1}(\pi_i) = \sum_{j=1}^{p} \beta_j x_{ij}. \quad (7.26)$$

Los modelos más utilizados corresponden a este tipo de modelos.

Modelo probit

Un ejemplo de lo que acabamos de ver correspondería al caso en que Φ es la función de distribución de la normal estándar. Una función de

enlace a utilizar es Φ^{-1}. Cuando utilizamos esta función de enlace hablamos de un **modelo probit**. En este caso tenemos, por ecuación 7.26, que

$$\Phi^{-1}(P(Y_i = 1)) = \Phi^{-1}(\pi_i) = \sum_{j=1}^{p} \beta_j x_{ij}, \tag{7.27}$$

o bien que

$$P(Y_i = 1) = \pi_i = \Phi\left(\sum_{j=1}^{p} \beta_j x_{ij}\right). \tag{7.28}$$

Este es el modelo de **regresión probit**.

Modelo logit

Sin duda este modelo es el más usado en las aplicaciones y es conocido como *regresión logística*. La función de densidad de una **distribución logística estándar** es

$$f(x) = \frac{e^{-x}}{(1+e^{-x})^2} = \frac{1}{(e^{x/2}+e^{-x/2})^2} \tag{7.29}$$

con $x \in \mathbb{R}$. Su función de distribución viene dada por

$$F(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{1+e^x}. \tag{7.30}$$

En este modelo tendremos que

$$P(Y_i = 1) = \pi_i = F\left(\sum_{j=1}^{p} \beta_j x_{ij}\right) = \frac{e^{\sum_{j=1}^{p} \beta_j x_{ij}}}{1+e^{\sum_{j=1}^{p} \beta_j x_{ij}}} \tag{7.31}$$

Pero

$$F^{-1}(\pi_i) = \ln \frac{\pi_i}{1-\pi_i} = \sum_{j=1}^{p} x_{ij}\beta_j. \tag{7.32}$$

La función **logit** se define precisamente como

$$logit(\pi_i) = \ln \frac{\pi_i}{1-\pi_i}. \tag{7.33}$$

Por lo que en un modelo de **regresión logística** tenemos que el logit de la probabilidad de éxito es igual a la componente sistemática.

$$logit(\pi_i) = \sum_{j=1}^{p} x_{ij}\beta_j. \tag{7.34}$$

¿Qué significan los coeficientes β_j? Obviamente (como en todo modelo lineal o lineal generalizado) si $\beta_j = 0$ indica que la variable respuesta es condicionalmente independiente (dadas el resto de variables predictoras) de la j-ésima variable predictora. Notemos que $\frac{\pi_i}{1-\pi_i}$ son los odds. Nos fijamos en la j-ésima variable predictora. Asumimos que el efecto de las demás y la constante se mantiene constante y comparamos subpoblaciones con un valor dado x_{ij} con otra subpoblación con valor $x_{ij}+1$. Supongamos que los predictores de la primera y segunda subpoblaciones los denotamos $\boldsymbol{x}_i$ y $\boldsymbol{x}_i^*$. Denotamos por π_i y π_i^* las probabilidades correspondientes a los vectores $\boldsymbol{x}_i$ y $\boldsymbol{x}_i^*$. De (7.34) se sigue inmediatamente que

$$logit(\pi_i^*) - logit(\pi_i) = \beta_j, \tag{7.35}$$

pero

$$\beta_j = logit(\pi_i^*) - logit(\pi_i) = \ln\left(\frac{\pi_i^*/(1-\pi_i^*)}{\pi_i/(1-\pi_i}\right), \tag{7.36}$$

y el cociente que tenemos a la parte derecha de (7.36) es el logaritmo del cociente de los odds o log-odds ratio. O si lo preferimos

$$e^{\beta_j} = \frac{\pi_i^*/(1-\pi_i^*)}{\pi_i/(1-\pi_i}, \tag{7.37}$$

es decir, e^{β_j} corresponde con el cociente de los odds. El usuario suele encontrar más fácil interpretar el valor de e^{β_j}. Si es mayor que la unidad indica que el cociente de odds se incrementa en esa cantidad. Menor que la unidad indica un decremento. Obviamente el sentido del cambio viene en función de la codificación de la variable predictora que estamos considerando.

Ejemplo 7.7. *Vamos a considerar dos variables categóricas X e Y con I y J categorías. Un sujeto puede venir clasificado en una de $I \times J$ categorías.*

Nos fijamos en el modelo dominante y consideramos las variables `SNP1` *y* `clinica.DM2`.

```
(tab = table(dfd[,c("SNP1","DM2")]))
```

```
   DM2
SNP1  0  1
   0 351 30
   1 746 50
```

Podemos calcular el cociente de odds estos datos.

```
fisher.test(tab)
```

```
        Fisher's Exact Test for Count Data

data:  tab
p-value = 0.3232
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
 0.4796518 1.3014415
sample estimates:
odds ratio
 0.7843749
```

O simplemente

```
DescTools::OddsRatio(tab,conf.level=.95)
```

```
Error in loadNamespace(x): no hay paquete llamado 'DescTools'
```

Observemos que los intervalos de confianza son distintos pues se han calculado con diferentes métodos.

Podemos ajustar un modelo de regresión logística con respuesta `DM2` *y predictora la variable* `SNP1`.

```
fit = glm(DM2 ~SNP1, family="binomial",data=dfd)
summary(fit)
```

```
Call:
glm(formula = DM2 ~SNP1, family = "binomial", data = dfd)

Coefficients:
          Estimate Std. Error z value Pr(>|z|)
(Intercept) -2.4596 0.1902 -12.930 <2e-16
SNP1 -0.2431 0.2398 -1.014 0.311

(Intercept) ***
SNP1
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for binomial family taken to be 1)

   Null deviance: 584.63 on 1176 degrees of freedom
Residual deviance: 583.62 on 1175 degrees of freedom
AIC: 587.62

Number of Fisher Scoring iterations: 5
```

Podemos ver el odds ratio con

```
exp(coef(fit)[2])
```

```
    SNP1
0.7841823
```

Obviamente utilizando el modelo de regresión logística podemos considerar otras variables predictoras que nos permitan evitar confundir la influencia que tiene el uso de la aspirina con otras posibles variables predictoras no consideradas. El utilizar el cociente de odds se acaba aquí.

7.4.4 Predicción y error de clasificación

Una vez hemos ajustamo un modelo de regresión logística tenemos, para cada observación, una probabilidad de éxito estimada, $\hat{\pi}_i$. Si queremos una clasificación de la observación en éxito o fracaso podemos fijar una probabilidad de corte, π_0, y fijar una regla de clasificación que sea: si $\hat{\pi}_i > \pi_0$ entonces tomamos $\hat{y}_i = 1$, esto es, clasificamos la i-ésima observación como éxito. Si $\hat{\pi}_i \leq \pi_0$ entonces clasificamos como fracaso, $\hat{y}_i = 0$. Una probabilidad de corte habitualmente usada es $\pi_0 = 0.5$. Obviamente conocemos el grupo al que pertenece la observación. Conocemos y_i y por ello podemos construir una tabla de contingencia en donde consideremos los conteos de $(y_i, \hat{y}_i)$. Esta tabla recibe el nombre de **tabla de clasificación**.

Es habitual cuando aplicamos el procedimiento indicado utilizar el procedimiento leaving-one-out. ¿Cómo se hace? Para cada observación ajustamos el modelo sin utilizar esta observación, la dejamos fuera. Con este modelo ajustado sin la observación hacemos la predicción de la probabilidad π_i que podemos denotar, $\hat{\pi}_{(i)}$. Aplicamos la regla de decisión indicada con estas probabilidades y tendremos las predicciones $\hat{y}_{(i)}$. Construimos la tabla de clasificación con $(y_i, \hat{y}_{(i)})$.

La tabla de clasificación depende de la probabilidad de corte π_0. ¿Qué ocurre cuando queremos evaluar los resultados para distintos valores de π_0? Dado un valor de π_0 podemos considerar la sensibilidad y la especificidad. La sensibilidad sería la proporción de éxitos que son declarados

como éxitos. Dicho de otro modo, la tasa de verdaderos positivos. En términos probabilísticos $P(\hat{Y}=1|Y=1)$. También podemos considerar la especificidad o proporción de fracasos declarados como fracasos, la tasa de verdaderos negativos: $P(\hat{Y}=0|Y=0)$.

Supongamos que vamos tomando valores de π_0 de 1 a 0 decreciendo. Para cada valor de π_0 consideramos uno menos la especificidad en abscisas ($P(\hat{Y}=1|Y=0)$)) y la sensibilidad ($P(\hat{Y}=1|Y=1)$) en ordenadas. Esta es la curva ROC.[87] Una curva ROC es tanto mejor cuando más rápidamente crece y cuanto más próxima a uno está. El puro azar, la clasificación completamente aleatoria, daría una curva ROC igual a la función identidad.

[87] Receiver operating characteristic.

7.4.5 Desviación y bondad de ajuste

Una forma de evaluar el modelo que utilizamos es compararlo con modelos más complejos y ver que el ajuste no es mejor. De algún modo, estamos viendo que nuestro modelo es razonable, no es un mal modelo.

Otra forma de ver si el modelo falla es utilizar la desviación o estadísticos de Pearson.

7.4.6 Desviación y estadísticos de Pearson

La desviación compara nuestro modelo con un modelo saturado en donde igualamos cada una de las medias con la observación, $\tilde{\pi}_i = y_i$. La desviación viene dada por

$$-2\ln\frac{\prod_{i=1}^{N}\hat{\pi}_i^{n_iy_i}(1-\hat{\pi}_i)^{n_i-n_iy_i}}{\prod_{i=1}^{N}\tilde{\pi}_i^{n_iy_i}(1-\tilde{\pi}_i)^{n_i-n_iy_i}} = 2\sum_{i=1}^{N}n_iy_i\ln\frac{n_iy_i}{n_i\hat{\pi}_i}+2\sum_{i=1}^{N}(n_i-n_iy_i)\ln\frac{n_i-n_iy_i}{n_i-n_i\hat{\pi}_i}. \quad (7.38)$$

Tenemos N valores distintos de las covariables y n_iy_i es el número de éxitos en la situación i mientras que el número de fracasos viene dado por $n_i-n_iy_i$. Por tanto, n_iy_i y $n_i-n_iy_i$ son las valores **observados** de exitos y fracasos mientras que $n_i\hat{\pi}_i$ y $n_i-n_i\hat{\pi}_i$ son los éxitos y fracasos **ajustado** por el modelo. Por ello, de un modo abreviado, podemos escribir

$$D(\boldsymbol{y};\hat{\boldsymbol{\mu}}) = 2\sum Observado\ln\frac{Observado}{Ajustado}.$$

La desviación es distinta si consideramos las observaciones agrupadas o sin agrupar.

Con datos agrupados se puede utilizar el estadístico de Pearson como medida de bondad de ajuste del modelo.

$$X^2 = \sum_{i=1}^{n}\frac{(n_iy_i-n_i\hat{\pi}_i)^2}{n_i\hat{\pi}_i}+\sum_{i=1}^{n}\frac{(n_i-n_iy_i)-(n_i-n_i\hat{\pi}_i))^2}{n_i-n_i\hat{\pi}_i} = \sum\frac{(Observado\ -\ Ajustado)^2}{Ajustado}. \quad (7.39)$$

7.4.7 Residuos

Cuando tenemos datos agrupados, es útil comparar las proporciones observadas y ajustadas: y_i con $\hat{\pi}_i$. El residuo de Pearson viene dado por

$$e_i = \frac{y_i - \hat{\pi}_i}{\sqrt{\hat{\pi}_i(1-\hat{\pi}_i)/n_i}}.$$

Notemos que por la definición de este residuo tenemos que

$$X^2 = \sum_{i=1}^{N} e_i^2.$$

Podemos usar los residuos de la desviación. O bien utilizar los residuos estandarizados en donde el residuo $y_i - \hat{\pi}_i$ por su error estándar estimado. Si consideramos la matriz

$$\hat{\boldsymbol{H}}_{\boldsymbol{W}} = \hat{\boldsymbol{W}}^{1/2}\boldsymbol{X}(\boldsymbol{X}^T\hat{\boldsymbol{W}}\boldsymbol{X})^{-1}\boldsymbol{X}^T\hat{\boldsymbol{W}}^{1/2}$$

siendo $\hat{\boldsymbol{W}}$ la matriz diagonal con $\hat{w}_{ii} = n_i\hat{\pi}_i(1-\hat{\pi}_i)$. Si $\hat{h}_{ii}$ denota el elemento i-ésimo en la diagonal principal de $\hat{\boldsymbol{H}}_{\boldsymbol{W}}$ entonces el residuo estandarizado viene dado por

$$r_i = \frac{e_i}{\sqrt{1-\hat{h}_{ii}}}.$$

Si el modelo es correcto y n_i es grande entonces los residuos estandarizados tienen aproximadamente una distribución normal estándar.

7.4.8 Analizando SNPs

En esta sección vamos a evaluar cada uno de los modelos genéticos buscando asociación con la respuesta `clinica.DM2` que nos indica si la persona tiene la diabetes mellitus tipo 2.

Vamos a evaluar los modelos genéticos buscando asociación con la respuesta así como otras variables predictoras. Empezamos considerando el modelo codominante. Empezamos ajustando un modelo en el que consideramos como predictoras `SNP1`, `age`, `gender`, `weight` y `smoking`.

```
fit.c1 = glm(DM2 ~SNP1 + age + gender + weight +
                    smoking, family="binomial",data=dfc)
fit.c2 = glm(DM2 ~SNP2 + age + gender + weight +
                    smoking, family="binomial",data=dfc)
fit.c3 = glm(DM2 ~SNP3 + age + gender + weight +
                    smoking, family="binomial",data=dfc)
fit.c4 = glm(DM2 ~SNP4 + age + gender + weight +
                    smoking, family="binomial",data=dfc)
fit.c5 = glm(DM2 ~SNP5 + age + gender + weight +
                    smoking, family="binomial",data=dfc)
```

Para poder comparar podemos

```
fit.c0 = glm(DM2 ~age + gender + weight + smoking,
                family="binomial",data=dfc)
```

Podemos ver que las desviaciones obtenidas cuando comparamos los modelos son claramente no significativas ya que tenemos unas distribuciones aproximadas de una ji-cuadrado con un grado de libertad.

```
anova(fit.c1,fit.c0)
anova(fit.c2,fit.c0)
anova(fit.c3,fit.c0)
anova(fit.c4,fit.c0)
anova(fit.c5,fit.c0)
```

7.4.9 Ejercicios

Ex. 12 — Repetir el análisis que hemos realizado en § 7.4.8 para el modelo dominante.

Ex. 13 — Repetir el análisis que hemos realizado en § 7.4.8 para el modelo recesivo.

Ex. 14 — Programar una función que nos permita elegir el módelo genético más significativo para cada uno de los SNPs.

7.5 Datos de conteo

En esta sección tratamos la situación en que la variable respuesta son datos de conteo, un número entero no negativo. Es la situación habitual cuando tratamos con datos obtenidos mediante secuenciación.

7.5.1 Modelos loglineales Poisson

La variable respuesta Y es un conteo (número de lecturas alineadas sobre un genoma de referencia por ejemplo). Si asumimos que la distribución de Y es Poisson, $Y \sim Po(\mu)$, con media y varianza μ, donde $EY = var(Y) = \mu$. La función de probabilidad es

$$f(y;\mu) = \frac{e^{-\mu}\mu^y}{y!},$$

con $y = 0, 1, \ldots$. y la podemos expresar como un elemento de la familia exponencial natural del siguiente modo:

$$f(y;\mu) = \frac{e^{-\mu}\mu^y}{y!} = \exp\{y \ln \mu - \mu - \ln(y!)\}, \tag{7.40}$$

tomando $\theta_i = \ln \mu_i$, $b(\theta_i) = \exp \theta_i$, $a(\phi) = 1$, $c(y_i, \phi) = -\ln(y_i!)$. Por tanto la función enlace canónica es $\eta = \ln \mu$.

Si consideramos la situación más simple con una sola variable explicativa x y la función de enlace canónica entonces

$$\log \mu = \beta_0 + \beta_1 x.$$

En este modelo

$$\mu = \exp(\beta_0 + \beta_1 x) = e^{\beta_0}\left(e^{\beta_1}\right)^x.$$

¿Qué interpretación tiene este coeficiente? Supongamos que consideras x y $x+1$ y denotamo por μ_x y μ_{x+1} las medias respectivas para cada situación. Obviamente $\ln \mu_{x+1} - \ln \mu_x = \beta_1$ o $\ln\left(\frac{\mu_{x+1}}{\mu_x}\right) = \beta_1$.

Puesto que con el enlace canónico $\ln(\mu_i) = \eta_i$ entonces

$$\frac{\partial \mu_i}{\partial \eta_i} = \frac{1}{\mu_i},$$

y las ecuaciones de verosimilitud adoptan la expresión

$$\sum_{i=1}^{n} \frac{y_i - \mu_i}{var(Y_i)} \frac{x_{ij}}{\mu_i} = 0.$$

Vamos a obtener la desviación. Hemos visto que $\hat{\theta}_i = \ln \hat{\mu}_i$ y $b(\hat{\theta}_i) = \exp \hat{\theta}_i = \hat{\mu}_i$. En el modelo saturado: $\tilde{\theta}_i = \ln y_i$ y $b(\tilde{\theta}_i) = y_i$. Además $a(\phi) = 1$. En consecuencia, tanto la desviación como la desviación escalada vienen dadas por

$$D(y; \hat{\boldsymbol{\mu}}) = 2 \sum_{i=1}^{n} [y_i \ln \frac{y_i}{\hat{\mu}_i} - y_i + \hat{\mu}_i].$$

Si se asumen que utilizamos el enlace logarítmico y que tenemos la constante en el modelo tendremos que $\sum_{i=1}^{n} y_i = \sum_{i=1}^{n} \hat{\mu}_i$. La desviación queda como

$$D(\boldsymbol{y}; \hat{\boldsymbol{\mu}}) = 2 \sum_{i=1}^{n} y_i \ln \frac{y_i}{\hat{\mu}_i}.$$

Si observamos la expresión que hemos obtenido también puede expresarse (como en el caso de la binomial) en la forma simplificada

$$D(\boldsymbol{y}; \hat{\boldsymbol{\mu}}) = 2 \sum Observado \ln \left(\frac{Observado}{Esperado} \right). \tag{7.41}$$

La desviación que nos aparece en este caso corresponde con el estadístico G^2 ([4, 5]).

Una de las aplicaciones fundamentales de los GLM Poisson es modelizar conteos en tablas de contingencia. En este caso n, total de conteos es fijo y que crece son los valores esperados en cada celda. Bajo esta condición la desviación converge a una ji-cuadrado con $n-p$ grados de libertad donde p es el número de parámetros del modelo.

Ejemplo

Ejemplo 7.8. *Utilizamos para ilustrar los datos* `tamidata2:PRJNA218851` ↪ *. Son datos de cáncer colorrectal en donde tenemos una variable fenotípica con tres niveles.*

```
pacman::p_load(SummarizedExperiment)
data(PRJNA218851,package="tamidata2")
```

```
table(colData(PRJNA218851)[,"Stage"])
```

```
Cancer Metastasis Normal
    18 18 18
```

Nos fijamos en el gen que ocupa la fila 1000. Construimos un `data.frame` *en donde incluimos la variable* `Stage` *y el conteo correspondiente a este gen.*

```
df = data.frame(count = assay(PRJNA218851)[1000,],
                Stage=colData(PRJNA218851)[,"Stage"])
head(df)
```

```
                          count Stage
SRR975551Aligned.out.sam.bam 539 Cancer
SRR975552Aligned.out.sam.bam 563 Cancer
SRR975553Aligned.out.sam.bam 1018 Cancer
SRR975554Aligned.out.sam.bam 393 Cancer
SRR975555Aligned.out.sam.bam 398 Cancer
SRR975556Aligned.out.sam.bam 672 Cancer
```

Ajustamos un modelo loglineal de Poisson.

```
fit = glm(count ~Stage, family = poisson(link = log),data = df)
summary(fit)
```

```
Call:
glm(formula = count ~Stage, family = poisson(link = log), data = df)

Coefficients:
               Estimate Std. Error z value
(Intercept) 6.729957 0.008146 826.14
StageMetastasis -0.306800 0.012512 -24.52
StageNormal 0.429249 0.010467 41.01
              Pr(>|z|)
(Intercept) <2e-16 ***
StageMetastasis <2e-16 ***
StageNormal <2e-16 ***
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for poisson family taken to be 1)

   Null deviance: 13236.9 on 53 degrees of freedom
Residual deviance: 8723.7 on 51 degrees of freedom
AIC: 9189.2

Number of Fisher Scoring iterations: 4
```

La desviacion nula es la desviacion para el modelo que tiene solo la constante. La desviacion residual es la desviacion del modelo que tiene la constante y las variables binarias que describen `Stage`*. La diferencia entre los valores tiene una distribucion ji-cuadrado con dos grados de libertad y nos permite contrastar si los coeficientes de* `StageMetastasis` *y* `StageNormal` *pueden considerarse simultáneamente nulos.*

```
fit$null.deviance - fit$deviance
```

```
[1] 4513.206
```

Podemos rechazar confortablemente la hipotesis nula. Hay un aportacion significativa `Stage`*. Veamos lo que nos devuelve la funcion genérica* `summary` ↪ *sobre un objeto de clase* `glm`*.*

```
attributes(summary(fit))
```

```
$names
 [1] "call" "terms"
 [3] "family" "deviance"
 [5] "aic" "contrasts"
 [7] "df.residual" "null.deviance"
 [9] "df.null" "iter"
[11] "deviance.resid" "coefficients"
[13] "aliased" "dispersion"
[15] "df" "cov.unscaled"
[17] "cov.scaled"

$class
[1] "summary.glm"
```

Por ejemplo, podemos extraer los coeficientes.

```
summary(fit)$coefficients
```

```
               Estimate Std. Error z value
(Intercept) 6.7299568 0.008146275 826.13916
StageMetastasis -0.3068000 0.012512160 -24.52015
StageNormal 0.4292487 0.010467369 41.00827
                 Pr(>|z|)
(Intercept) 0.000000e+00
StageMetastasis 9.006867e-133
StageNormal 0.000000e+00
```

Si nos interesa uno dado no hay mas que ver la posicion y recuperarlo.

```
summary(fit)$coefficients[1, 2]
```

```
[1] 0.008146275
```

Veamos los atributos del objeto de clase `glm`.

```
attributes(fit)
```

```
$names
 [1] "coefficients" "residuals"
 [3] "fitted.values" "effects"
 [5] "R" "rank"
 [7] "qr" "family"
 [9] "linear.predictors" "deviance"
[11] "aic" "null.deviance"
[13] "iter" "weights"
[15] "prior.weights" "df.residual"
[17] "df.null" "y"
[19] "converged" "boundary"
[21] "model" "call"
[23] "formula" "terms"
[25] "data" "offset"
[27] "control" "method"
[29] "contrasts" "xlevels"

$class
[1] "glm" "lm"
```

En particular los valores esperados vienen dados por

```
head(fit$fitted.values)
```

```
SRR975551Aligned.out.sam.bam
                  837.1111
SRR975552Aligned.out.sam.bam
                  837.1111
SRR975553Aligned.out.sam.bam
                  837.1111
SRR975554Aligned.out.sam.bam
                  837.1111
SRR975555Aligned.out.sam.bam
                  837.1111
SRR975556Aligned.out.sam.bam
                  837.1111
```

El mismo resultado lo obtenemos mediante

```
head(fitted(fit))
```

```
SRR975551Aligned.out.sam.bam
                  837.1111
SRR975552Aligned.out.sam.bam
                  837.1111
SRR975553Aligned.out.sam.bam
                  837.1111
```

```
SRR975554Aligned.out.sam.bam
                  837.1111
SRR975555Aligned.out.sam.bam
                  837.1111
SRR975556Aligned.out.sam.bam
                  837.1111
```

Los coeficientes los tenemos con

```
fit$coefficients
```

```
   (Intercept) StageMetastasis StageNormal
     6.7299568 -0.3068000 0.4292487
```

o bien con

```
coef(fit)
```

```
   (Intercept) StageMetastasis StageNormal
     6.7299568 -0.3068000 0.4292487
```

Podemos predecir la media de la respuesta para el valor de `Stage` *que queramos con* `predict`.

```
predict(fit,type = "response",
        newdata = data.frame(Stage =c("Cancer","Metastasis","Normal")))
```

```
       1 2 3
 837.1111 615.9444 1285.8889
```

Sobredispersión en GLM Poisson

En una distribución de Poisson, la media y la varianza son iguales. Cuando trabajamos con conteos reales no suele ser cierta esta hipótesis. Con frecuencia la varianza es mayor que la media. A esto se le llama *sobre dispersión*. No es un problema cuando Y tiene una distribución normal pues la normal tiene un parámetro que la modeliza.

Ejemplo 7.9. *Vamos a estimar la sobre dispersión en un GLM Poisson.*

```
fit1 = glm(count ~Stage, family = quasipoisson(link = log),data = df)
summary(fit1)$dispersion
```

```
[1] 198.0956
```

La estimacion del parametro de dispersion no es mas que la suma de los residuos de Pearson dividida por los grados de libertad residuales.

7.5.2 GLM binomiales negativos

La densidad de la distribución binomial negativa es

$$f(y;\phi,\mu) = \frac{\Gamma(y+\phi)}{\Gamma(\phi)\Gamma(y+1)}\left(\frac{\phi}{\mu+\phi}\right)^{\phi}\left(1-\frac{\phi}{\mu+\phi}\right)^{y}$$

con $y = 0, 1, 2, \ldots$ donde ϕ y μ son los parámetros. Se tiene que

$$E(Y) = \mu, \quad var(Y) = \mu + \frac{\mu^2}{\phi}.$$

El parámetro $1/\phi$ es un *parámetro de dispersión*. Si $1/\phi$ tiende a 0 entonces $var(Y)$ tiende a μ y la distribución binomial negativa converge a una distribución de Poisson. Con ϕ fijo esta densidad está en la familia exponencial natural y podríamos hablar de un GLM binomial negativo.

Ejemplo 7.10.

```
library(MASS)
fit = MASS::glm.nb(count~Stage,data=df)
summary(fit)
```

```
Call:
MASS::glm.nb(formula = count ~Stage, data = df, init.theta =
    ↪ 5.191786539,
   link = log)

Coefficients:
              Estimate Std. Error z value
(Intercept) 6.7300 0.1038 64.858
StageMetastasis -0.3068 0.1468 -2.090
StageNormal 0.4292 0.1467 2.927
              Pr(>|z|)
(Intercept) < 2e-16 ***
StageMetastasis 0.03666 *
StageNormal 0.00343 **
---
Signif. codes:
0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

(Dispersion parameter for Negative Binomial(5.1918) family taken to be
    ↪ 1)

   Null deviance: 81.344 on 53 degrees of freedom
Residual deviance: 55.733 on 51 degrees of freedom
AIC: 796.61

Number of Fisher Scoring iterations: 1


            Theta: 5.192
        Std. Err.: 0.976

 2 x log-likelihood: -788.611
```

Capítulo 8

Comparaciones múltiples

8.1 Introducción

Empezamos con algunos comentarios bibliográficos. Esta sección se basa fundamentalmente en [42]. Sin embargo, un artículo a consultar es [120].

Supongamos que nuestras muestras corresponden a dos grupos que de un modo genérico podemos llamar casos y controles. Nuestro interés no está en evaluar si uno o dos genes concretos se expresan de un modo distinto entre las dos condiciones consideradas. Se pretende una visión global. Pretendemos responder a una pregunta como: ¿qué genes se expresan de un modo distinto (o diferencial en la jerga habitual en esta literatura) en los dos grupos que consideramos? Tenemos miles de genes: ¿cuáles de ellos tienen realmente una expresión diferencial? Y, lo importante, pretendemos responder la pregunta de modo que controlemos, de algún modo, las veces que admitimos una expresión diferencial cuando no la tiene. ¿Qué contrastes de hipótesis estamos evaluando? Si numeramos los genes con $i = 1, \ldots, N$ entonces para el i-ésimo gen estamos considerando el contraste de hipótesis siguiente:

- H_i: El gen i **no tiene** una expresión diferencial entre las condiciones consideradas.
- K_i: El gen i **tiene** una expresión diferencial entre las condiciones consideradas.

El contraste anterior se puede reformular como

- H_i: La expresión del gen i no tiene asociación con la condición.
- K_i: La expresión del gen i tiene asociación con la condición.

[1]

Estos contrastes nos lo planteamos para cada uno de los N genes que evaluamos. Denotemos por $G = \{1, \ldots, N\}$ el conjunto de hipótesis nulas que estamos evaluando. El número de hipótesis que vamos a contrastar es conocido a priori ya que corresponde con el número de genes que estamos evaluando. Denotamos por G_0 (con $G_0 \subset G$) las hipótesis nulas que son ciertas. Denotamos por $N_0 = |G_0|$ el cardinal del conjunto

[1] De este modo, creo que englobamos de un modo más natural el caso en que consideramos asociada a las muestras una covariable que no sea necesariamente un factor experimental con dos niveles.

Hipótesis nula	No rechazadas	Rechazadas	Total
Verdadera	U	V	N_0
Falsa	T	S	$N - N_0 = N_1$
Total	N - R	R	N

Tabla 8.1: Errores tipo I y II en contraste de múltiples hipótesis

G_0. Notemos que el conjunto G_0 es desconocido para nosotros. No sabemos ni cuántas ni cuáles son las hipótesis nulas ciertas. Esto supondría que conocemos qué genes se expresan diferencialmente y esto es precisamente nuestro objetivo. La situación en la en que nos encontramos viene descrita en la tabla 8.1 (de [16]).

En esta tabla conocemos N, esto es, el número de contrastes de hipótesis. Una vez hemos realizado todos los contrastes y tomado una decisión sobre si rechazamos o no cada hipótesis nula podemos **observar** R que nos indica cuántas hipótesis nulas hemos rechazado. Obviamente R es una variable aleatoria. Distintos datos nos darán distintos valores de R. Los valores de S, T, U, V son también aleatorios. Sin embargo, estas variables no son observables. La variable aleatoria V nos está dando el número (desconocido) de falsos positivos o errores tipo I (no hay expresión diferencial pero decidimos que la hay de un modo erróneo) mientras que T nos da el número de falsos negativos o error tipo II (genes que se expresan diferencialmente pero no admitimos que no lo hacen). Ambas variables indican error y son importantes.

Cuando realizamos un solo contraste el procedimiento consiste en fijar una cota al error tipo I, el nivel de significación, con un valor determinado que solemos denotar por α. Supongamos que denotamos por p_i el p-valor asociado al i-ésimo contraste. Entonces si aplicamos la regla de rechazar H_i cuando $p_i \leq \alpha$ y en otro caso aceptarla (o no rechazarla como se prefiera). Con este procedimiento sabemos que tenemos controlado el error tipo I **para el contraste** i**-ésimo** a un nivel α. Pero, y esto es fundamental, solamente para el ese contraste. No sabemos nada de lo que ocurre **simultáneamente** para todos los contrastes. Supongamos que tenemos un estadístico T_i que utilizamos para contrastar H_i y supongamos además que rechazamos la hipótesis nula para valores grandes de $|T_i|$[2] En este caso tendremos un valor c_α tal que rechazamos la hipótesis H_i cuando $\{|T_i| > c_\alpha\}$ y se tendrá que

$$P(|T_i| > c_\alpha | H_i) = \alpha.$$

Esto es, la probabilidad de rechazar cuando es cierta la hipótesis nula es α. O la probabilidad de no rechazar cuando es cierta la hipótesis nula H_i será de $1 - \alpha$, es decir,

$$P(|T_i| \leq c_\alpha | H_i) = 1 - P(|T_i| > c_\alpha | H_i) = 1 - \alpha.$$

Cuando realizamos simultáneamente muchos contrastes nos podemos equivocar rechazando la hipótesis nula cuando no lo es en más de una ocasión. De hecho, el número de hipótesis nulas falsamente rechazadas es una variable aleatoria que denotamos por V y nos da el número de veces que rechazamos la hipótesis nula erróneamente. Nuestro interés es que la variable aleatoria V tome valores pequeños con probabilidades altas.

[2] Una situación así la tenemos cuando observamos las muestras bajo dos condiciones distintas y utilizamos un test de la t de comparación de medias.

Las distintas medidas de error que se utilizan esencialmente cuantifican valores asociados a las variables que hemos definido en la tabla 8.1. Se habla de *tasas de error tipo I* abandonando la expresión de error tipo I utilizada con un solo test. Son extensiones del único error tipo I que se nos plantea cuando realizamos un solo contraste.

Tasa de error por comparación [3] Por $E(V)$ denotamos la media o valor medio de la variable V. Se define la tasa de error por comparación como el cociente entre la media de V y el número de contrastes.

$$PCER = \frac{E(V)}{N},$$

es la proporción esperada de errores tipo I entre las m decisiones que tomamos.

FWER: Tasa de error global [4] Esta medida de error es la que tiene una mayor tradición en Estadística. Quizás es la forma natural de pensar: no quiero equivocarme nunca rechazando una hipótesis nula cuando es cierta. La definimos como

$$FWER = P(V > 0) = P(V \geq 1).$$

Sin duda es la medida ideal. Ningún error. Esto tiene un pago. Es un criterio muy exigente si tenemos un número de hipótesis N muy grande. Notemos que nos fijamos en cometer al menos un error cuando con frecuencia tendremos decenas de miles de contrastes. Esto puede parecer algo bueno pero también estaremos construyendo tests que procurarán no rechazar una hipótesis nula salvo que tengan una evidencia contra ella muy grande. Procedimientos para contrastar miles de hipótesis nula y que hagan pequeña esta tasa de error tenderán a ser muy conservadores. Como (mala) contrapartida muchos genes que tendrán una expresión diferencial realmente no serán detectados. O dicho de un modo más técnico perderemos potencia en los contrastes donde la potencia es la probabilidad de rechazar H_i cuando realmente es falsa.

FDR: Tasa de falsamente rechazados [5] Se propuso en [16]. De alguna manera se vió que la tasa FWER era demasiado exigente y se trataba de conseguir procedimientos más potentes sin que por ello dejaramos de tener una tasa de error razonable controlada. Definimos la variable aleatoria $Q = V/R$ si $R > 0$ y $Q = 0$ en otro caso. Esta variable simplemente nos da la proporción de test que rechazamos erróneamente. Rechazamos R de los cuales V no debieran de ser rechazados, por tanto, nuestra proporción de hipótesis nulas falsamente rechazadas es Q. Tenemos el problema de si no rechazamos ninguna hipótesis. En este caso tendremos 0/0. Lo que se hace es definir el cociente como cero. Otra vez, Q es algo aleatorio, es una proporción aleatoria. ¿Qué queremos controlar de este valor aleatorio? La conocida como tasa de falsamente rechazados o *false*

[3] Per-comparison error rate.
[4] Familywise error rate.
[5] False discovery rate.

discovery rate denotada como FDR y que se define como

$$FDR = E(Q) = \\ E\left(\frac{V}{R}\middle| R > 0\right) P(R > 0) + 0 \times P(R = 0) = \\ E\left(\frac{V}{R}\middle| R > 0\right) P(R > 0), \quad (8.1)$$

que nos da la proporción esperada de hipótesis erróneamente rechazadas entre aquellas hipótesis que hemos rechazado. A la tasa FDR la hemos llamada tasa de falsamente rechazados. Hay una cierta confusión con la tasa de falsos positivos. Esta tasa sería la proporción de hipótesis nulas que son ciertas y que son rechazadas. Esto es, sería el valor medio de la variable V/N_0. Por ejemplo, si tenemos una FDR de 5% entonces el 5% de las hipótesis nulas que rechazamos son realmente ciertas. Si tenemos una tasa de falsos positivos del 5% entonces una media del 5% de las hipótesis nulas ciertas son erróneamente rechazadas. Si tenemos p-valores p_i y aplicamos simplemente la regla de rechazar H_i cuando $p_i \leq \alpha$ entonces tenemos una tasa de falsos positivos α.

pFDR: Tasa de falsamente rechazados positiva [6] Es una modificación de la anterior. Se define como

$$pFDR = E\left(\frac{V}{R}|R > 0\right).$$

Cuando tenemos un gran número de hipótesis nulas N entonces la probabilidad de que rechacemos al menos una es prácticamente segura, es decir, $P(R > 0) \approx 1$ y por lo tanto ambas tasas de error son también prácticamente iguales $FDR \approx pFDR$. Definimos la tasa pFDR porque está muy relacionada con el concepto de q-valor que definimos más adelante.

En lo que sigue vamos a considerar distintos procedimientos para contrastar muchas hipótesis. Un procedimiento concreto se dice que controla una tasa de error tipo I al nivel α cuando su tasa de error es menor o igual que α cuando aplicamos el procedimiento para producir una lista de R hipótesis nulas rechazadas o de genes declarados como significativos. La tasa de error que utilizaremos fundamentalmente en lo que sigue será la tasa tasa de falso rechazo (FDR).

8.2 Control fuerte y control débil del error

Cuando hemos definido las tasas de error hemos considerado el comportamiento aleatorio de V y R (conjunto y marginal de hecho). Ese comportamiento aleatorio depende de qué hipótesis nulas son ciertas y qué hipótesis son falsas. Por ejemplo, cuando hemos definido

$$FWER = P(V \geq 1),$$

realmente la probabilidad depende de qué hipótesis nulas son ciertas y qué hipótesis nulas son falsas. Lo correcto hubiera sido escribir:

$$FWER = P(V \geq 1 \mid \cap_{i \in M_0} H_i),$$

[6] Positive false discovery rate.

es decir, la probabilidad de rechazar al menos una hipótesis nula equivocadamente condicionando a que se verifican las hipótesis de M_0.[7] Un detalle importante es que cuándo controlamos una tasa de error tipo I es saber qué hipótesis nulas asumimos que son ciertas. Un par de definiciones que son importantes.

Definición 8.1 (Control fuerte)**.** *Se dice que un procedimiento de contraste de múltiples hipótesis realiza un control fuerte de la tasa de error tipo I elegida cuando dicha tasa de error es menor o igual que el límite que especifiquemos para cualquier combinación de hipótesis ciertas y falsas, en otras palabras, para cualquier subconjunto $M_0 \subset \{1, \ldots, m\}$ de hipótesis ciertas (el resto las suponemos falsas).*

Definición 8.2 (Control débil)**.** *Se dice que un procedimiento de contraste de múltiples hipótesis realiza un control débil de la tasa de error tipo I elegida cuando dicha tasa de error es menor o igual que el límite que especifiquemos asumiendo que todas las hipótesis nulas son ciertas. Asumimos pues que se verifica la hipótesis nula*

$$H_0^C = \cap_{i=1}^N H_i.$$

En el control débil asumimos algo que no tiene muchos visos de realidad. Estamos asumiendo que **todas las hipótesis nulas son ciertas**. Si pensamos en nuestro problema estamos diciendo que ningún gen tiene una expresión diferencial entre condiciones. No sé pero mal lo tenemos si es así. Parece poco realista asumir esto. Lo natural es asumir que algunas hipótesis nulas son ciertas y otras falsas. Es decir, que existe un conjunto N_0 que nos da las que son ciertas y aquellos que no están en N_0 son falsas. ¿Y quién es N_0? Ese es nuestro problema precisamente. En el control fuerte nos preparamos *para cualquier* N_0 sea el que sea. En lo que sigue cuando pongamos probabilidades referidas a V (o cualquier otra variable) siempre han de entenderse como probabilidades condicionadas a la realidad (desconocida para nosotros), es decir, probabilidades condicionadas a $\cap_{i \in N_0} H_i$ siendo N_0 el conjunto de hipótesis nulas ciertas (en nuestro caso, los genes que están en N_0 no se expresan de un modo diferencial y sí lo hacen los que no están en N_0).

8.3 Relación entre las tasas de error tipo I

Tenemos el siguiente resultado que relaciona las distintas tasas de error.

Proposición 8.1.

$$PCER \leq FDR \leq pFDR \leq FWER.$$

Prueba. Se sigue de $PCER = E(V/N)$, $FDR = E(Q)$ y $FWER = E(1_{V>0})$ y tenemos que , si $R > 0$, entonces

$$\frac{V}{N} \leq Q \leq 1_{V>0},$$

y tomando esperanzas se sigue. Además notemos que

$$FDR = pFDRP(R > 0) \leq pFDR \leq E(1_{V>0}).$$

□

[7] Un detalle, no conocemos M_0.

Esto significa que si un procedimiento controla la FWER entonces tenderá a rechazar menos hipótesis nulas, tenderá a ser más conservador por tanto y, posiblemente, nos estaremos perdiendo genes que tienen expresión diferencial y no lo apreciamos.

La aproximación clásica al problema de las contrastes múltiples se basa en el control fuerte de la FWER. Una aproximación más reciente propuesta en [16] consiste en controlar la FDR en un sentido débil.

8.4 p valores y p valores ajustados

Para cada contraste H_i tendremos un p-valor p_i. En el test de la t para comparar medias tenemos

$$p_i = P(|T_i| \geq t_i | H_i).$$

donde t_i es el i-ésimo estadístico observado. A los p-valores p_i los llamaremos p-valores originales. Podemos contrastar el contraste H_i con un nivel de significación α_i rechazando la hipótesis nula H_i si $p_i \leq \alpha_i$ y no rechazando en otro caso. Cuando consideramos simultáneamente todos los tests los valores α_i serán distintos y por ello tendríamos que ir comprobando si la desigualdad $p_i \leq \alpha_i$ se verifica o no. La idea del p-valor ajustado es transformar el p-valor p_i en otro valor $\tilde{p}_i$, el i-ésimo p-valor ajustado, de modo que sean equivalentes: $p_i \leq \alpha_i$ y $\tilde{p}_i \leq \alpha$ siendo α el valor que especificamos para controlar alguna de las tasas de error tipo I.

8.5 Métodos que controlan la FWER

Veremos procedimientos que actúan en un solo paso y procedimientos por pasos bien de bajada o de subida.

8.5.1 Los métodos en un solo paso

Veamos el ejemplo más conocido, es el método de Bonferroni. Supongamos que tenemos el p-valor p_i asociado al contraste H_i. Si solamente estuvieramos contrastando la hipótesis H_i entonces rechazamos con un error tipo I α si $p_i < \alpha$ y aceptamos o no rechazamos en otro caso. La modificación de Bonferroni es simple. Ahora tenemos en cuenta que hay m contrastes y rechazamos H_i si

$$p_i \leq \frac{\alpha}{N} \tag{8.2}$$

o, equivalentemente, si

$$Np_i \leq \alpha. \tag{8.3}$$

Si tenemos en cuenta que $0 < \alpha \leq 1$ entonces la desigualdad 8.3 es equivalente a que

$$\min\{Np_i, 1\} \leq \alpha. \tag{8.4}$$

De este modo el p-valor ajustado será $\tilde{p}_i = \min\{Np_i, 1\}$. El método de Bonferroni controla la FWER de un modo débil. Veámoslo. Denotemos por E_i el suceso consistente en que no rechazamos H_i o, si se prefiere,

aceptamos H_i. Se busca que $P(\cap_{i=1}^N E_i | \cap_{i=1}^N H_i) = 1 - \alpha$. Las probabilidades que siguen se entienden que son condicionadas a que todas las hipótesis nulas son simultáneamente ciertas. Tenemos

$$P(\cap_{i=1}^N E_i) = 1 - P(\cup_{i=1}^N E_i^c) \leq 1 - \sum_{i=1}^N P(E_i^c). \tag{8.5}$$

Pero $P(E_i^c)$ es la probabilidad de rechazar H_i siendo cierta $\cap_{i=1}^N H_i$. Supongamos que consideramos una región crítica de tamaño α/m, es decir, $P(E_i^c) \leq \alpha/m$. Teniendo en cuenta 8.5 se sigue

$$P(\cap_{i=1}^N E_i) \leq 1 - \sum_{i=1}^N P(E_i^c) \leq 1 - m\frac{\alpha}{N}. \tag{8.6}$$

Método de Bonferroni Este método nos da un control fuerte de la FWER al nivel α.En este método rechazamos la hipótesis nula H_i si el correspondiente p-valor sin ajustar es menor o igual a $\frac{\alpha}{N}$. El p-valor ajustado sería:

$$\tilde{p}_i = \min\{Np_i, 1\}$$

Método de Sĭdák Nos permite un control débil del FWER. Los p-valores ajustados son

$$\tilde{p}_i = 1 - (1 - p_i)^N.$$

Asume la independencia de los p valores sin ajustar, en definitiva, asume la independencia de los tests. En nuestro contexto no es muy asumible esta hipótesis pues los genes tienen expresiones relacionadas, no independientes.

Método de minP en un solo paso Los p-valores ajustados son

$$\tilde{p}_i = P\left(\min_{1 \leq l \leq N} P_l \leq p_i \mid H_o^C\right),$$

siendo H_o^C la hipótesis nula completa (la intersección de todas las hipótesis nulas) y P_l la variable aleatoria que nos da el p-valor aleatorio de la l-ésima hipótesis.

maxT en un solo paso Podemos considerar como una opción alternativa como el máximo de los valores de los estadísticos observados. Es el maxT en un solo paso donde los p-valores ajustados son

$$\tilde{p}_i = P\left(\max_{1 \leq l \leq N} \mid T_l \mid \geq \mid t_i \mid\mid H_o^C\right),$$

Los procedimientos basados en maxT o minP nos dan un control débil de FWER.

En general, los métodos de un solo paso son simples de implementar pero tiende a ser conservadores, esto es, tienden a no rechazar la hipótesis nula. No son pues muy potentes en el sentido de detectar genes con expresión diferencial. Los métodos por pasos son más potentes.

8.5.2 Los métodos por pasos de bajada

Los p-valores originales son $p_1, \ldots, p_m$. Los ordenamos de menor a mayor:

$$p_{r_1} \leq p_{r_2} \leq \ldots \leq p_{r_N}.$$

Tendremos las correspondientes hipótesis nulas $H_{r_1} \leq H_{r_2} \leq \ldots \leq H_{r_N}$

Método de Holm Definimos

$$i^* = \min\{i : p_{r_i} > \frac{\alpha}{m-i+1}\}$$

y rechazamos las hipótesis nulas H_{r_i} para $i = 1, \ldots, i^* - 1$. Si no existe i^* entonces rechazamos todas las hipótesis nulas. Si consideramos los p-valores ajustados entonces vienen dados por

$$\tilde{p}_{r_i} = \max_{k=1,\ldots,i}\{\min\{m-k+1)p_{r_k}, 1\}\}$$

El método de Holm es menos conservador que el método de Bonferroni.

Método de Sĭdák por pasos de bajada Definimos los p-valores ajustados como:

$$\tilde{p}_{r_i} = \max_{k=1,\ldots,i}\{1-(1-p_{r_k})^{m-k+1}\}.$$

min P por pasos de bajada Se definen los p-valores ajustados como:

$$\tilde{p}_{r_i} = \max_{k=1,\ldots,i}\left\{P\left(\min_{l\in\{r_k,\ldots,r_m\}} P_l \leq p_{r_k} \mid H_o^C\right)\right\}$$

max T por pasos de bajada Los p-valores ajustados son:

$$\tilde{p}_{r_i} = \max_{k=1,\ldots,i}\left\{P\left(\max_{l\in\{r_k,\ldots,r_m\}} \mid T_j \mid \geq \mid t_{r_k} \mid\mid H_o^C\right)\right\}$$

donde $|t_{r_1}| \geq |t_{r_2}| \geq \ldots \geq |t_{r_N}|$ son los estadísticos ordenados.

8.5.3 Método por pasos de subida

Vamos a ver el método de Hochberg. Se pretende, como en los anteriores, controlar el FWER a un nivel α. Consideramos

$$i^* = \max\{i : p_{r_i} \leq \frac{\alpha}{m-i+1}\}$$

y se rechazan las hipótesis nulas H_{r_i} para $i = 1, \ldots, i^*$. Los p-valores ajustados de Hochberg son

$$\tilde{p}_{r_i} = \min_{k=i,\ldots,N}\left\{\min\{(m-k+1)p_{r_k}, 1\}\right\}.$$

Este procedimiento es una versión en sentido inverso del procedimiento de Holm.

8.6 Métodos que controlan el FDR

Los métodos de la sección anterior controlan el FWER. En resumen estamos controlando el no cometer ningún error tipo I (admitir un gen con expresión diferencial cuando no la tiene). Esto produce procedimientos conservadores. En [16] se propuso la idea de que cuando contrastamos muchas hipótesis podemos tolerar algunos errores tipo I, siempre que el número de estos errores sea pequeño en relación con el número de hipótesis que se rechazan. Esta es la idea de controlar el FDR. Dos son los procedimientos que vamos a ver para controlar esta tasa.

Benjamini y Hochberg Este procedimiento fue propuesto en [16]. Siendo $p_{r_1} \leq \ldots \leq p_{r_N}$ son los p-valores originales ordenados entonces consideramos

$$i^* = \max\{i : p_{r_i} \leq \frac{i}{N}\alpha\}$$

y rechazamos H_{r_i} para $i = 1, \ldots, i^*$. Si no existe i^* entonces no rechazamos ninguna hipótesis. Los p-valores ajustados se definen como

$$\tilde{p}_{r_i} = \min_{k=i,\ldots,N} \left\{ \min\{\frac{N}{k} p_{r_k}, 1\} \right\}.$$

Benjamini y Yekutieli Propuesto en [17]. Como en el anterior, $p_{r_1} \leq \ldots \leq p_{r_N}$ son los p-valores originales ordenados. En este caso los p-valores ajustados se definen como

$$\tilde{p}_{r_i} = \min_{k=i,\ldots,N} \left\{ \min\{\frac{m \sum_{j=1}^{N} 1/j}{k} p_{r_k}, 1\} \right\}.$$

8.7 Utilizando genefilter y p.adjust

No vamos a realizar en principio ninguna selección no específica y nos planteamos trabajar con todos los genes.

8.7.1 Cálculo de p-valores ajustados

Calculamos los estadísticos t (test de la t con varianzas distintas) y los p-valores asociados.

Utilizamos los datos tamidata::gse1397.

```
data(gse1397,package ="tamidata")
eset = gse1397; y = pData(gse1397)[,"type"]
```

Aplicamos los t-tests para cada gen.

```
tt = genefilter::rowttests(eset,y)
```

Los p-valores originales los guardamos en `p0`.

```
p0 = tt[,"p.value"]
```

Supongamos que vamos a utilizar el método de Benjamini-Hochberg. Entonces pasamos de los p-valores originales (raw p-values) a los p-valores ajustados con la función `stats::p.adjust()`.

```
p.BH = p.adjust(p0, method = "BH")
```

Podemos incorporar los p-valores ajustados al `data.frame` original.

```
tt1 = data.frame(tt,p.BH)
```

En `stats::p.adjust()` tenemos otros procedimientos de ajuste de p-valores.

8.7.2 Genes con expresión diferencial

Lo primero es fijar una tasa de error α. En concreto podemos considerar el valor $\alpha = 0.05$.

```
alpha = 0.05
```

Consideremos los p-valores ajustados utilizando Benjamini-Hochberg manteniendo el orden original de la matriz de expresión. Observad que la segunda columna tiene los p-valores ajustados por el método Benjamini-Hochberg.

```
p1 = p.adjust(p0, "BH")
```

Los genes significativos, esto es, aquellos que tienen un p-valor ajustado menor a la tasa especificada ocupan las siguientes filas de la matriz de expresión.

```
(significativos.BH = which(p.BH < alpha))
```

```
[1] 346 1170 1853 6303 7889
```

¿Cuántos eran significativos con los p-valores originales?

```
significativos.p0 = which(p0 < alpha)
length(significativos.p0)
```

```
[1] 902
```

8.8 El q-valor

Esta sección se basa en [132]. Supongamos que ordenamos de menor a mayor los p-valores calculados para los distintos contrastes.

$$p_{r_1} \leq \ldots \leq p_{r_N}$$

Cuando hacemos este ordenamiento, como hemos visto en los procedimientos anteriores, declarar significativo uno de estos p-valores supone declarar significativos a todos los que son menores o iguales que él. Los métodos anteriores buscaban un punto de corte para estos p-valores ordenados. Por debajo de este punto de corte admitimos que los genes se expresan diferencialmente (rechazamos al hipótesis nula correspondiente) y por encima aceptamos la hipótesis nula. La idea del q-valor es asociar a cada test un valor numérico que nos diga lo extremo que es su p-valor considerando el resto de p-valores. De un modo intuitivo, si consideramos un test determinado nos da la proporción esperada de falsos positivos en la que incurrimos cuando declaramos significativo ese test.

¿Qué es el q-valor?

Posiblemente el mejor modo de entender qué es el q-valor sea ver un buen procedimiento para estimarlo propuesto en [132] e implementado en el paquete [133, qvalue]. Fijamos un valor t y consideremos que rechazamos H_i cuando $P_i \leq t$.[8] Consideremos los siguientes valores

$$V(t) = |\{P_i : P_i \leq t; H_i \text{ es cierta}; i = 1, \ldots, N\}| \tag{8.7}$$

y

$$R(t) = |\{P_i : P_i \leq t; i = 1, \ldots, N\}| \tag{8.8}$$

Se puede aproximar la pFDR con

$$E\left[\frac{V(t)}{R(t)}\right]$$

Tenemos un gran número de contrastes N por ello aproximadamente se tiene que

$$E\left[\frac{V(t)}{R(t)}\right] \approx \frac{EV(t)}{ER(t)}.$$

Podemos estimar $R(t)$ simplemente contando el número de p_i observados que son menores o iguales a t,

$$\hat{R}(t) = |\{P_i : P_i \leq t\}|$$

Para estimar $V(t)$ hemos de tener en cuenta que si la hipótesis nula H_i es cierta entonces el p-valor aleatorio P_i se distribuye uniforme en el intervalo $[0, 1]$ y por tanto la probabilidad de que sea menor o igual a t es precisamente t,

$$P(P_i \leq t | H_i) = t.$$

Por tanto

$$EV(t) = N_0 t.$$

Pero no conocemos N_0 número de hipótesis nulas ciertas. Lo que vamos a hacer es estimar la proporción de hipótesis nulas ciertas $\pi_0 = N_0/N$ (notemos que el total de hipótesis, N, sí es conocido). Bajo H_i, el p-valor aleatorio P_i es uniforme en $[0, 1]$, por tanto los p-valores correspondientes a hipótesis nulas ciertas se distribuyen sobre todo el intervalo $[0, 1]$, sin embargo, aquellos p-valores muy pequeños, muy próximos a cero, es más probable que correspondan a hipótesis nulas falsas. Fijamos un valor $\lambda \in [0, 1]$, un estimador razonable para π_0 es

$$\hat{\pi}_0 = \frac{|\{p_i : p_i > \lambda; i = 1, \ldots, N\}|}{N(1 - \lambda)}.$$

Podemos pues estimar pFDR con

$$\widehat{pFDR} = \frac{\hat{\pi}_0 N t}{|\{p_i : p_i \leq t\}|}.$$

El q-valor asociado a un contraste sería el mínimo valor de pFDR que se alcanza cuando el contraste es rechazado. Es decir, el q-valor asociado al test i-ésimo sería

$$q(p_i) = \min_{t \geq p_i} pFDR(t)$$

[8] Como es habitual denotamos por P_i el p-valor antes de ser observado, esto es, el valor aleatorio antes de tomar los datos.

y su estimador sería

$$\hat{q}(p_i) = \min_{t \geq p_i} \widehat{pFDR}(t).$$

El estimador $\hat{\pi}_0$ depende de un valor λ. De hecho el procedimiento exacto no utiliza un valor λ concreto sino que estima utilizando distintos valores y luego hace un ajuste tomando como estimador el valor estimado a partir del ajuste en 1.

El algoritmo exacto es el siguiente:

1. Sean $p_{r_1} \leq \ldots \leq p_{r_N}$ los p-valores ordenados.
2. Para un rango de valores de λ, por ejemplo, λ de 0 a 0.95 con incrementos de 0.01 calculamos
$$\hat{\pi}_0(\lambda) = \frac{|\{p_i : p_i > \lambda; i = 1, \ldots, N\}|}{N(1-\lambda)}.$$
3. Determinamos $\hat{f}$ el spline natural cúbico con 3 grados de libertad a partir de los valores $\hat{\pi}_0(\lambda)$.
4. Fijamos como estimador de π_0 el valor
$$\hat{\pi}_0 = \hat{f}(1).$$
5. Calculamos
$$\hat{q}(p_{r_m}) = \min_{t \geq p_{r_N}} \frac{\hat{\pi}_0 N t}{|\{p_i : p_i \leq t\}|} = \hat{\pi}_0 p_{r_N}.$$
6. Para $i = N-1, \ldots, 1$ calculamos
$$\hat{q}(p_{r_i}) = \min_{t \geq p_{r_i}} \frac{\hat{\pi}_0 N t}{|\{p_i : p_i \leq t\}|} = \min\{\frac{\hat{\pi}_0 N p_{r_i}}{i}, \hat{q}(p_{r_{i+1}})\}.$$
7. El q-valor estimado para el i-ésimo contraste más significativo es $\hat{q}(p_{r_i})$.

Calculando el q-valor

El método de estimación que hemos visto en § 8.8 está implementado en el paquete [133, qvalue]. Consultar http://genomics.princeton.edu/storeylab/qvalue/. Utilizamos los datos tamidata::gse1397.

```
pacman::p_load("Biobase")
data(gse1397,package="tamidata")
tt = genefilter::rowttests(gse1397,pData(gse1397)[,"type"])
pvalue = tt$p.value
```

Utilizamos el paquete [133, qvalue].

```
pacman::p_load(qvalue)
qobj = qvalue(pvalue)
```

Podemos ver una descripción gráfica con

```
plot(qobj)
```

que aparece en la figura 8.1. En concreto tenemos:

Arriba izquierda: Los distintos valores estimados de π_0, proporción de hipótesis nulas ciertas, cuando variamos el valor de λ. El valor estimado por el método antes propuesto nos da un 96.3% de genes sin expresión diferencial.

Arriba derecha: La transformación que nos lleva de los p-valores originales (ordenados de menor a mayor) a los q-valores.

Abajo izquierda: Para distintos puntos de corte para los q-valores el número de genes que declaramos con expresión diferencial significativa.

Abajo derecha: En abscisas el número de test significativos y en ordenadas el número de falsos positivos esperado.

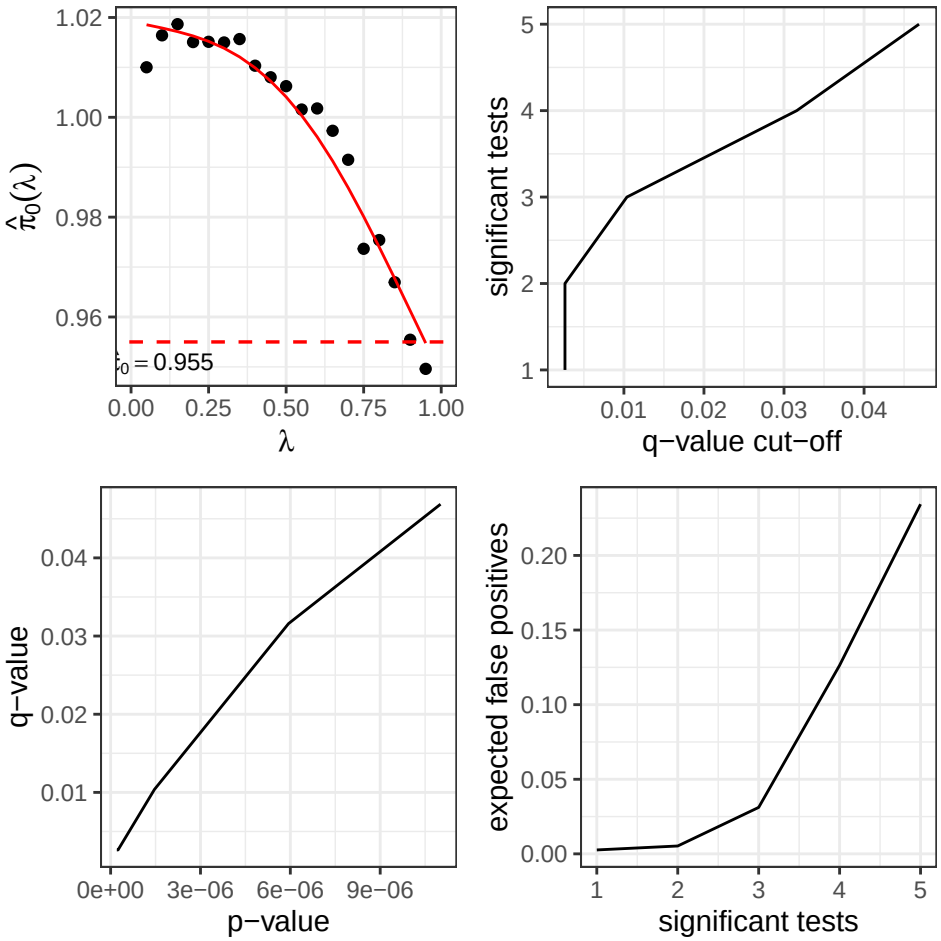


Figura 8.1: Distintos gráficos asociados a un objeto qvalue::qvalue.

Si simplemente queremos los q-valores los obtenemos con

```
q.value = qvalue(pvalue)$qvalues
```

8.9 Inferencia selectiva

En este capítulo nos hemos ocupado del problema de muchos tests y la corrección de los p-valores originales para controlar errores de falsos positivos. No es la única situación en donde se presenta esta problemática. En modelos lineales la selección de un buen modelo, esto es, la selección de un conjunto de predictoras que expliquen la respuesta también nos lleva a situaciones de este tipo. El término que se está utilizando es el de **Inferencia Selectiva** porque hacemos inferencia después de haber seleccionado variables de un conjunto mucho mayor de predictores. Para una presentación sencilla del problema en el contexto de la selección de variables hacia adelante (forward selection) en regresión lineal múltiple y en regresión lasso es muy interesante [**TaylorTibshirani15**]. Una presentación más amplia en [69, capítulo 6].

8.10 Ejercicios

* **Ex. 15 —** Utilizando los datos `tamidata::gse20986` se pide:

1. Seleccionar las muestras correspondientes a iris y huvec.
2. Aplicamos, a cada gen, un test de la t (asumiendo una misma varianza). Obtener los t-estadísticos y los p-valores correspondientes. Utilizad `genefilter::rowttests()`.
3. Utilizando el método de corrección de Benjamini-Hochberg obtener los p-valores ajustados.[88]
4. Si utilizamos un valor de FDR igual a 0.05: ¿qué genes declaramos como significativos?
5. Repetir los apartados anteriores comparando los muestras correspondientes a retina y huvec.
6. Repetir los apartados anteriores comparando las muestras correspondientes a coroides y huvec.
7. Utilizando la función `base::intersect()` y la función `Biobase::featureNames()` determinar los genes comunes a cada par de comparaciones y los comunes a las tres comparaciones.

[88] Utilizad [115, multtest].

7.* **Ex. 16 —** Vamos a utilizar los datos `tamidata::gse1397`. Vamos a realizar un estudio de expresión diferencial marginal comparando las personas con y sin la trisomía del cromosoma 21.

1. Aplicamos, a cada gen, un test de la t (asumiendo una misma varianza). Obtener los t-estadísticos y los p-valores correspondientes.
2. Si aplicamos la regla de decisión consistente en admitir una expresión diferencial cuando el p-valor es menor que $\alpha = 0.05$: ¿cuántos y qué genes son declarados significativos?
3. Utilizando los métodos de corrección de Bonferroni y de Benjamini-Hochberg obtener los p-valores ajustados para cada uno de los procedimientos.
4. Representar, en un mismo dibujo, los p-valores originales y los ajustados obtenidos en el paso anterior. Utilizad tipos de línea y colores diferentes para cada conjunto de p-valores.
5. Si utilizamos un valor de FDR igual a 0.05: ¿qué genes declaramos como significativos utilizando la corrección de Benjamini-Hochberg? ¿Y utilizando la corrección de Bonferroni?

6. ¿Todos los significativos según Bonferroni lo son con Benjamini-Hochberg? Comparar los grupos de genes significativos utilizando las funciones `intersect()` y `Biobase::featureNames()`.

Capítulo 9

Expresión diferencial con respuesta continua

Aunque inicialmente el tipo de dato más habitual para análisis de transcripción eran los DNA microarrays con el tiempo no es así. Sin embargo, la metodología de análisis que se ha desarrollado para este tipo de análisis es aplicable, con mínimas variaciones, a cualquier tipo de dato ómico donde la respuesta, la variable de interés, es de tipo continuo, esto es, un número real. De este tipos son los DNA microarrays, los de proteínas, los datos de metilación, etc. Obviamente según la técnica cambia el preprocesado que realizamos pero el tratamieno estadístico posterior es el mismo. En esencia el problema que vamos a abordar consiste en considerar las variables fenotípicas (o metadatos) como las variables predictoras. La variable respuesta para cada sonda (gen, proteína, etc.) serán las expresiones cuantificadas (correspondiendo a una fila de la matriz de expresión). Vamos a ajustar un modelo lineal para cada perfil de expresión. El problema es que el número de observaciones es pequeño (número de muestras que coincide con el número de columnas de la matriz de expresión). Esto supone que la estimación que hacemos de la varianza del error aleatorio es mala. ¿Cómo mejorar esta estimación? Agregando, basándonos en algún modelo estocástico, los estimadores de los distintos ajustes correspondiendo con las distintas sondas. Una idea sencilla pero muy efectiva. Vemos dos procedimientos. Uno de los primeros conocido como método SAM § 9.1 y el más utilizado método limma § 9.2.

9.1 Método SAM

Es la abreviatura de **S**ignificance **A**nalysis of **M**icroarrays. Fue propuesto en [140].[1]

[1] Tiene un gran interés consultar la dirección http://www-stat.stanford.edu/~tibs/SAM/. De hecho, no se puede entender el método tal cual se aplica hoy a partir de la referencia original. Ha habido una gran evolución posterior del procedimiento incluyendo el manejo de datos de secuencia. Existe una versión comercial para su uso con Excel. En la parte de los detalles técnicos del manual de la versión comercial se puede encontrar la explicación detallada del método y, por lo tanto, de las salidas que realizan las funciones. Este manual técnico lo podemos conseguir en http://www-stat.stanford.edu/~tibs/SAM/sam.pdf.

9.1.1 El método

El conjunto de muestras, como es habitual, lo denotaremos como por $\{1, \ldots, n\}$. Empezamos con el caso en que pretendemos comparar dos grupos. Esta información la podemos tener con un vector $y = (y_1, \ldots, y_n)$ donde y_j vale 1 si la muestra j está en el primer grupo y vale 2 si la muestra en esa columna está en el segundo grupo. Para un gen dado las expresiones del grupo 1 (o del grupo 2) corresponden con los valores donde $y_j = 1$ (respectivamente $y_j = 2$). Consideramos $J_g = \{j : y_j = g\}$ con $g = 1, 2$. Tendremos J_1 y J_2 donde J_g son las muestras del grupo g con g= 1,2.[2] Denotamos por n_1 y n_2 los cardinales de J_1 y J_2. Podemos denotar

$$\bar{x}_{ig} = \bar{x}_{i,J_g} = \frac{\sum_{j \in J_g} x_{ij}}{n_g}$$

siendo n_g el número de muestras en J_g. Para el i-ésimo gen consideramos la **diferencia relativa** dada por

$$d_i = \frac{\bar{x}_{i1} - \bar{x}_{i2}}{s_i + s_0} \tag{9.1}$$

donde

$$s_i = \sqrt{a \sum_{g=1}^{2} \sum_{j \in J_g} (x_{ij} - \bar{x}_{i,g})^2} \tag{9.2}$$

y

$$a = \frac{1/n_1 + 1/n_2}{n_1 + n_2}$$

Observemos que el estadístico d_i es el estadístico del contraste de igualdad de medias o simplemente el t-estadístico al que le modificamos el error estándar. ¿Para qué se hace esta modificación? Si el valor s_i es muy pequeño entonces el valor d_i es muy grande. En valores de expresión pequeños esto puede pasar. Los distintos valores d_i no son comparables para los distintos genes (la expresión que utilizan los propios autores es que no son intercambiables). Por ello se le suma un valor s_0 que **estabiliza** la varianza (y nos permite comparar los t-valores entre si). Este valor hay que **estimarlo** o calcularlo a partir de los propios datos (§ 9.1.2). Veamos el procedimiento enumerando los pasos.

1. Calculamos los estadísticos d_i según la ecuación 9.1.

2. Ordenamos de menor a mayor los d_i's observados: $d_{(1)} \leq \ldots \leq d_{(N)}$ de modo que $d_{(i)}$ es el i-ésimo menor.

3. Realizamos B permutaciones aleatorias del vector y que nos indica el grupo de la columna. Para cada permutación calculamos los nuevos valores d_i y los denotamos por $d_i(b)$. Ordenamos, como antes, estos nuevos valores de menor a mayor: $d_{(1)}(b) \leq \ldots \leq d_{(N)}(b)$.

4. Calculamos, para las B permutaciones, el valor medio de estos estadísticos ordenados, es decir, calculamos

$$\bar{d}_{(i)} = \sum_{b=1}^{B} \frac{d_{(i)}(b)}{B}.$$

[2] Notemos que J_1 y J_2 son una partición de $\{1, \ldots, n\}$ de modo que $J_1 \cup J_2 = \{1, \ldots, n\}$ y son disjuntos $J_1 \cap J_2 = \emptyset$.

Bajo la hipótesis nula de que no hay diferencias entre los dos grupos los valores de $d_{(i)}$ y de $\bar{d}_{(i)}$ debieran de ser **similares**.

5. Representamos gráficamente $d_{(i)}$ frente a $\bar{d}_{(i)}$.

6. Para un valor fijo positivo Δ, empezando en el origen y moviéndonos a la derecha determinamos el primer $i = i_1$ tal que $d_{(i)} - \bar{d}_{(i)} > \Delta$. Todos los genes a la derecha de i_1 son llamados **significativamente positivos** (estos genes verifican la desigualdad anterior). Del mismo modo, empezando en el origen y moviéndonos a la izquierda determinamos el primer $i = i_2$ tal que $\bar{d}_{(i)} - d_{(i)} > \Delta$. Todos los genes con la diferencia anterior más a la izquierda de i_2 los llamamos **significativamente negativos**. Para cada Δ definimos un punto de corte superior como

$$u(\Delta) = \min\{d_i : i \text{ es significativamente positivo}\}$$

y

$$l(\Delta) = \max\{d_i : i \text{ es significativamente negativo}\}$$

7. Para distintos valores de Δ, calculamos:

   - El número total de genes significativos (como hemos visto en el paso anterior).
   - En la b-ésima permutación aleatoria hemos obtenido $d_i(b)$ con $i = 1, \ldots, N$. Calculamos el número de genes **falsamente llamados** en la aleatorización b-ésima como

$$c_b = |\{i : d_i(b) > u(\Delta) \text{ o } d_i(b) < l(\Delta)\}$$

   Tendremos los valores c_b con $b = 1, \ldots, B$. Notemos que los valores $d_i(b)$ han sido calculados bajo la hipótesis de no asociación, es cierta la hipótesis nula para cada gen. En consecuencia debieran de estar en el intervalo $[l(\Delta), u(\Delta)]$. Calculamos la mediana y el percentil de orden 0.90 de los valores c_b con $b = 1, \ldots, B$ y los denotamos como $q_{0.5}(c)$ y $q_{0.90}(c)$.

8. Vamos a estimar π_0, la proporción de genes sin expresión diferencial, de genes que no se asocian con la covariable y. En definitiva cuál es la proporción de hipótesis nulas ciertas.

   (a) Calculamos los percentiles de orden 0.25 y 0.75, $q_{0.25}$ y $q_{0.75}$, de todos los valores $d_i(b)$ que hemos obtenido realizando las permutaciones: $\{d_i(s); i = 1, \ldots, N; s = 1, \ldots, B\}$ (tenemos un total de $N \times B$ valores).

   (b) Calculamos

$$\hat{\pi}_0 = \frac{|\{d_i : d_i \in (q_{0.25}, q_{0.75})\}|}{N/2}$$

   donde $\{d_1, \ldots, d_N\}$ son los d valores originales.

   (c) Truncamos el valor de $\hat{\pi}_0$ en 1, es decir, consideramos

$$\hat{\pi}_0 = \min\{\hat{\pi}_0, 1\}.$$

[3]

[3] En la opción en que estamos comparando varios grupos que vemos más adelante los valores de d son todos positivos y se trabaja con los percentiles 0 y 0.5.

9. Multiplicamos la mediana, $q_{0.5}(c)$, y el percentil 0.9, $q_{0.90}(c)$, determinados en el paso 7 por $\hat{\pi}_0$.

10. Elegimos un valor de Δ y determinamos el conjunto de genes significativos.

11. La tasa de falsos positivos FDR es estimada como el cociente entre la mediana, $q_{0.5}(c)$, (o el percentil 0.90, $q_{0.90}(c)$) del número de genes falsamente llamados y el número de genes declarados significativos.

12. El **q-valor** (que aparece en la salida de [**R-samr**]) es la tasa pFDR cuando la lista de genes significativos está compuesta por este gen y todos los genes que son más significativos que este. Se calcula buscando cuál es el valor más pequeño de Δ para el cual este gen es declarado como significativo. Si este valor es $\hat{\Delta}$ entonces la pFDR asociada a este valor es el q-valor asociado al gen.

9.1.2 Cálculo de s_0

La selección de s_0 se hace del siguiente modo:

1. Consideramos $\boldsymbol{s} = (s_1, \ldots, s_N)$ las desviaciones estándar muestrales.

2. Sea $q_\alpha(\boldsymbol{s})$ el percentil de orden α de los valores s_i. Definimos

$$d_i^{(\alpha)} = \frac{\bar{x}_{i1} - \bar{x}_{i2}}{s_i + q_\alpha(\boldsymbol{s})}.$$

3. Consideramos los percentiles $q_{j/100}(\boldsymbol{s})$ con $j = 1, \ldots, 100$.

4. Para cada $\alpha \in \{0, 0.1, \ldots, 1\}$:

   (a) Se calcula

$$v_j = \frac{1}{0.64} MAD(\{d_i^{(\alpha)} : s_i \in [q_{j/100}(\boldsymbol{s}), q_{(j+1)/100}(\boldsymbol{s})]\})$$

   para $j = 1, \ldots, 100$ donde MAD es la mediana de las desviaciones absolutas respecto de la mediana.

   (b) Calculamos el valor $CV(\alpha)$, el coeficiente de variación de los valores v_j.[89]

   (c) Se elige como valor de α: $\hat{\alpha}$ que minimiza $CV(\alpha)$.

   (d) Finalmente tomamos para s_0: $\hat{s}_0 = q_{\hat{\alpha}}(\boldsymbol{s})$.

[89] El coeficiente de variación es el cociente de la media y la desviación estándar.

9.1.3 SAM con samr

Utilizamos los datos golub. Cargamos el paquete [**R-samr**].

```
pacman::p_load("samr")
```

Realizamos el análisis. Tenemos dos grupos y en el tipo de respuesta le indicamos la opción **Two class unpaired**. La variable que indica el grupo ha de tomar valores 1 y 2. Vamos a tomar una tasa de falsamente rechazados o FDR muy pequeña. En concreto, $FDR = 0.001$.

```
data(golub,package="multtest")
fac0 = golub.cl +1
samfit = SAM(golub,fac0,resp.type="Two class unpaired",
    nperms=1000,fdr.output=.001)
```

```
Error in SAM(golub, fac0, resp.type = "Two class unpaired", nperms =
    ↪ 1000, : no se pudo encontrar la función "SAM"
```

¿Qué valor de Δ se ha utilizado?

```
samfit$del
```

```
Error: objeto 'samfit' no encontrado
```

¿Y qué genes encuentra el procedimiento significativo?

```
(sigtabla = samfit$siggenes.table)
```

```
Error: objeto 'samfit' no encontrado
```

Podemos comprobar en la salida anterior que diferencia entre genes **up** y genes **down**. ¿Qué es un gen up en este paquete? Indica asociación positiva entre expresión y la covariable y que nos indica el grupo (para golub recordad que la covariable es golub.cl que codifica ALL como 0 y AML como 1). En definitiva un gen up es que tiene una media mayor cuando la covariable es mayor, es decir, para los datos golub que la media en el grupo AML (valor de golub.cl 1) es mayor que la media en el grupo ALL (valor de golub.cl 0). Como es lógico los genes **down** son lo contrario. También podemos ver, para cada uno de ellos: el valor de d_i, su numerador y denominador, el cociente de las medias y el q-valor.

Tenemos también una ilustración gráfica en donde en abscisas nos muestra $\bar{d}_i$ y en ordenadas d_i. La línea en trazo continuo indica la línea donde $\bar{d}_i = d_i$. Las líneas en trazo discontinuo por arriba y por abajo indican los genes up y los genes down. En la figura 9.1 podemos ver los resultados.

```
plot(samfit)
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'plot': objeto 'samfit
    ↪ ' no encontrado
```

```
pdf
  2
```

Podemos ver también una exploración de posibles valores para Δ con

```
head(samfit$delta.table,n=1)
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'head': objeto 'samfit
    ↪ ' no encontrado
```

figures/mdDE6.png

Figura 9.1: Método SAM utilizando [**R-samr**]. En negro los genes no significativos. En rojo los genes up que corresponden con asociación positiva con la covariable y en verde los que tienen asociación negativa.

9.1.4 SAM con otro tipo de covariables

Acabamos de utilizar este procedimiento en la comparación de dos grupos. Este mismo procedimiento se puede adaptar a los casos en que el vector y (covariable) nos indica la pertenencia de la muestra a más de un grupo o bien es un valor numérico.

El método es el mismo modificando las definiciones de d_i y s_i dadas en las ecuaciones 9.1 y 9.2.

Comparación de más de dos grupos Tenemos K grupos a comparar ($K > 2$) por lo que $y_j \in \{1, \ldots, K\}$. Supongamos que J_k denota los índices de las observaciones en grupo k.[4] Y supongamos que n_k es el cardinal de J_k ($\sum_{k=1}^{K} n_k = n$). Definimos d_i como

$$d_i = \frac{r_i}{s_i + s_0} \tag{9.3}$$

con

$$r_i = \sqrt{\frac{\sum_{k=1}^{K} n_k}{\prod_{k=1}^{K} n_k} \sum_{k=1}^{K} n_k (\bar{x}_{i,J_k} - \bar{x}_{i\cdot})^2} \tag{9.4}$$

y

$$s_i = \sqrt{\frac{1}{\sum_{k=1}^{K}(n_k - 1)} \left(\sum_{k=1}^{K} \frac{1}{n_k} \right) \sum_{k=1}^{K} \sum_{j \in J_k} (x_{ij} - \bar{x}_{i,J_k})^2}. \tag{9.5}$$

Covariable continua Supongamos que los valores de y son valores numéricos en este caso definimos d_i como en 9.3 y los valores de r_i y s_i son

$$r_i = \frac{\sum_{j=1}^{n} y_j (x_{ij} - \bar{x}_{i\cdot})}{\sum_{j=1}^{n} (y_j - \bar{y})^2} \tag{9.6}$$

y

$$s_i = \frac{\hat{\sigma}_i}{\sqrt{\sum_{j=1}^{n} (y_j - \bar{y})^2}} \tag{9.7}$$

donde

$$\hat{\sigma}_i = \sqrt{\frac{\sum_{j=1}^{n} (x_{ij} - \hat{x}_{ij})^2}{n-2}}, \tag{9.8}$$

y

$$\hat{x}_{ij} = \hat{\beta}_{i0} + r_i y_j, \tag{9.9}$$

$$\hat{\beta}_{i0} = \bar{x}_{i\cdot} - r_i \bar{y}. \tag{9.10}$$

9.2 Limma

En § 6 damos una breve introducción a modelos lineales. En los ejemplos que tratamos en § 6 se elige una sola fila de la matriz de expresión. Esta fila constituye la variable respuesta y consideramos como variables predictoras en el modelo lineal que ajustamos las variables fenotípicas.

Hemos estudiado la posible asociación entre el perfil de expresión de un gen y las variables fenotípicas. Para ello hemos seleccionado una sonda y ajustado un modelo lineal. Incluso cuando comparamos dos grupos (caso frente a control, cepa mutante frente a salvaje) estamos ajustando un modelo lineal. Pero tenemos muchas sondas y muchos modelos a ajustar y cada uno de ellos con poca muestra.

El paquete básico en esta sección es [128, limma] siendo sus espléndida viñetas que son, en sí mismas, las referencias básicas junto con el artículo en donde se plantea el procedimiento [129].

[4] Los conjuntos J_k son disjustos entre sí y su unión es el total de muestras, es decir, $\cup_{k=1}^{K} J_k = \{1, \ldots, n\}$.

9.2.1 El modelo

Vamos a ajustar un modelo lineal para cada fila de la matriz de expresión. Con datos de microarrays de DNA asumimos que se ha preprocesado corrigiendo fondo y normalizado y que los datos están en escala logaritmo en base 2 o $\log_2$ de la expresión original. Las expresiones aleatorias correspondientes al gen i-ésimo o perfil de expresión lo denotamos por $\boldsymbol{Y}_i = (Y_{i1}, \ldots, Y_{in})^T$. [90] Los valores observados las denotamos en minúscula: $\boldsymbol{y}_i = (y_{i1}, \ldots, y_{in})^T$. El vector $\boldsymbol{Y}_i$ es el vector de respuestas para el i-ésimo ajuste que vamos a realizar. El vector de medias de las respuestas es $E\boldsymbol{Y}_i = \boldsymbol{\mu}_i = (EY_{i1}, \ldots, EY_{in})^T$ compuesto por las medias de cada una de las variables. Asumimos un modelo lineal. Por ello, asumimos que la media de la j-esima respuesta se relaciona con los predictores (variables fenotípicas) como

[90] $\boldsymbol{Y}^T$ es el vector traspuesto de $\boldsymbol{Y}$.

$$\mu_{ij} = \beta_{i1}x_{j1} + \beta_{ip}x_{jp} = \boldsymbol{x}_j^T\boldsymbol{\beta}_i,$$

siendo $\boldsymbol{x}_j = (x_{j1}, \ldots, x_{jp})^T$ es el vector de variables fenotípicas para la j-ésima muestra y $\boldsymbol{\beta}_i = (\beta_{i1}, \ldots, \beta_{ip})^T$ es el vector de coeficientes. En resumen, $\boldsymbol{x}_j^T\boldsymbol{\beta}_i$ es la media de la respuesta j-ésima. La matriz modelo, $\boldsymbol{X}$, tiene en la fila j el vector $\boldsymbol{x}_j^T$ y posiblemente el primer elemento de $\boldsymbol{x}_j^T$ será el valor uno correspondiendo con la constante. Estamos asumiendo

$$\boldsymbol{\mu}_i = E\boldsymbol{Y}_i = \boldsymbol{X}\boldsymbol{\beta}_i. \tag{9.11}$$

En lo que sigue asumimos también que la matriz $\boldsymbol{X}$ tiene todas las columnas linealmente independientes. **Notemos que la matriz de modelo es común para todas las filas**. Vamos a ajustar un modelo para cada fila en donde tenemos la misma matriz modelo pero nos cambia el vector de respuestas. Suponemos que la matriz de covarianzas del vector aleatorio $\boldsymbol{Y}_i$, $var(\boldsymbol{Y}_i)$, es proporcional (siendo desconocida la constante de proporcionalidad) a una matriz conocida, es decir, asumimos que

$$var(\boldsymbol{Y}_i) = \sigma_i^2\boldsymbol{V}_i, \tag{9.12}$$

donde $\boldsymbol{V}_i$ es una matriz definida positiva **conocida**. Estamos pues en un caso de mínimos cuadrado ponderados (o generalizados) modest18-§ 4.12.

Si tomamos los estimadores de los coeficientes obtenidos por el método de los mínimos cuadrados entonces la matriz de covarianzas de dichos estimadores sería ([6, pág. 69], modest18-§ 4.12)

$$var(\boldsymbol{\beta}_i) = \sigma_i^2(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}.$$

Como es habitual en modelos lineales estamos interesados en ciertos contrastes (entendidos como combinaciones lineales) sobre el vector de coeficientes $\boldsymbol{\beta}_i$. Si consideremos un vector $\boldsymbol{c} \in \mathbb{R}^p$ entendemos por un contraste (hay una cierta confusión en castellano con la expresión contraste de hipótesis) la combinación lineal $\boldsymbol{c}^T\boldsymbol{\beta} = \sum_{j=1}^p c_j\beta_j$. Contrastar un contraste significa que consideramos el contraste de hipótesis $H_0 : \sum_{j=1}^p c_j\beta_j = 0$ vs $H_1 : \sum_{j=1}^p c_j\beta_j \neq 0$. Eligiendo adecuadamente los valores del vector $\boldsymbol{c}$ podemos bien evaluar si los coeficientes son marginalmente nulos (todos valores de $\boldsymbol{c}$ nulos salvo el j-ésimo) o bien en evaluar si una combinación lineal de especial interés de dichos coeficientes es nulo. De hecho, realizaremos distintos contrastes, por ejemplo q, para cada gen o característica que estemos estudiando. Tomamos una matriz

$\boldsymbol{C} \in \mathbb{R}^{p\times q}$ (en cada columna un contraste) de modo que los q contrastes considerados simultáneamente pueden expresarse como

$$\boldsymbol{a}_i = \boldsymbol{C}^T\boldsymbol{\beta}_i$$

donde a_{ij} es el j-ésimo contraste para la i-ésima fila y estamos interesados en contrastar si a_{ij} es nulo para cada j y para cada gen i, es decir, en la hipótesis nula: $H_{ij} : a_{ij} = 0$ vs $K_{ij} : a_{ij} \neq 0$. Ajustamos un modelo lineal para cada fila y obtenemos los estimadores mínimo cuadráticos de los coeficientes, $\hat{\boldsymbol{\beta}}_i$, el estimador insesgado de s_i^2 de σ_i^2 (suma de cuadrados de residuos dividido por el número de observaciones menos el número de parámetros estimados p) y la matriz de covarianzas estimada del vector de coeficientes que, bajo la hipótesis indicada de $\boldsymbol{V}_i$ conocida, es

$$\widehat{var(\hat{\boldsymbol{\beta}}_i)} = s_i^2(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1} \tag{9.13}$$

Los contrastes (combinaciones lineales de coeficientes) estimados son

$$\hat{\boldsymbol{a}}_i = \boldsymbol{C}^T\hat{\boldsymbol{\beta}}_i,$$

La matriz de covarianzas de los contrastes $\hat{\boldsymbol{a}}_i$ es

$$var(\hat{\boldsymbol{a}}_i) = \sigma_i^2\boldsymbol{C}^T(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}\boldsymbol{C},$$

y su matriz de covarianzas estimada es

$$\widehat{var(\hat{\boldsymbol{a}}_i)} = s_i^2\boldsymbol{C}^T(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}\boldsymbol{C}.$$

En el trabajo original se dice que no se asume necesariamente normalidad para $\boldsymbol{Y}_i$ ni tampoco que el ajuste de los modelos lineales se hace mediante el procedimiento de los mínimos cuadrados. Sin embargo, todas las hipótesis distribucionales que siguen se basan en estas hipótesis. En la presentación que hacemos aquí asumimos este procedimiento. Tampoco cuesta tanto asumirlo. Como es bien conocido por la teoría de los modelos lineales normales suponiendo que asumimos las hipótesis del modelo lineal y que los estimadores de los coeficientes son los mínimo cuadráticos y máximo verosímiles entonces:

1. $\hat{\boldsymbol{a}}_i$ tiene una distribución normal multivariante con vector de medias $\boldsymbol{a}_i$ y matriz de covarianzas $\sigma_i^2\boldsymbol{C}^T(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}\boldsymbol{C}$,

$$\hat{\boldsymbol{a}}_i \sim N_q(\boldsymbol{a}_i, \sigma_i^2\sigma_i^2\boldsymbol{C}^T(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}\boldsymbol{C}).$$

2. También asumimos que $1/\sigma_i^2$ siguen aproximadamente una distribución ji-cuadrado escalada. [5]

La matriz $\boldsymbol{V}_i$ podría depender de $\boldsymbol{\beta}_i$. Si esta dependencia se produce entonces consideramos $\boldsymbol{V}_i$ evaluada en $\hat{\boldsymbol{\beta}}_i$ y la dependencia puede ser ignorada. Si denotamos por $v_{jj}^{(i)}$ el j-ésimo elemento de la diagonal principal de $\boldsymbol{C}^T(\boldsymbol{X}^T\boldsymbol{V}_i^{-1}\boldsymbol{X})^{-1}\boldsymbol{C}$ entonces todas las hipótesis asumidas en este método son

[5] Este es el punto clave en lo que sigue ya que los autores adoptan una aproximación bayesiana. No consideran el parámetro desconocido σ_i^2 como algo fijo sino como un valor aleatorio lo que constituye la esencia de las técnicas bayesianas que lo estimarán combinando una distribución a priori sobre ese parámetro aleatorio con la verosimilitud para conseguir una distribución a posteriori que utilizará para estimar el parámetro.

1. $\hat{a}_{ij}|a_{ij}, \sigma_i^2 \sim N(a_{ij}, \sigma_i^2 v_{jj}^{(i)})$.

2. $s_i^2|\sigma_i^2 \sim \frac{\sigma_i^2}{d_i}\chi_{d_i}^2$ siendo d_i los grados de libertad de la suma de cuadrados residual del modelo ajustado para el i-ésimo gen.

Bajo la hipótesis nula de que el contraste a_{ij} es nulo entonces

$$\tilde{t}_{ij} = \frac{\hat{a}_{ij}}{s_i\sqrt{v_{jj}^{(i)}}}$$

aproximadamente tiene una distribución t de Student con d_i grados de libertad.

En lo que sigue se **asumirá** (y no tienen porqué ser cierto) que $\hat{\boldsymbol{a}}_i$ y s_i^2 son independientes para las distintas filas de la matriz de expresión (genes, sondas, etc.).

Tenemos muchos test simultáneos. Se trata de modelizar el comportamiento entre genes. La opción que se elige es suponer una distribución de probabilidad sobre la varianza σ_i^2, esto es, adoptamos una aproximación bayesiana al problema que viene dada por

$$\frac{1}{\sigma_i^2} \sim \frac{1}{d_0 s_0^2}\chi_{d_0}^2,$$

esto es, que tiene una distribución proporcional a una distribución ji-cuadrado. Intervienen dos parámetros en la distribución indicada d_0 y s_0 que son desconocidos y habrá que estimar.

Para un j dado, suponemos que el coeficiente β_{ij} es no nulo con una probabilidad conocida que no depende de i y, por lo tanto, describe la fracción coeficientes no nulos entre el conjunto total de genes,

$$P(\beta_{ij} \neq 0) = p_j.$$

Notemos que p_j tiene el sentido de la proporción esperada de genes que se expresan diferencialmente. Para los coeficientes no nulos se asume adicionalmente que

$$\beta_{ij}|\sigma_i^2, \beta_{ij} \neq 0 \sim N(0, v_{0j}\sigma_i^2).$$

Admitimos que el coeficiente es no nulo pero el valor real se mueve alrededor de cero con varianza $v_{0j}\sigma_i^2$.

Asumiendo el modelo que acabamos de especificar se tiene que la media de la distribución a posteriori de $1/\sigma_i^2$ condicionada a s_i^2 viene dada por

$$E\left[\frac{1}{\sigma_i^2}|s_i^2\right] = \frac{1}{\tilde{s}_i^2},$$

con

$$\tilde{s}_i^2 = \frac{d_0 s_0^2 + d_i s_i^2}{d_0 + d_i}.$$

Podemos definir el estadístico t moderado (Moderated t-statistic) como

$$\tilde{t}_{ij} = \frac{\hat{\beta}_{ij}}{\tilde{s}_i^2\sqrt{v_{jj}^{(i)}}}. \tag{9.14}$$

Se demuestra que los t-estadísticos moderados $\tilde{t}_{ij}$ y las varianzas muestrales residuales s_i^2 se distribuyen independientemente. Y de hecho, tendremos que bajo la hipótesis nula $H_0 : \beta_{ij} = 0$, el t-estadístico moderado

$\tilde{t}_{ij}$ sigue una distribución t de Student con d_i+d_0 grados de libertad ([129, sección 4]). Los grados de libertad que estamos añadiendo d_0 expresan la ganancia de información que obtenemos de utilizar todos los genes siempre **asumiendo el modelo jerárquico** indicado previamente. Los valores d_0 y s_0 que se suponen conocidos en lo previo serán estimados a partir de los datos.

En [129, sección 4] se demuestra que s_i^2 y $\tilde{t}_{ij}$ son independientes y además sus distribuciones marginales son las siguientes:

$$s_i^2 \sim s_0^2 F_{d_i,d_0},$$

siendo $F_{u,v}$ una distribución F de Fisher con u y v grados de libertad. También prueba que

$$\tilde{t}_{ij}|\beta_{ij} = 0 \sim t_{d_0+d_i},$$

con t_u una t de Student con u grados de libertad. Considera los odds a posteriori de que el coeficiente β_{ij} no sea nulo frente a que lo sea. Estos odds son calculados condicionados a los valores de los estadísticos $\tilde{t}_{ij}$ y s_i^2.

$$O_{ij} = \frac{P(\beta_{ij} \neq 0|\tilde{t}_{ij}, s_i^2)}{P(\beta_{ij} = 0|\tilde{t}_{ij}, s_i^2)}. \tag{9.15}$$

Utilizando la independencia de $\tilde{t}_{ij}$ y s_i^2 y las distribuciones marginales calculadas previamente llega a la siguiente expresión para los odds a posteriori.

$$O_{ij} = \frac{p_j}{1-p_j}\left(\frac{v_{jj}^{(i)}}{v_{jj}^{i}+v_{0j}}\right)^{1/2}\left(\frac{\tilde{t}_{ij}^2+d_0+d_i}{\tilde{t}_{ij}^2\frac{v_{jj}^{(i)}}{v_{jj}^{(i)}+v_{oj}}+d_o+d_i}\right)^{(1-d_0+d_i)/2} \tag{9.16}$$

Finalmente los autores proponen trabajar con el logaritmo natural del cociente de odds a posteriori.

$$B_{ij} = \ln O_{ij}. \tag{9.17}$$

De facto, aunque la salida del paquete proporciona estos odds a posteriori es habitual trabajar con los p-valores asociados a los contrastes algo no habitual en la metodología bayesiana.

En el procedimiento que acabamos de comentar falta estimar los parámetros de las distribuciones a priori. Hemos de estimar d_0, s_0, v_{0j} (para todos los j) y las proporciones p_j de contrastes no nulos.

En principio, son valores que el propio usuario debiera de especificar. Representan la información previa y por lo tanto es la información que a priori de la que se dispone y sin usar nada de la información muestral debieran de ser especificados. Es claro que esto no parece muy factible. Los autores proponen lo que se conoce como un método empírico bayesiano y utilizan los propios datos para determinar estos valores.

En [129, sección 6.2] vemos cómo utilizando las varianzas estimadas s_i^2 podemos estimar los valores de d_0 y s_0. Esencialmente utiliza un método de los momentos estudiando la distribución de $z_i = \ln s_i^2$. Utilizando media y varianza de los valores z_i propone un procedimiento que estima d_0 y s_0.

En [129, sección 6.3] se estudia la estimación de v_{0j} para cada contraste. Se utilizan los valores de los estadísticos $\tilde{t}_i$ cuya distribución es conocida cuando el contraste es nulo y cuando no lo es. Igualan la función de distribución teórica (que depende de d_0 previamente estimados)

a la muestral. Finalmente la estimación de p_j o proporción de contrastes no nulos se hacen utilizando las probabilidades estimadas de que el gen i tenga un contraste significativo.

Se comenta que la estimación de v_{0j} y de p_j son poco robustas a diferencia de los hiperparámetros d_0 y s_0 lo que hace más recomendable el uso de los t-estadísticos moderados (9.14) que de los B-estadísticos (9.17).

Es claro que los log-odds tienen una interpretación sencilla y, por ello, son útiles. Sin embargo, para decidir qué genes son significativos para el contraste en que estemos interesados es mejor opción trabajar con los t-estadísticos moderados cuya definición es una especie de mezcla inteligente de Estadística clásica y bayesiana.

Lo más importante, sin duda,el t-estadístico moderado tiene la ventaja sobre el t-estadístico usual de que es menos probable que aparezcan valores muy grandes simplemente porque las varianzas muestrales infraestiman las varianzas poblaciones. Trabajamos con pequeñas muestras lo que hace que la estimación de la varianza sea poco precisa. Una infraestimación notable de esta varianza ocasiona que el estadístico sea muy grande aunque no tengamos una diferencia de medias grande. Este mismo problema se comenta en § 9.1. Allí se proponía añadir una constante (estimada a partir de los propios datos) a la desviación estándar de la diferencia de medias (el error estándar). ¿Esto es lo mismo? Es similar pero no es lo mismo. Si los grados de libertad verifican $d_0 < +\infty$ y $d_i > 0$ entonces los t-estadísticos moderados tienen la siguiente expresión

$$\tilde{t}_{ij} = \left(\frac{d_0 + d_i}{d_i}\right)^{1/2} \frac{\hat{\beta}_{ij}}{\sqrt{s_{*,i}^2 v_{ij}}},$$

siendo

$$s_{*,i}^2 = s_i^2 + \frac{d_0}{d_i} s_0^2.$$

Cada varianza muestral es incrementada un valor positivo en principio distinto para cada gen. La idea que vimos en § 9.1 consistía en considerar

$$t_{*,ij} = \frac{\hat{\beta}_{ij}}{(s_i + a)\sqrt{v_{ij}}}$$

donde a es un valor positivo determinado por un método ad-hoc. En este método lo que se incrementa es la desviación estándar y no la varianza. El t-estadístico moderado incrementa la varianza. Además el incremento puede ser distinto para cada gen.

¿Y si queremos contrastar más de un contraste al mismo tiempo? En lugar de t-estadísticos moderados tendremos F-estadísticos moderados. Se comprueba que los F-estadísticos moderados no son más que los F-estadísticos usuales en los cuales sustituimos las varianzas s_i^2 por $\tilde{s}_i^2$. Además su distribución es $F_i \sim F_{r,d_0+d_i}$, siendo r el número de contrastes linealmente independientes. Vemos que los grados de libertad del denominador también se incrementan. En las siguientes secciones vamos a considerar distintos diseños experimentales y los analizaremos con [128, limma], un paquete fundamental en el proyecto Bioconductor y de los más antiguos.

9.2.2 Limma aplicado a gse25171

Este ejemplo es utilizado en § 6 para ilustrar cómo ajustar un modelo lineal. Hacemos una presentación más ligada al desarrollo teórico del

modelo limma. Pretendemos evaluar cómo influye la presencia de fosfatos en la arabidopsis. Leemos los datos `tamidata2::gse25171`.

```
pacman::p_load(Biobase)
data(gse25171,package="tamidata2")
```

Nuestras variables fenotípicas son

```
head(pData(gse25171),n=2)
```

```
               time time2 Pi replication
GSM618324.CEL.gz 0 Short Treatment 1
GSM618325.CEL.gz 0 Short Control 2
```

Vamos a plantearnos cómo pueden influir el tiempo de observación de la muestra (`time` y la presencia o no de fósforo `Pi` en la expresión de cada una de las sondas que tenemos en el microarray. Con objeto de ilustrar vamos a ir considerando modelos cada vez más complejos. Empezamos con un modelo que considera simplemente el tiempo. Para ello hemos de construir previamente la matriz de modelo.

```
design = model.matrix(~ pData(gse25171)[,"time"])
```

Si vemos las primeras filas de dicha matriz.

```
head(design,n=3)
```

```
  (Intercept) pData(gse25171)[, "time"]
1 1 0
2 1 0
3 1 1
```

podemos ver que la primera columna nos da la constante del modelo mientras que la segunda columna nos da el tiempo de observación. Podemos ver que, a diferencia del uso habitual de `lm()` no indicamos la variable respuesta ya que cada fila será la variable respuesta en cada uno de los ajustes lineales que aplicamos. Cambiamos los nombres de la columnas por razones estéticas.

```
colnames(design) = c("intercept","time")
```

Vamos a ajustar todos los modelos lineales que podemos construir utilizando siempre las mismas variables predictoras (tenemos una misma matriz de modelo previamente construida) y donde la variable respuesta va cambiando correspondiendo a cada una de las filas de la matriz de expresión. La función está preparada para trabajar con `ExpressionSet`.

```
fit = limma::lmFit(gse25171,design)
```

Podemos el vector de coeficientes para cada uno de los ajustes con cualquiera de las siguientes opciones (vemos que están definidos los métodos `coefficients` y `coef`).

```
fit$coefficients
coef(fit)
coefficients(fit)
```

Vemos los primeros coeficientes y errores estándar estimados.

```
head(coef(fit))
```

```
          intercept time
244919_at 5.027965 -0.0043155130
244920_s_at 8.328613 -0.0049779961
244922_s_at 4.214618 0.0060266161
244927_at 5.514228 0.0006156454
244954_s_at 3.683001 -0.0060599794
244959_s_at 6.121318 0.0014638159
```

```
head(fit$sigma)
```

```
  244919_at 244920_s_at 244922_s_at 244927_at
  0.2015610 0.2993295 0.1940182 0.1622302
244954_s_at 244959_s_at
  0.1855284 0.3711833
```

Es interesante ver un estimador de densidad (figura 9.2) de los distintos errores estándar observados, los valores s_i definidos arriba.

```
pacman::p_load(ggplot2)
df = data.frame(sigma = fit$sigma^2)
p = ggplot(df,aes(x=sigma)) + geom_density()
```

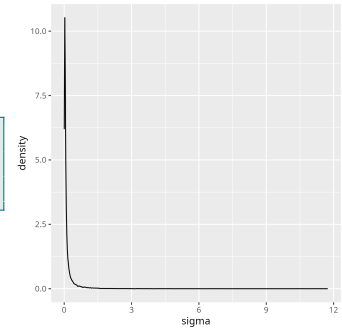


Figura 9.2: Errores estándar estimados para las distintas sondas.

Realmente la hipótesis distribucional era que $1/\sigma_i^2$ multiplicada por una constante sigue una distribución ji-cuadrado. En la figura 9.3 mostramos un estimador de la densidad de $1/s_i^2$.

```
pacman::p_load(ggplot2)
df = data.frame(sigma = 1/fit$sigma^2)
p = ggplot(df,aes(x=sigma)) + geom_density()
```

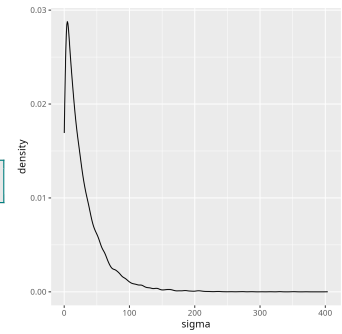


Figura 9.3: Estimador de la densidad de $1/s_i^2$ siendo s_i^2 los errores estándar estimados para las distintas sondas.

En principio, podríamos simplemente contrastar, sin utilizar el modelo limma, si cada uno de los coeficientes estimados para el predictor `time` puede ser considerado nulo como hacemos en § 6.2 ya que realmente estamos realizando un ajuste de regresión lineal simple para cada sonda.

```
t0 = coef(fit)[,2] / fit$sigma
```

Estos valores tienen, aproximadamente, una distribución `t` de Student con $n-2$ grados de libertad. Si nos fijamos en el primer coeficiente, el p-valor correspondiente al test de si podemos considerar dicho coeficiente nulo sería dos veces el área de la cola (la distribución t es simétrica). Lo tenemos con

```
2*(1-pt(abs(t0[1]),df = ncol(gse25171)-2))
```

```
244919_at
0.9831112
```

Todos los p-valores los calculamos con

```
p0 = 2*(1-pt(abs(t0),df = ncol(gse25171)-2))
```

Un diagrama de densidad no viene mal.

```
pacman::p_load(ggplot2)
df = data.frame(p0)
p = ggplot(df,aes(x=p0)) + geom_density()
```

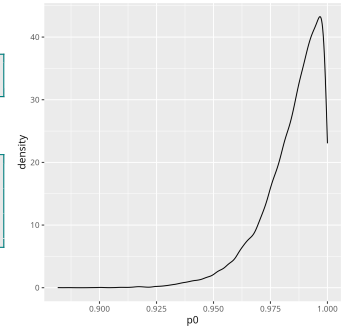


Figura 9.4: p-valores correspondiente al coeficiente del tiempo en un modelo que considera la constante y dicha variable. Todos son p-valores muy grandes.

Podemos ver en la figura 9.4 que son p-valores muy grandes. ¿Cuántos son menores que 0.05?

```
table(p0 <0.05)
```

```
FALSE
20773
```

Como vemos ninguno lo es. No tenemos ningún test significativo. Aplicamos ahora el método limma.

```
fit1 = limma::eBayes(fit)
```

Tenemos los t-estadísticos moderados con

```
head(fit1$t,n=3)
```

```
          intercept time
244919_at 95.59784 -1.0157539
244920_s_at 109.20012 -0.8079886
244922_s_at 82.96958 1.4687043
```

Los p-valores con

```
head(fit1$p.value,n=3)
```

```
             intercept time
244919_at 1.026170e-32 0.3197956
244920_s_at 4.133080e-34 0.4269692
244922_s_at 3.131542e-31 0.1547937
```

Los valores B o logaritmo natural del cociente de odds a posteriori.

```
head(fit1$lods,n=3)
```

```
          intercept time
244919_at 65.09013 -7.262310
244920_s_at 68.21524 -7.452630
244922_s_at 61.71137 -6.713511
```

El valor estimado de s_o^2 y de d_0.

```
fit1$s2.prior
```

```
[1] 0.03779788
```

```
fit1$df.prior
```

```
[1] 2.198413
```

Tenemos los estimadores de los errores estándar a posteriori.

```
head(fit1$s2.post)
```

```
  244919_at 244920_s_at 244922_s_at 244927_at
 0.04036981 0.08489214 0.03765713 0.02736151
244954_s_at 244959_s_at
 0.03472760 0.12869397
```

La función `limma::topTable()` nos hace un resumen de los ajustes en donde ordena (podemos indicar que no lo haga con distintos criterios o que no lo haga con el argumento **`sort.by`**) las sondas por su p-valor original o, equivalentemente, por el p-valor ajustado.

```
limma::topTable(fit1,coef=2,adjust ="BH",number = 2)
```

```
          PROBEID ENTREZID GO EVIDENCE
261892_at 261892_at 844423 GO:0000976 IDA
246253_at 246253_at 829880 GO:0000976 IPI
         ONTOLOGY TAIR logFC AveExpr
261892_at MF AT1G80840 -0.13307030 6.937141
246253_at MF AT4G37260 -0.04069718 8.829745
                t P.Value adj.P.Val
261892_at -7.451575 1.028302e-07 0.002136091
246253_at -6.993712 2.985545e-07 0.003100936
               B
261892_at 7.185741
246253_at 6.100381
```

Vemos en la salida anterior los descriptores que hemos incluido en el slot `ExpressionSet::fData`. Tenemos el coeficiente estimado en el ajuste (`logFC`)[91], luego nos incluye la expresión media (en logaritmo en base 2), el valor del estadístico del test de la t moderado, el p-valor original, el p-valor ajustado según el método que elijamos (por defecto, Benjamini-Hochberg) y el valor estimado del logaritmo de los odds a posteriori (9.17). ¿Cuántas sondas tienen un coeficiente significativamente no nulo?

[91] De verdad que esto no induce más a confusión.

```
padj = limma::topTable(fit1,coef=2,adjust ="BH",
                       number=nrow(gse25171))[,"adj.P.Val"]
table(padj < 0.05)
```

```
FALSE TRUE
20726 47
```

Supongamos que ahora consideramos un modelo en que incorporamos la variable `Pi` de caracter categórico (binario). Consideramos un modelo con los efectos principales.

```
design = model.matrix(~ pData(gse25171)[,"time"] +
                        pData(gse25171)[,"Pi"])
colnames(design) = c("constante","time","Pi")
head(design)
```

```
  constante time Pi
1 1 0 1
2 1 0 0
3 1 1 1
4 1 1 0
5 1 6 1
6 1 6 0
```

En la matriz de modelo hemos añadido una columna en donde recogemos la presencia (1) o no (0) de fosfatos. Ajustamos los modelos.

```
fit = limma::lmFit(gse25171,design)
fit1 = limma::eBayes(fit)
```

Ahora tenemos tres coeficientes correspondiendo con cada una de las columnas de la matriz de modelo y podemos evaluar aquellas sondas para las cuales el correspondiente coeficiente es significativamente no nulo. Observemos que no ordenamos (`sort.by="none"`) ni seleccionamos (`number=nrow(gse25171)`) las sondas. Primero evaluamos si los coeficientes correspondientes a la columna 2 (`coef=2` son nulos. Notemos que la columna 2 corresponde con la variable `time`.

```
tt2 = limma::topTable(fit1,coef="time",adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
```

Ahora evaluamos si los coeficientes correspondientes a la columna 3 (`coef =3` son nulos. Notemos que la columna 3 corresponde con la variable `Pi`. Podemos poner el número de la columna o su nombre. Elegimos, por seguridad, indicar el nombre.

```
tt3 = limma::topTable(fit1,coef="Pi",adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
```

¿Cualés y cuántas sondas tienen el coeficiente correspondiente a `time` significativamente no nulo ajustando por el método de Benjamini-Hochberg?

```
tt2.row = which(tt2[,"adj.P.Val"] < .05)
length(tt2.row)
```

```
[1] 154
```

Es importante notar que el modelo que tenemos ahora no es el mismo que cuando tenemos solamente `time` y por ello los resultados no tienen porqué ser los mismos. Dicho con propiedad, estamos ajustando por las variables `time` y `Pi`. Y ahora: ¿Cualés y cuántas sondas tienen el coeficiente correspondiente a `Pi` significativamente no nulo ajustando por el método de Benjamini-Hochberg?

```
tt3.row = which(tt3[,"adj.P.Val"] < .05)
length(tt3.row)
```

```
[1] 41
```

¿Y cuántas sondas tienen coeficiente significativamente no nulo tanto para una como para la otra variable?

```
intersect(tt2.row,tt3.row)
```

```
 [1] 909 1943 1974 2672 3502 5316 6254
 [8] 7164 7746 8582 11704 12686 13731 14932
[15] 14978 15639 15683 18710 19667 20498
```

¿Qué sondas son?

```
tt2[intersect(tt2.row,tt3.row),"PROBEID"]
```

¿A qué genes corresponden utilizando su código `ENTREZID`?

```
tt2[intersect(tt2.row,tt3.row),"ENTREZID"]
```

Puede ser una buena idea realizar un diagrama de Venn en donde recojamos el número de genes significativos en las distintas comparaciones. Vamos a utilizar el paquete [34, VennDiagram].

```
pacman::p_load(ggVennDiagram)
x = list(time = (tt2[,"adj.P.Val"] < .1),
         Pi = (tt3[,"adj.P.Val"] < .1))
p = ggVennDiagram(x)
ggsave(paste0(dirTamiFigures,"timepi.png"),p)
```

Podemos comprobar que son coincidentes los dos grupos.

¿Hay interacción entre las variables predictoras `time` y `Pi`? Observemos que sustituimos + por `*`.

```
design = model.matrix(~ pData(gse25171)[,"time"] *
                        pData(gse25171)[,"Pi"])
colnames(design) = c("intercept","time","Pi","time:Pi")
head(design,n=3)
```

```
  intercept time Pi time:Pi
1 1 0 1 0
2 1 0 0 0
3 1 1 1 1
```

La nueva columna de la matriz de modelo se define como el producto de las columnas 2 y 3 correspondientes a las variables originales `time` y `Pi`. El modelo es distinto, los coeficientes estimados para la nueva variable y para las ya existentes son distintos y no necesariamente tendremos las mismas sondas. Este sería el modelo correcto puesto que incorpora la posible interacción. ¿Qué sondas tienen interacciones significativas?

```
fit = limma::lmFit(gse25171,design)
fit1 = limma::eBayes(fit)
tt2 = limma::topTable(fit1,coef=2,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt3 = limma::topTable(fit1,coef=3,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt4 = limma::topTable(fit1,coef=4,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
```

Ahora podemos evaluar los que tienen interacciones significativas.

```
tt4.row = which(tt4["adj.P.Val"] < .05)
tt4[tt4.row,c("PROBEID","ENTREZID")]
```

```
          PROBEID ENTREZID
246071_at 246071_at 832137
246576_at 246576_at 840052
248545_at 248545_at 835091
253386_at 253386_at 829440
266132_at 266132_at 819120
267611_at 267611_at 817207
```

Cuando una sonda tiene una interacción positiva se suele asumir que los efectos principales no se deben eliminar del modelo. Si hay interacción quiere decir que influyen significativamente y además de un modo distinto.

Hemos utilizado la variable `time` como numérica (y lo es). Sin embargo, podemos ver que en las variables fenotípicas hay una variable `time2` categórica con dos niveles.

```
pData(gse25171)["time2"]
```

```
 [1] Short Short Short Short Medium Medium
 [7] Medium Medium Short Short Short Short
[13] Medium Medium Medium Medium Short Short
[19] Short Short Medium Medium Medium Medium
Levels: Short Medium
```

Vamos a realizar un análisis en donde se considera en lugar del tiempo original esta nueva variable. Ajustamos directamente el modelo con las dos variables categóricas, `time2` y `Pi` y con la posible interacción.

```
design = model.matrix(~ pData(gse25171)["time2"] *
                        pData(gse25171)["Pi"])
colnames(design) = c("constante","time2","Pi","time2:Pi")
head(design,n=5)
```

```
  constante time2 Pi time2:Pi
1 1 0 1 0
2 1 0 0 0
3 1 0 1 0
4 1 0 0 0
5 1 1 1 1
```

Podemos repetir el análisis previo en el cual analizábamos la interacción.

```
fit = limma::lmFit(gse25171,design)
fit1 = limma::eBayes(fit)
tt2 = limma::topTable(fit1,coef=2,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt3 = limma::topTable(fit1,coef=3,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt4 = limma::topTable(fit1,coef=4,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt4.row = which(tt4[,"adj.P.Val"] < .05)
head(tt4[tt4.row,c("PROBEID","ENTREZID")],n=3)
```

```
          PROBEID ENTREZID
245571_at 245571_at 827120
245579_at 245579_at 827137
247477_at 247477_at 836355
```

Podemos ir evaluando aquellos que son significativos en la interacción (y consecuentemente asumimos que los efectos principales de cada variable también lo son). Podemos evaluar aquellas sondas que no tienen interacción significativa pero sí que significativa alguna de los efectos principales. Por ejemplo, no hay interacción significativa pero sí lo es el efecto de `time2`.

```
sel = which(tt4[,"adj.P.Val"] >.05 & tt2[,"adj.P.Val"] < .05)
length(sel)
head(tt4[sel,c("PROBEID","ENTREZID")],n=3)
```

No hay interacción significativa pero sí lo es el efecto de `Pi`.

```
sel = which(tt4[,"adj.P.Val"] >.05 & tt3[,"adj.P.Val"] < .05)
length(sel)
```

```
[1] 0
```

Esta forma de plantear el análisis supone que sabemos interpretar que la presencia de la constante hace que cada uno de los coeficientes se ha de interpretar como la modificación respecto de un grupo control que en este caso sería cuando las variables categóricas toman el primer nivel.

```
levels(pData(gse25171)[,"time2"])
```

```
[1] "Short" "Medium"
```

```
levels(pData(gse25171)[,"Pi"])
```

```
[1] "Control" "Treatment"
```

Por tanto, el grupo de referencia corresponde a la situación en que `time2` es `Short` y `Pi` es `Control`.

Vamos a adoptar una aproximación basada en contrastes, esto es, combinaciones lineales de los coeficientes para realizar el mismo análisis. Primero vamos a construir una variable categórica que combina ambas variables categóricas.

```
time2Pi = vector("list",ncol(gse25171))
for(i in seq_along(time2Pi))
  time2Pi[[i]] = paste0(pData(gse25171)[,"time2"][i],
                        pData(gse25171)[,"Pi"][i])
(time2Pi = factor(unlist(time2Pi)))
```

```
 [1] ShortTreatment ShortControl
 [3] ShortTreatment ShortControl
 [5] MediumTreatment MediumControl
 [7] MediumTreatment MediumControl
 [9] ShortTreatment ShortControl
[11] ShortTreatment ShortControl
[13] MediumTreatment MediumControl
[15] MediumTreatment MediumControl
[17] ShortTreatment ShortControl
[19] ShortTreatment ShortControl
[21] MediumTreatment MediumControl
[23] MediumTreatment MediumControl
4 Levels: MediumControl ... ShortTreatment
```

Ahora vamos a ajustar un modelo en donde no vamos a incorporar una constante sino que las variables predictoras nos van a indicar cada una de las categorías que tenemos.

```
design = model.matrix(~ 0 + time2Pi) ## Quitamos constante (también -1)
colnames(design) = levels(time2Pi)
head(design)
```

```
  MediumControl MediumTreatment ShortControl
1 0 0 0
2 0 0 1
3 0 0 0
4 0 0 1
5 0 1 0
6 1 0 0
  ShortTreatment
1 1
2 0
3 1
4 0
5 0
6 0
```

Ahora para evaluar los efectos de los dos factores experimentales hemos de construir los contrastes. Todo el planteamiento que hemos visto del método limma se formulaba directamente para contrastes. Ajustamos los modelos lineales.

```
fit = limma::lmFit(gse25171,design)
```

Construimos los contrastes en que estamos interesados.[92] Los nombre indican claramente lo que estamos evaluando.

[92] Un detalle: los nombres de los niveles no pueden empezar por un número.

```
cont.matrix = limma::makeContrasts(
    dif1 = (MediumControl + MediumTreatment)−
        (ShortControl + ShortTreatment),
    dif2 = (MediumControl + ShortControl)−
        (MediumTreatment + ShortTreatment),
    dif3 = (MediumControl − ShortControl),
    dif4 = (MediumTreatment − ShortTreatment),
    dif5 = (MediumControl − ShortControl) −
        (MediumTreatment − ShortTreatment),
    levels = design)
```

Hemos construido la siguiente matriz de contrastes.

```
cont.matrix
```

```
               Contrasts
Levels dif1 dif2 dif3 dif4 dif5
  MediumControl 1 1 1 0 1
```

```
 MediumTreatment 1 -1 0 1 -1
 ShortControl -1 1 -1 0 -1
 ShortTreatment -1 -1 0 -1 1
```

```
fit2 = limma::contrasts.fit(fit,cont.matrix)
fit2 = limma::eBayes(fit2)
```

Ahora podemos evaluar de un modo análogo a lo previo lo que es significativo para cada uno de los contrastes. Cada uno de los coeficientes corresponde ahora a uno de los contrastes en el orden que ocupan en la matriz de contrastes.

```
tt1 = limma::topTable(fit2,coef=1,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt2 = limma::topTable(fit2,coef=2,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt3 = limma::topTable(fit2,coef=3,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt4 = limma::topTable(fit2,coef=4,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
tt5 = limma::topTable(fit2,coef=1,adjust ="BH",
                      sort.by="none",number=nrow(gse25171))
```

Del mismo modo que antes podemos buscar sondas que sean significativas para cada uno de los contrastes. Nos fijamos como ilustración en el contraste `dif5` que evalua la interacción.

```
sel = which(tt5[,"adj.P.Val"] < .05)
length(sel)
```

```
[1] 1151
```

```
head(tt4[sel,c("PROBEID","ENTREZID")],n=3)
```

```
          PROBEID ENTREZID
245052_at 245052_at 817184
245076_at 245076_at 816849
245084_at 245084_at 816861
```

Notemos que al tener una única predictora que es categórica podemos tener interés en contrastar si las medias en los cuatro grupos que define la variable `time2Pi` son iguales. Tenemos un modelo de análisis de la varianza. Vamos a definir la matriz de modelo pero con la constante.

```
design = model.matrix(~ time2Pi)
colnames(design) = c("intercept",levels(time2Pi)[-1])
head(design)
```

```
  intercept MediumTreatment ShortControl
1 1 0 0
2 1 0 1
3 1 0 0
4 1 0 1
5 1 1 0
6 1 0 0
  ShortTreatment
1 1
2 0
3 1
4 0
5 0
6 0
```

Ajustamos los modelos y vemos los resultados sin ordenar en las dos primeras filas.

```
fit = limma::lmFit(gse25171,design)
fit = limma::eBayes(fit)
tt = limma::topTable(fit,coef=2:4,adjust ="BH",
                     sort.by="none",number=nrow(gse25171))
head(tt,n=2)
```

```
             PROBEID ENTREZID GO
244919_at 244919_at 3767772 GO:0005739
244920_s_at 244920_s_at 815347 GO:0005739
          EVIDENCE ONTOLOGY TAIR
244919_at ISM CC AT2G07768
244920_s_at ISM CC AT2G07751
          MediumTreatment ShortControl
244919_at 0.02320866 0.1856609
244920_s_at 0.14629369 0.3238749
          ShortTreatment AveExpr F
244919_at -0.070848387 4.994520 1.957237
244920_s_at 0.006101191 8.290033 1.837229
            P.Value adj.P.Val
244919_at 0.1495647 0.3876365
244920_s_at 0.1695399 0.4156209
```

¿Cuántos genes tienen p-valor significativo?

```
table(tt$adj.P.Val < .05)
```

```

FALSE TRUE
19117 1656
```

9.2.3 Limma aplicado a gse44456

Leemos los datos `tamidata::gse44456`. Se trata de evaluar el efecto del alcohol sobre el hipocampo. En https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE44456 tenemos información adicional del problema tratado.

```
pacman::p_load(Biobase,limma)
data(gse44456,package="tamidata")
```

Tenemos las siguientes variables fenotípicas para analizar su influencia en la expresión de las sondas.

```
names(pData(gse44456))
```

```
[1] "case" "gender"
[3] "age" "cirrhosis"
[5] "smoker" "postmortenint"
[7] "pH" "batch"
```

Con un predictor numérico

Vamos a considerar la variable que nos da el tiempo (en horas) desde el fallecimiento hasta la toma de la muestra.

```
summary(pData(gse44456)[,"postmortenint"])
```

Determinamos la matriz de modelo.

```
design = model.matrix(~ pData(gse44456)[,"postmortenint"])
colnames(design) = c("intercept","postmortenint")
head(design,n=2)
```

```
  intercept postmortenint
1 1 16.75
2 1 27.00
```

Vemos que en la matriz tiene la columna de unos correspondiente a la constante y la columna correspondiente a la variable predictora considerada que es numérica. Ajustamos los modelos lineales.

```
fit = limma::lmFit(gse44456,design)
```

Hemos realizado una regresión lineal simple para cada una de las sondas. Podemos ver los dos coeficientes (constante y el coeficiente del predictor `postmortenint` para cada uno de los ajustes.

```
head(coefficients(fit),n=2)
```

```
       intercept postmortenint
7892501 2.955131 0.001088853
7892502 4.283626 0.003912042
```

Los errores estándar estimados serían

```
head(fit$sigma)
```

```
  7892501 7892502 7892503 7892504 7892505
0.5416383 0.5042429 0.6797924 0.4761044 0.3454714
  7892506
0.5016761
```

```
fit1 = limma::eBayes(fit)
```

Si la constante por la que multiplicamos `postmortenint` es nula indicará que no hay una dependencia del nivel de expresión respecto de la tasa de crecimiento celular. Veamos los resultados.

```
topTable(fit1,coef=2,adjust ="BH",number=3)
```

```
            logFC AveExpr t
7898750 -0.01150852 6.747570 -6.443278
8170468 0.01311588 6.012464 5.689956
7969651 -0.01098933 7.051056 -5.687741
           P.Value adj.P.Val B
7898750 1.016754e-07 0.003385485 6.398649
8170468 1.196673e-06 0.013378383 3.937214
7969651 1.205368e-06 0.013378383 3.930008
```

Aunque alguno de los p-valores originales pudieran (marginalmente) considerarse como significativos cuando corregimos por comparaciones múltiples por el método de Benjamini-Hochberg no lo son salvo tres de ellos si trabajamos con una FDR de 0.05.

Con una predictora categórica

Analizamos la posible dependencia con el alcoholismo recogida en la covariable `case`.

```
design = model.matrix(~ 0 + pData(gse44456)[,"case"])
colnames(design) = c("control","alcoholic")
head(design)
```

```
  control alcoholic
1 1 0
2 0 1
3 1 0
4 0 1
5 1 0
6 0 1
```

Ajustamos el modelo.

```
fit = limma::lmFit(gse44456,design)
```

Podemos definir un contraste como la diferencia de las medias entre los dos grupos.

```
(contrast.matrix = limma::makeContrasts(dif = control − alcoholic,
                                        levels = design))
```

```
         Contrasts
Levels dif
  control 1
  alcoholic -1
```

Estimamos.

```
fit2 = limma::contrasts.fit(fit,contrast.matrix)
fit2 = limma::eBayes(fit2)
```

Veamos cuáles son significativos.

```
topTable(fit2,coef=1,adjust="BH",number=3)
```

```
           logFC AveExpr t
7927186 -0.5424921 7.826424 -5.914411
8125919 -1.1358866 8.313619 -5.628792
8021081 -1.2885800 8.593822 -5.481851
           P.Value adj.P.Val B
7927186 5.718213e-07 0.01903993 5.029195
8125919 1.455678e-06 0.02236997 4.307423
8021081 2.351231e-06 0.02236997 3.935029
```

Con dos predictoras categóricas

En el segundo análisis nos fijamos en dos covariables, el alcoholismo y el sexo de la persona. Es un diseño factorial 2×2. Veamos cuántos datos tenemos en cada celda.

```
table(pData(gse44456)[,"case"],pData(gse44456)[,"gender"])
```

```
         male female
  control 13 6
  alcoholic 14 6
```

En un diseño como este las preguntas habituales son las siguientes:

1. ¿Qué genes muestran un comportamiento diferenciado o relacionado con el alcoholismo?

2. ¿Qué genes se comportan de un modo distinto para hombres y mujeres?

3. ¿Para qué genes los cambios en su expresión según el alcoholismo son distintos en cada uno de los sexos?

Una aproximación simple y efectiva es construir un solo factor con todas las combinaciones de los factores.

```
casegender = vector("list",ncol(gse44456))
for(i in seq_along(casegender))
  casegender[[i]] = paste0(pData(gse44456)[,"case"][i],
                           pData(gse44456)[,"gender"][i])
casegender = factor(unlist(casegender))
```

Consideremos la siguiente matriz de diseño.

```
design = model.matrix(~ 0 + casegender)
colnames(design) = levels(casegender)
head(design)
```

```
  alcoholicfemale alcoholicmale controlfemale
1 0 0 0
2 0 1 0
3 0 0 0
4 0 1 0
5 0 0 0
6 0 1 0
  controlmale
1 1
2 0
3 1
4 0
5 1
6 0
```

Podemos ver que cada coeficiente corresponde con la expresión media para la correspondiente combinación de factores.

```
fit = limma::lmFit(gse44456,design)
```

Construimos los contrastes en que estamos interesados.

```
cont.matrix = makeContrasts(
    dif1 = controlmale - alcoholicmale,
    dif2 = controlfemale - alcoholicfemale,
    dif12 = (controlmale - alcoholicmale)-
        (controlfemale - alcoholicfemale),
    levels = design)
fit2 = contrasts.fit(fit,cont.matrix)
fit2 = limma::eBayes(fit2)
```

Podemos ejecutar el siguiente código y podemos comprobar que ningún contraste es significativo.

```
topTable(fit2,coef=1,adjust="BH")
topTable(fit2,coef=2,adjust="BH")
topTable(fit2,coef=3,adjust="BH")
```

Otra opción con dos predictoras categóricas

Vamos a utilizar las dos variables fenotípicas sin combinarlas en un solo factor. Es una opción más natural posiblemente. Vamos a considerar el modelo que tiene a las predictoras `case` y lstinline|gender| y su posible interacción (indicado con `*` en la `formula` que sigue.

```
design = model.matrix(~ pData(gse44456)[,"case"] * pData(gse44456)[,"gender"])
colnames(design) = c("intercept","case","gender","case:gender")
head(design)
```

```
  intercept case gender case:gender
1 1 0 0 0
2 1 1 0 0
3 1 0 0 0
4 1 1 0 0
5 1 0 0 0
6 1 1 0 0
```

Ajustamos los modelos

```
fit = limma::lmFit(gse44456,design)
fit1 = limma::eBayes(fit)
```

¿Qué sondas muestran una interacción significativa.

```
topTable(fit1,coef=4)
```

```
           logFC AveExpr t
7895726 0.8494104 7.018058 4.773443
8027746 -1.3351585 4.740675 -4.486664
7893842 -0.9733824 5.149378 -4.418380
7895503 1.1804148 4.922862 4.338423
8162729 -0.5078698 6.114405 -4.312096
7893210 -1.3570688 6.241701 -4.306972
8050352 -1.0632392 7.273278 -4.258376
8117458 -0.7605210 5.932542 -4.204022
8009685 -0.6736454 8.596666 -4.169532
8002403 -0.8195714 9.355693 -4.147799
           P.Value adj.P.Val B
7895726 2.534820e-05 0.5735803 -2.519938
8027746 6.192538e-05 0.5735803 -2.701908
7893842 7.645580e-05 0.5735803 -2.745647
7895503 9.775600e-05 0.5735803 -2.797025
8162729 1.059678e-04 0.5735803 -2.813976
7893210 1.076427e-04 0.5735803 -2.817277
8050352 1.248731e-04 0.5735803 -2.848612
8117458 1.473500e-04 0.5735803 -2.883714
8009685 1.636148e-04 0.5735803 -2.906014
8002403 1.747505e-04 0.5735803 -2.920075
```

No hay ninguno. Parece lógico que ajustaramos el modelo sin interacciones.

```
design = model.matrix(~ pData(gse44456)[,"case"]+ pData(gse44456)[,"gender"])
colnames(design) = c("intercept","case","gender")
```

9.3 Ejercicios

* **Ex. 17 —** Utilizando los datos `tamidata::gse20986` se pide:

1. Seleccionar las muestras correspondientes a iris y huvec.
2. Aplicar el método SAM y obtener el grupo de genes significativos.
3. Comparar con el grupo obtenido en el apartado 4 del ejercicio 15.

* **Ex. 18 —** Utilizando los datos `tamidata::gse20986` se pide determinar los genes significativos cuando comparamos los cuatro grupos considerados en el estudio.

Capítulo 10

Expresión diferencial con datos de conteo

10.1 Introducción

Tenemos datos de expresión de gen obtenidos mediante la técnica conocida como RNASeq o cualquier procedimiento que nos proporcione como respuesta un número entero no negativo, un conteo en lo que sigue.

Ya hemos realizado todo el preprocesado. Ya hemos alineado y contado lecturas asociadas **a gen o a transcritos o exones o bacterias o, genéricamente, al objeto biológico de interés**. Nuestra información muestral es una matriz de expresión **observada** $\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$ donde hemos considerado N genes y tenemos n muestras. Además tendremos covariables, variables fenotípicas o metadatos. [93] Estas covariables son otra matriz $n \times p$ donde n es el número de muestras y p es el número de covariables de las que disponibles sobre las muestras. Tendremos $\boldsymbol{x} = [x_{jk}]_{j=1,\ldots,n;k=1,\ldots,p}$. Habitualmente $p = 1$ y $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ es simplemente un vector.

[93] Un estadístico dice covariable, un bioinformático dice variable fenotípica y un moderno de los de los Big Data diría cualquier cosa.

Como indicamos el estudio se puede realizar a nivel de transcrito o a nivel de gen. Si utilizamos sus abreviaturas en inglés podemos hablar de DTE (differential transcript expression) y DGE (differential gene expression). En lo que sigue abreviamos de este modo.

10.2 Una muestra por condición

¿Y si no tenemos réplicas en cada condición que queremos comparar? Lo primero sería pensar en tomar alguna muestra más por condición que los milagros solamente son los jueves.http://es.wikipedia.org/wiki/Los_jueves,_milagro y https://www.youtube.com/watch?v=-yDNya7XP-4. Lo mejor, esperar a tener más. ¿Y si no voy a tener más? Sería de agradecer que las guardaras y no publicaras nada. Lo que sigue en esta sección intenta explicar este comentario.

Queremos comparar dos condiciones y disponemos de una muestra en cada una de las condiciones. ¿Qué podemos hacer para comparar los conteos para cada gen? En esta (mala) situación la información que tenemos para cada gen la podemos recoger en una tabla de contingencia 2×2 (10.1). En esta tabla y_j recoge el número de lecturas alineadas sobre el gen que estamos estudiando en la muestra j (con $j = 1, 2$). Los tamaños de las librerías en cada muestra son m_j (con $j = 1, 2$).

Tabla 10.1: Una muestra por condición en donde la primera sería caso y la segunda control.

	Muestra 1 (Caso)	Muestra 2 (Control)	Total
Gen i	y_{i1}	y_{i2}	$y_{i\cdot} = y_{i1} + y_{i2}$
Resto	$m_1 - y_{i1}$	$m_2 - y_{i2}$	$m_1 + m_2 - y_{i\cdot}$
Total	m_1	m_2	$m_1 + m_2$

Vamos a considerar dado el total de conteos asociados al gen i-ésimo, esto es, consideramos constante el conteo $y_{i1} + y_{i2}$.[94] El gen que consideramos tiene un conteo aleatorio Y_{i1} en la primera muestra (por ejemplo, correspondiente a caso). Su distribución de probabilidad será binomial con $y_{i1}+y_{i2}$ pruebas y una probabilidad de éxito (lectura procedente de la primera muestra) desconocida p. Si no hay diferencias entre las muestras el valor de p debiera de coincidir con $m_1/(m_1 + m_2)$, es decir, debe coincidir la proporción con la proporción que la primera muestra representa en el total de las dos muestras. Por tanto, evaluar la expresión diferencial del gen se traduce en un contraste sobre p la probabilidad de que la lectura corresponda al caso. Una primera referencia en este sentido es [75].

[94] Que obviamente es aleatorio. Trabajamos condicionados a ese valor.

En `edgeR::binomTest()` utilizan como p-valor la suma de las probabilidades de los conteos que tienen una probabilidad menor o igual al observado bajo la hipótesis nula.

Consideremos los datos `tamidata::PRJNA297664`. Nos quedaremos con una muestra "wild" y otra "SEC66 deletion". Compararemos entre sí estas dos muestras.

```
pacman::p_load("SummarizedExperiment")
data(PRJNA297664,package="tamidata")
```

La variable que indica el tipo es `treatment`.

```
colData(PRJNA297664)[,"treatment"]
```

```
[1] Wild Wild SEC66 deletion
[4] Wild SEC66 deletion SEC66 deletion
Levels: Wild SEC66 deletion
```

Vamos a comparar las muestras 1 y 3 con la función `edgeR::binomTest` ↪ `()`.

```
pacman::p_load(edgeR)
pvalor13 = edgeR::binomTest(assay(PRJNA297664)[,1],
                            assay(PRJNA297664)[,3])
```

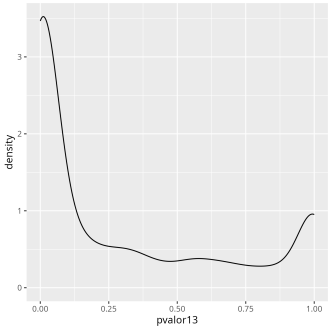


Figura 10.1: Densidad de p-valores.

Podemos hacernos una idea de cómo son estos p-valores con una estimación de su función de densidad (figura 10.1).

También podemos aplicar para el mismo problema el test de Fisher. En cualquier caso, **solamente (y no mucho por no decir nada) tiene sentido hacer este análisis cuando tenemos una muestra en cada condición**.

¿Qué hacemos con más de una muestra? ¿Realizar todas las comparaciones dos a dos? No. Obviamente hay que buscar procedimientos más adecuados.

¿Se puede realizar algo similar cuando tenemos varias muestras por cada una de las dos condiciones? Es tentador simplemente sumar los conteos de cada gen en cada condición. Tendríamos el total de lecturas

por gen y condición. Sin embargo, si consideramos la tabla 10.1 correspondiente estaríamos ignorando la variabilidad intra condición de los conteos correspondientes. En este caso hemos de acudir a alguno de los procedimientos que vemos en la siguientes secciones.

Vamos a evaluar de un modo empírico porqué necesitamos más muestras por condición. Porqué esa práctica (más frecuente de lo que se debiera) de tener una muestra por condición no se debe de utilizar. Siguiendo con `PRJNA297664` tenemos 9 posibles pares formados por una muestra por condición. Vamos a calcular los p-valores para cada una de estas nueve posibles comparaciones y ver cómo varían estos p-valores.

```
pares = rbind(c(1,3),c(1,5),c(1,6),c(2,3),c(2,5),c(2,6),
              c(4,3),c(4,5),c(4,6))
aux = function(rr)
    binomTest(assay(se)[,rr[1]],assay(se)[,rr[2]])
pvalores = apply(pares,1,aux)
```

Por cada gen tenemos el p-valor resultado de comparar cada uno de los pares. Tenemos pues 9 p-valores por gen. Calculamos el p-valor medio, el mínimo y el máximo y los ordenamos de menor a mayor. Finalmente representamos en abscisas el p-valor medio y en ordenadas dos líneas correspondientes al mínimo y al máximo. Dada la gran cantidad de puntos hemos de representar una versión suavizada del mínimo y máximo.

```
pvalores.mean = apply(pvalores,1,mean) ##Media de p-valores
pvalores.min = apply(pvalores,1,min) ##Mínimo de p-valores
pvalores.max = apply(pvalores,1,max) ##Máximo de p-valores
df = data.frame(pvalores.mean,pvalores.min,pvalores.max)
df=df[sort(pvalores.mean,index.return=TRUE)$ix,] ## Ordenamos
df1 = reshape2::melt(df,id="pvalores.mean")
pp = ggplot(df1,aes(x=pvalores.mean,y=value,
                    color=variable))
pp = pp + geom_smooth() # Suavizamos mínimo y máximo
ggsave(paste0(dirTamiFigures,"RNASeqDE10.png"))
```

En la figura 10.2 tenemos el resultado. Podemos ver la tremenda variabilidad que tenemos.

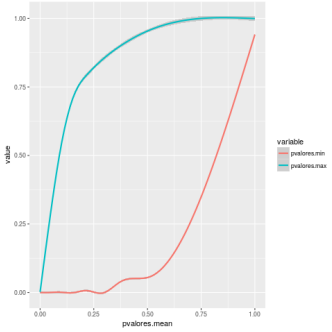


Figura 10.2: p-valores.

10.3 edgeR

El título de la sección hace referencia al paquete (de largo desarrollo) [38, edgeR]. Con frecuencia se habla de método `edgeR` sin más indicación. Este paquete lleva un cuerpo metodológico importante y lo correcto es referenciar las funciones que se están utilizando y las referencias bibliográficas en donde se explican. Hay aproximaciones muy distintas contenidas en este paquete.[95] Sin duda, la mejor referencia a consultar son las viñetas del paquete y [37].

10.3.1 edgeR clásico

En esta sección tratamos un procedimiento propuesto en [125, 124][96] y que está implementado en [38].

Estimación de una dispersión común

Consideramos una característica en n muestras (o librerías) de tamaños distintos.[97] Sea m_j el tamaño de la j-ésima librería (total de lec-

[95] Un comentario similar se puede aplicar a muchos otros paquetes de R. Una costumbre incorrecta es denotar el método de análisis por el paquete que lo incluye y no por la referencia bibliográfica que lo propone. Muchos paquetes incluyen métodos muy diversos.

[96] Hay que leer primero [125] y luego [124] pues es la secuencia temporal de los trabajos aunque se publicaron en orden invertido.

[97] Son réplicas de una misma experimentación con tamaños distintos.

turas). Sea λ la proporción que hay en una librería cualquiera de la característica en que estamos interesados. *Vamos a asumir* que Y_j tiene una distribución binomial negativa con media $\mu_j = m_j\lambda$ y con dispersión ϕ. Y tiene una distribución binomial negativa con media μ y dispersión ϕ, $Y \sim NB(\mu, \phi)$, si su función de probabilidad es $P(Y = y|\mu, \phi) = \frac{\Gamma(y+\phi^{-1})}{\Gamma(\phi^{-1})\Gamma(y+1)}\left(\frac{1}{1+\mu\phi}\right)^{\phi^{-1}}\left(\frac{\mu}{\phi^{-1}+\mu}\right)^{y}$ para $y = 0, 1, \ldots$. Su media es $E(Y) = \mu$ y su varianza $var(Y) = \mu + \phi\mu^2$.

Supongamos el caso ideal (e irreal) en que todas las librerías tienen el mismo tamaño: $m_j = m$ para $j = 1, \ldots, n$. Bajo esta condición tenemos que $Z = \sum_{i=j}^{n} Y_j \sim NB(nm\lambda, \phi/n)$. Además la logverosimilitud condicionada al valor de $Z = z$ no depende de λ y podemos estimar el valor de ϕ. Se estimaría ϕ maximizando la logverosimilitud condicionada dada por

$$\ell_{\mathbf{y}|z} = \left[\sum_{j=1}^{n} \log\Gamma(y_j + 1/\phi)\right] + \log\Gamma(n/\phi) - \log\Gamma(z + n/\phi) - n\log\Gamma(1/\phi), \quad (10.1)$$

donde $\mathbf{y} = (y_1, \ldots, y_n)$ son los conteos observados. Si los tamaños de las librerías son distintos la verosimilitud no tiene una expresión simple. La idea en [125] es modificar los valores observados y_i y generar unos nuevos datos en que los tamaños de las librerías coincidan, los *pseudodatos*. El tamaño común será la media geométrica de los tamaños observados, $m^* = \sqrt[n]{\prod_{j=1}^{n} m_j}$. ¿Cómo transformamos los conteos y_j? El procedimiento propuesto es:

1. Inicializamos ϕ (por ejemplo, con el estimador maxímo verosímil condicionado sin realizar ningún ajuste).

2. Dado el valor estimado de ϕ, estimamos λ maximizando la verosimilitud para el valor estimado de la dispersión.

3. Suponemos que cada conteo y_j es un valor observado de una distribución binomial negativa con media $m_j\lambda$ y parámetro de dispersión ϕ. Calculamos los percentiles

$$p_j = P(Y < y_j|m_j\lambda, \phi) + \frac{1}{2}P(Y = y_j|m_j\lambda, \phi), \quad (10.2)$$

   para $j = 1, \ldots, n$.

4. Suponemos ahora una distribución binomial negativa con media $m^*\lambda$ y dispersion ϕ. Determinamos qué valor sería el percentil de orden p_j en la nueva distribución. Ya no tiene porqué ser un dato entero e incluso puede ser negativo. Los pseudodatos para un mismo gen tienen aproximadamente la misma distribución.

5. Estimamos la dispersión ϕ con los pseudodatos utilizando la veromilitud condicionada a la (pseudo) suma total de un gen.

6. Repetimos desde 2 hasta 5 hasta que converja ϕ.

A este procedimiento de estimación de ϕ y λ los autores lo denominan *máxima verosimilitud condicionada ajustada por cuantiles*.[98] Lo fundamental es entender que estamos modificando los datos para conseguir un tamaño común de las librerías.

[98] Quantile-adjusted condicional maximum likelihood (qCML).

Un test exacto

Tenemos dos condiciones a comparar o las comparamos de dos en dos.

Estimación de la dispersión común Vamos a asumir en un primer momento que el parámetro de dispersión es común. Tenemos y_{ijk} el conteo para la muestra k en la condición j del gen i donde $j = 1, 2$ y $k = 1, \ldots, n_j$. El tamaño de la librería de la condición (j, k) en el gen i es m_{jk}. Supongamos que aplicamos el método qCML dentro de cada condición j de modo que por condición tenemos un tamaño de librería común. Con los pseudodatos que denotamos, por simplicidad notacional, del mismo modo como y_{ijk} podemos condicionar a la suma total por gen dentro de cada clase. **Como el parámetro de dispersión se asume el mismo para todos los genes y condiciones** podemos considerar la logverosimilitud condicionada a $z_{ij} = y_{ij\cdot} = \sum_{k=1}^{n_j} y_{ijk}$ para la dispersión ϕ dada por

$$l_i(\phi) = \sum_{j=1}^{2} \Bigg(\sum_{k=1}^{n_j} \log\Gamma(y_{ijk} + \phi^{-1}) + \log\Gamma(n_j\phi^{-1}) - \\ \log\Gamma(z_{ij} + n_j\phi^{-1}) - n_j \log\Gamma(\phi^{-1}) \Bigg). \quad (10.3)$$

En consecuencia la logverosimilitud común es[99]

$$l(\phi) = \sum_{i=1}^{N} l_i(\phi). \quad (10.4)$$

[99] Sería más correcto hablar de pseudoverosimilitud ya que los distintos genes son dependientes.

La estimación de ϕ en 97 se obtiene maximizando la función dada en la ecuación 10.4. Denotamos a este estimador común por $\hat{\phi}_C$ y es el usado en lo que sigue.

Contraste de hipótesis Tenemos dos condiciones a comparar y estamos asumiendo

$$EY_{ijk} = m_{jk}\lambda_{ij},$$

para cualquier i, j, k siendo Y_{ijk} el conteo aleatorio del gen i en la condición j y en la muestra k. Si consideramos la hipótesis nula de que no hay diferencia entre los valores de λ,

$$H_i: \quad \lambda_{i1} = \lambda_{i2},$$
$$K_i: \quad \lambda_{i1} \neq \lambda_{i2}.$$

Bajo la hipótesis nula el valor λ no depende de la condición. Aplicamos el método qCML a **todas** las muestras (tenemos $n = n_1 + n_2$). Como es habitual denotamos los nuevos pseudodatos como los originales. El tamaño de las librerías es la media geométrica de los tamaños originales y la denotamos por m^*.[100] Estos nuevos psedodatos tienen una distribución común. En concreto para el gen i denotamos por y_{ijk} el conteo para la muestra k en la condición j donde $j = 1, 2$ y $k = 1, \ldots, n_j$. El tamaño común de todas las librerías es m^*. Si consideramos la variable aleatoria Y_{ijk} (de la cual y_{ijk} sería el valor observado) entonces

[100] Importante, los pseudodatos no son los mismos utilizados para estimar la dispersión común.

$$EY_{ijk} = m^*\lambda_{ij},$$

para cualquier i, j, k. Bajo la hipótesis nula H_0 no tendríamos diferencia en el valor de λ entre las condiciones y sería un valor común $\lambda_i = \lambda_{i1} = \lambda_{i2}$ para el gen i. Utilizando el estimador previamente calculado $\hat{\phi}_C$ y los pseudatos (que obtienen un tamaño de librería común) y_{ijk} con $k = 1, \ldots, n_j$ podemos estimar λ_i. Simplemente el estimador máximo verosímil no es más que el cociente de la suma de los conteos dividido por la suma de los tamaños de las librerías. Tenemos las estimaciones $\hat{\lambda}_i$ y $\hat{\phi}_C$.

Bajo la hipótesis nula de no diferencia entre grupos tendríamos que $Y_{ij\cdot} = \sum_{k=1}^{n_j} Y_{ijk} \sim NB(n_j m^* \hat{\lambda}_i, \hat{\phi}_C/n_j)$ para $j = 1, 2$. Además $Y_{i1\cdot}$ e $Y_{i2\cdot}$ son independientes. La suma $Y_{i1\cdot} + Y_{i2\cdot}$ también tiene una distribución binomial negativa: $Y_{i1\cdot} + Y_{i2\cdot} = \sum_{i=1}^{2} \sum_{k=1}^{n_j} Y_{ijk} \sim NB((n_1 + n_2)m^* \hat{\lambda}_i, \hat{\phi}_C/(n_1 + n_2))$. Podemos considerar la distribución condicionada del vector aleatorio $(Y_{i1\cdot}, Y_{i2\cdot})$ a la suma $Y_{i1\cdot} + Y_{i2\cdot}$ y considerar las probabilidades de los conteos conjuntos que *son menos probables que el observado*. La suma de estas probabilidades nos daría el p-valor del test.[101]

[101] Es un test similar al test de Fisher bilateral en donde las probabilidades hipergeométricas son sustituidas por probabilidades por la distribución binomial negativa. Ver [3, Página 93].

Dispersiones posiblemente distintas

En el test exacto que hemos visto en 99 hemos asumido una dispersión común ϕ y se ha estimado su valor utilizando los conteos asociados a todos los genes y condiciones bajo las cuales se han observado a estos genes. Bajo la hipótesis nula de un misma media, y asumiendo la dispersión común esto tenía sentido. Es, sin duda, una hipótesis bastante exigente. Una misma dispersión sobre una cantidad grande de genes no es muy razonable.[102] En [124] proponen no asumir un mismo valor para cada gen y considerar que las dispersiones dependen del gen. Tenemos ahora, en lugar del valor común ϕ, un valor (posiblemente) distinto para el gen i, ϕ_i. ¿Y cómo lo estimamos? Proponen un compromiso entre la contribución que hacen los conteos del i-ésimo gen, l_i (ecuación 10.3), a la logverosimilitud global, l (ecuación 10.4), y la propia logverosimilitud global. En concreto hablan de la logverosimilitud condicionanda ponderada definida como

[102] Buena desde el punto de vista del modelo y su estimación pero nos aleja de los datos.

$$WL(\phi_i) = l_i(\phi_i) + \alpha l(\phi_i), \tag{10.5}$$

siendo α el peso que se da a la verosimilitud global. Es una función que propone un compromiso entre considerar l como función a maximizar considerando la misma contribución a todos los genes y l_i en donde solamente consideramos los conteos del propio gen. El problema fundamental a considerar ahora es la elección del valor de α. Un mayor α nos aproxima a un valor común para la dispersión y un menor valor nos dará estimaciones distintas de la dispersión para cada gen. ¿En qué punto nos quedamos? En [124, Sección, 3.2, pág. 2883] dan una motivación que no es un criterio de optimalidad. Simplemente argumentan lo razonable de su elección. Para explicar cómo lo hacen hemos de considerar la **función score** definida como

$$S_i(\phi) = \frac{\partial l_i(\phi)}{\partial \phi} \tag{10.6}$$

y la información esperada definida como

$$I_i(\phi) = E(J_i), \text{ siendo } J_i = -\frac{\partial^2 l_i(\phi)}{\partial \phi^2}. \tag{10.7}$$

El procedimiento de estimación de α y la posterior estimación de ϕ_i es el siguiente:

1. Estimamos la dispersión común maximizando la verosimilitud global l: $\hat{\phi}_0$.
2. Evaluamos para cada gen i: $S_i(\hat{\phi}_0)$ y $I_i(\hat{\phi}_0)$.
3. Estimamos τ_0 resolviendo la ecuación

$$\sum_{i=1}^{N}\left[\frac{S_i(\hat{\phi}_0)}{I_i(\hat{\phi}_0)(1+I_i(\hat{\phi}_0)\tau_0^2)}-1\right]=0.$$

   Si $\sum_{i=1}^{N} S_i^2(\hat{\phi}_0)/I_i(\hat{\phi}_0) < N$ entonces $\tau_0 = 0$.

4. Fijamos

$$1/\alpha = \tau_0^2 \sum_{i=1}^{N} I_i(\hat{\phi}_0).$$

5. Una vez estimado α en el paso anterior, maximizamos la función $WL(\phi_i)$ definida en la ecuación 10.5.

Finalmente los autores utilizan en lugar de la información esperada $I_i(\hat{\phi}_0)$ una aproximación que consiste en utilizar la observada (entendiendo por observada la correspondiente a los conteos dados). Es decir, sustituyen $I_i(\hat{\phi}_0)$ por $J_i(\hat{\phi}_0)$. Además comentan que trabajan con $\delta = \phi/(\phi+1)$.[103]

[103] Lo cual es irrelevante. Estimamos δ estimamos ϕ pues hay una correspondencia 1-1.

PRJNA297664

En esta sección analizamos los datos `tamidata::PRJNA297664`. Utilizamos parcialmente código de [36].

```
data(PRJNA297664,package="tamidata")
```

Construimos el objeto de clase `edgeR::DGEList`.

```
pacman::p_load(edgeR,SummarizedExperiment)
dge = edgeR::DGEList(counts=assay(PRJNA297664),
                     group=colData(PRJNA297664)[,"treatment"])
```

Podemos ver la variable de agrupación `group` y los tamaños de las librerías (número total de lecturas por muestra).

```
head(dge$samples,n=2)
```

```
       group lib.size norm.factors
Sample1 Wild 4788536 1
Sample2 Wild 9387986 1
```

Añadimos al objeto `dge` la anotación que ya tenemos definida en el `slot` `SummarizedExperiment::rowData`.

```
dge$genes = rowData(PRJNA297664)
```

Podemos ver que no de todos los genes tenemos su anotación.

```
sapply(rowData(PRJNA297664),function(x) sum(is.na(x)))
```

```
    ORF SGD ENTREZID ENSEMBL
      0 10 794 1250
```

Vamos a eliminar genes con conteos bajos. Esto podría deberse a la no expresión del gen o bien indica una actividad baja y estadísticamente no sería muy fiable su análisis. En ambos casos parece razonable eliminarlo del estudio. La selección la haremos básandonos en los conteos por millón. Si y_{ij} es el conteo de la j-ésima muestra correspondiente al i-ésimo y $y_{\cdot j} = \sum_{i=1}^{N} y_{ij}$, el tamaño de la j-ésima librería, entonces el conteo por millón para la muestra j-ésima del gen i-ésimo se define como

$$y_{ij}^{*} = cpm(y_{ij}) = 10^{6} \frac{y_{ij}}{y_{\cdot j}}. \tag{10.8}$$

Los conteos por millón se pueden calcular con `edgeR::cpm()`.

```
head(edgeR::cpm(dge),n=2)
```

```
        Sample1 Sample2 Sample3 Sample4 Sample5
15S_rRNA 0 0 0 0 0
21S_rRNA 0 0 0 0 0
        Sample6
15S_rRNA 0
21S_rRNA 0
```

¿Qué criterio utilizar? Siguiendo indicaciones de [36] podríamos considerar que los conteos por millón superen al menos 0.5 en un número de muestras (que obviamente han de depender de nuestro tamaño muestral). Tenemos 6, podemos pedir que en al menos un par de muestras se verifique la condición (en la referencia indicada son menos exigentes y piden 2 de 12).

```
to_keep1 = rowSums(cpm(dge) > 0.5) > 2
table(to_keep1)
```

```
to_keep1
FALSE TRUE
  948 6178
```

No perdemos más que 948. También podemos basar la selección de genes en valores absolutos. Por ejemplo, podemos ver los conteos por fila. Buscamos el percentil de orden 0.1.

```
quantile(rowSums(dge$counts),probs=.1)
```

```
10%
  6
```

Y podemos quedarnos con genes con una expresión absoluta superior al valor anterior redondeando por exceso que sería 0.

```
to_keep2 = rowSums(dge$counts) > 6
table(to_keep2)
```

```
to_keep2
FALSE TRUE
  714 6412
```

No hay un criterio claro. Elegimos la primera opción en nuestro caso.

```
dge = dge[to_keep1,keep.lib.sizes=FALSE]
dge$samples$lib.size
```

```
[1] 4788094 9387257 9599214 8895385 9003074
[6] 9001313
```

La opción `keep.lib.sizes=FALSE` obliga a que se recalculen los tamaños de las librerías después de la selección que acabamos de hacer.

Se estima el parámetro de dispersión que asumimos un valor común para todos los genes con el siguiente código.

```
dge.c = estimateCommonDisp(dge)
dge.c$common.dispersion
```

```
[1] 0.01170845
```

Podemos estimar parámetros de dispersión por gen. Previamente necesitamos haber estimado el parámetro de dispersión común.

```
dge.t = estimateTagwiseDisp(dge.c)
head(dge.t$tagwise.dispersion)
```

```
[1] 0.01875437 0.01033898 0.05391892 0.02503803
[5] 0.02012587 0.01766447
```

Se realizan los test exactos asumiendo un parámetro de dispersión común.

```
et.c = exactTest(dge.c)
```

Y ver los genes significativos (mostramos los primeros).

```
topTags(et.c,n=2)
```

```
Comparison of groups: SEC66 deletion-Wild
DataFrame with 2 rows and 8 columns
              ORF SGD ENTREZID
       <character> <character> <character>
YBR171W YBR171W S000000375 852469
YCR021C YCR021C S000000615 850385
          ENSEMBL logFC logCPM
       <character> <numeric> <numeric>
YBR171W YBR171W -10.15022 6.11504
YCR021C YCR021C -1.92879 8.46385
           PValue FDR
         <numeric> <numeric>
YBR171W 2.0977e-258 1.29596e-254
YCR021C 4.8495e-47 1.49801e-43
```

Calculamos los tests exactos y mostramos los significativos asumiendo un parámetro de dispersión distinto por gen.

```
et.t = exactTest(dge.t)
topTags(et.t,n=2)
```

```
Comparison of groups: SEC66 deletion-Wild
DataFrame with 2 rows and 8 columns
              ORF SGD ENTREZID
       <character> <character> <character>
YBR171W YBR171W S000000375 852469
YGL255W YGL255W S000003224 852637
          ENSEMBL logFC logCPM
       <character> <numeric> <numeric>
YBR171W YBR171W -10.15052 6.11504
YGL255W YGL255W -1.84605 7.51858
           PValue FDR
         <numeric> <numeric>
YBR171W 1.20767e-297 7.46099e-294
YGL255W 1.00554e-29 3.10613e-26
```

Si queremos obtener los resultados para todos los genes manteniendo su orden original lo podemos hacer con

```
tt1 = topTags(et.t,n=nrow(PRJNA297664),sort.by="none")
```

Y si queremos obtener un `DFrame` para seguir trabajando los datos podemos hacerlo con

```
head(tt1$table,n=3)
```

```
DataFrame with 3 rows and 8 columns
           ORF SGD ENTREZID
    <character> <character> <character>
ICR1 ICR1 S000132612 9164906
LSR1 LSR1 S000006478 9164871
NME1 NME1 S000007436 9164967
        ENSEMBL logFC logCPM PValue
    <character> <numeric> <numeric> <numeric>
ICR1 NA 0.448839 3.511993 0.0290942
LSR1 NA -0.254297 3.382420 0.1519951
NME1 NA 0.417201 0.883312 0.2929631
          FDR
    <numeric>
ICR1 0.0915547
LSR1 0.2835223
NME1 0.4482234
```

TCGA-COAD

Utilizamos los datos obtenidos en § 3.8.4.

```
pacman::p_load(edgeR,SummarizedExperiment)
load(paste0(dirTamiData,"tcga_coad.rda"))
```

Nos centramos en la variable fenotípica `tissue_or_organ_of_origin` que tiene 9 niveles con etiquetas y conteos

```
table(colData(tcga_coad)[,"tissue_or_organ_of_origin"])
```

```
        Ascending colon Cecum
                   72 70
            Colon, NOS Descending colon
                   59 15
Hepatic flexure of colon Rectosigmoid junction
                   12 3
          Sigmoid colon Splenic flexure of colon
                   76 6
       Transverse colon
                   13
```

Eliminamos aquellas muestras que tienen algún dato faltante. Las funciones que siguen necesitan que no haya dato faltante.

```
toremove = which(is.na(colData(tcga_coad)[,"tissue_or_organ_of_origin"]))
tcga_coad = tcga_coad[,-toremove]
```

Contruimos el objeto de clase `DGEList`.

```
dge = DGEList(counts=assay(tcga_coad),
              group=colData(tcga_coad)[,"tissue_or_organ_of_origin"])
```

Eliminamos genes con conteos bajos.

```
to_keep = rowSums(cpm(dge) > 0.5) > 2
dge = dge[to_keep,keep.lib.sizes=FALSE]
```

Estimamos las dispersiones tanto en el que asumimos una común para todos los genes como las que consideramos distintas por gen.

```
dge.c = estimateCommonDisp(dge)
dge.t = estimateTagwiseDisp(dge.c)
```

Comparamos dos de las categorías que nos define nuestro factor experimental.

```
et.c = exactTest(dge.c,pair=c("Ascending colon",
                              "Descending colon"))
et.t = exactTest(dge.t,pair=c("Ascending colon",
                              "Descending colon"))
```

Determinamos los genes significativos.

```
topTags(et.c,n=3)
```

```
Comparison of groups: Descending colon-Ascending colon
         logFC logCPM PValue
PRAC1 4.450539 3.7744837 4.666555e-70
IGFN1 4.212251 0.1655621 5.306039e-59
CALCB -9.164919 3.0768624 6.055306e-58
             FDR
PRAC1 8.007809e-66
IGFN1 4.552581e-55
CALCB 3.463635e-54
```

```
topTags(et.t,n=3)
```

```
Comparison of groups: Descending colon-Ascending colon
        logFC logCPM PValue
ANO4 3.068582 -1.5921044 7.418709e-17
IGFN1 4.210518 0.1655621 1.585287e-15
ACTL8 3.639962 1.9367208 6.320150e-11
             FDR
ANO4 1.273050e-12
IGFN1 1.360177e-11
ACTL8 2.885746e-07
```

10.3.2 edgeR utilizando modelo lineal generalizado

Esta sección se basa en [95]. Se asume un cierto conocimiento de modelos lineales generalizados. Para una introducción rápida y simple podemos consultar [7, capítulo 5, sección 7.5]. Una introducción más completa a un nivel básico es [5] (con código en R) y, por último, una presentación amplia la encontramos en [4].

Siguiendo la notación del manual denotaremos por Y_{ij} el conteo aleatorio (número de lecturas alineadas) para el gen i en la muestra j. Denotamos por $m_j = \sum_{i=1}^{N} y_{\cdot j}$ el total de lecturas de la muestra j. Utilizamos como función de enlace el logarítmo natural y consideramos el tamaño de la librería como offset (un modelo de tasas sobre el tamaño de la librería). El modelo para la media es

$$\ln \mu_{ij} = \boldsymbol{x}_j^T \boldsymbol{\beta}_i + \ln m_j. \tag{10.9}$$

En el modelo que se propone en 10.9 hay que tener en cuenta que las variables predictoras (o variables fenotípicas en este contexto) son comunes a todos los genes. Por ello no aparece en el vector de predictoras $\boldsymbol{x}_j$ la dependencia del gen i. Asumiendo que la componente aleatoria

del modelo sigue una distribución binomial negativa (con el parámetro de dispersión conocido porque de lo contrario no estamos en la familia de dispersión exponencial) entonces la varianza de la respuesta es

$$var(Y_{ijk}) = \mu_{ij} + \phi_i \mu_{ij}^2, \tag{10.10}$$

siendo ϕ_i el parámetro de dispersión que hemos de asumir conocido o, de otro modo, tenemos que estimarlo previamente. En [95, pág. 4290] muestran cómo estimar por máxima verosimitud el vector de coeficientes $\boldsymbol{\beta}_i$. Utilizan una modificación de los mínimos cuadrados iterativamente reponderados (IRWLS). El parámetro de dispersión se estima maximizando la logverosimilitud penalizada definida como

$$APL_i(\phi_i) = \ell(\phi_i; \boldsymbol{y}_i, \hat{\boldsymbol{\beta}}_i) - \frac{1}{2} \ln |\mathbb{I}_i| \tag{10.11}$$

siendo $\boldsymbol{y}_i$ los conteos para el gen i, $\hat{\boldsymbol{\beta}}_i$ el vector de coeficientes, $l()$ es la función de logverosimilitud y $|\mathbb{I}_i|$ el determinante de la matriz de información de Fisher para el i-ésimo gen.

TCGA-COAD

Esta sección utiliza material de [87]. Vamos a utilizar como variables predictoras `age_at_diagnosis` y `tissue_or_organ_of_origin`.

```
pacman::p_load(edgeR,SummarizedExperiment)
load(paste0(dirTamiData,"tcga_coad.rda"))
```

Hemos de eliminar aquellas muestras que tienen las variables predictoras con datos faltantes ya que las funciones que siguen no los admiten.

```
torm1 = which(is.na(colData(tcga_coad)$"age_at_diagnosis"))
torm2 = which(is.na(colData(tcga_coad)$ "tissue_or_organ_of_origin"))
toremove = union(torm1,torm2)
tcga_coad = tcga_coad[,-toremove]
```

Construimos el objeto `DGEList` sin indicar ninguna variable `group` ni ninguna matriz de modelo y eliminamos genes con conteos bajos.

```
dge = DGEList(counts=assay(tcga_coad))
to_keep = rowSums(cpm(dge) > 0.5) > 20
dge = dge[to_keep,keep.lib.sizes=FALSE]
```

Construimos la matriz de modelo con las dos variables predictoras, una de caracter categórico y la otra numérica. También cambiamos los nombres de las columnas de la matriz de modelo.

```
design0 = model.matrix(~ 0 + colData(tcga_coad)$"tissue_or_organ_of_origin" +
    ↪ colData(tcga_coad)$"age_at_diagnosis")
y = levels(colData(tcga_coad)$"tissue_or_organ_of_origin")
y = sapply(y,function(x) gsub(" ","_",x)) ## Eliminamos espacios
y = sapply(y,function(x) gsub(",","_",x)) ## Eliminamos las comas
colnames(design0) = c(y,"age_at_diagnosis")
```

Estimamos las dispersiones por tres métodos distintos: asumiendo una dispersión común, una por gen y con una relación media-varianza. Los métodos con los que se estiman en las dos primeras opciones son distintos al caso del método edgeR clásico de 104.

```
dge = estimateDisp(dge,design=design0)
```

Si solo queremos una de las tres opciones podemos usar las funciones `estimateGLMCommonDisp()`, `estimateGLMTagwiseDisp()` y `estimateGl` ↪ . Podemos ver los estimadores de los parámetros de dispersión en la figura 10.3 generada del siguiente modo.

```
png(paste0(dirTamiFigures,
           "tcga_coad_dge_design0_dispersion.png"))
plotBCV(dge)
dev.off()
```

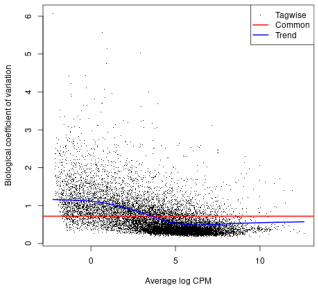


Figura 10.3: Estimadores de los parámetros de dispersión según los tres métodos.

Ajustamos los modelos lineales generalizados.

```
fit = glmFit(dge, design=design0)
```

Veamos si influye en los conteos observados la variable `age_at_diagnosis` ↪ . Si observamos la matriz de modelo `design0` definida previamente corresponde con la columna 10 de la matriz de modelo. Se realiza un test del cociente de verosimilitudes.

```
lrt1 = glmLRT(fit,coef="age_at_diagnosis")
lrt1 = glmLRT(fit,coef=10) ## Equivalente a la línea anterior
topTags(lrt1,n=3)
```

Podemos evaluar toda la variable `tissue_or_organ_of_origin`.

```
lrt2 = glmLRT(fit,coef=1:9)
topTags(lrt2,n=3)
```

Y elegir los contraste que queramos. Mostramos una comparación entre dos grupos.

```
AD = makeContrasts(contrast1 = Ascending_colon - Descending_colon,
                   levels=design0)
lrt3 = glmLRT(fit,contrast = AD)
topTags(lrt3,n=3)
```

PRJNA218851

Vamos a analizar los datos `tamidata2::PRJNA218851`.

```
pacman::p_load(SummarizedExperiment,edgeR)
data(PRJNA218851,package="tamidata2")
```

La variable fenotípica no tiene datos faltantes por lo que no necesitamos eliminar muestras del estudio. Definimos un objeto de clase `DGEList`.

```
dge = DGEList(counts=assay(PRJNA218851))
class(dge)
```

```
[1] "DGEList"
attr(,"package")
[1] "edgeR"
```

Hacemos que el grupo `Normal` sea el primer nivel de la variable categórica `Stage`.

```
colData(PRJNA218851)$Stage = relevel(colData(PRJNA218851)$Stage,
                                     ref = "Normal")
```

Construimos la matriz de modelo.

```
design0 = model.matrix(~ colData(PRJNA218851)$Stage)
colnames(design0) = c("Intercept","Cancer","Metastasis")
```

Estimamos el parámetro de dispersión asumiendo un valor común.

```
dge = estimateGLMCommonDisp(dge)
dge$common.dispersion
```

```
[1] 0.5104526
```

Ajustamos los modelos.

```
fit = glmFit(dge, design=design0)
```

Comparamos grupo normal con grupo con cáncer.

```
lrt1 = glmLRT(fit,coef="Cancer")
lrt1 = glmLRT(fit,coef=2) ## Equivalente a la línea anterior
topTags(lrt1,n=5)
```

```
Coefficient: Cancer
                logFC logCPM LR
ENSG00000136352 7.739779 1.4445070 225.5247
ENSG00000131142 7.063949 2.3721524 218.8761
ENSG00000123500 6.576358 3.3593454 212.1062
ENSG00000230316 7.792502 0.5143565 211.7648
ENSG00000172023 6.142726 2.4200271 196.4356
                   PValue FDR
ENSG00000136352 5.641157e-51 3.602217e-46
ENSG00000131142 1.590513e-49 5.078191e-45
ENSG00000123500 4.768185e-48 9.036184e-44
ENSG00000230316 5.660351e-48 9.036184e-44
ENSG00000172023 1.252258e-44 1.599284e-40
```

Comparamos grupo normal con grupo con metástasis.

```
lrt1 = glmLRT(fit,coef="Metastasis")
lrt1 = glmLRT(fit,coef=3) ## Equivalente a la línea anterior
topTags(lrt1,n=3)
```

```
Coefficient: Metastasis
                logFC logCPM LR
ENSG00000171557 9.413049 9.679592 361.9371
ENSG00000055957 9.472287 6.932345 360.7720
ENSG00000021852 9.622634 5.490674 360.4040
                   PValue FDR
ENSG00000171557 1.066075e-80 4.451018e-76
ENSG00000055957 1.911942e-80 4.451018e-76
ENSG00000021852 2.299366e-80 4.451018e-76
```

Queremos comparar los grupos de cáncer y metástasis con el modelo que tenemos. Para ello definimos un contraste.

```
AD = makeContrasts(contrast1 = Cancer - Metastasis,levels=design0)
lrt1 = glmLRT(fit,contrast = AD)
topTags(lrt1,n=3)
```

```
Coefficient: 1*Cancer -1*Metastasis
                 logFC logCPM LR
ENSG00000110245 -9.614463 7.038351 368.3290
ENSG00000171557 -9.524503 9.679592 367.3941
ENSG00000117601 -9.545839 6.802969 364.5240
                   PValue FDR
ENSG00000110245 4.325325e-82 2.206719e-77
ENSG00000171557 6.911548e-82 2.206719e-77
ENSG00000117601 2.914104e-81 4.403676e-77
```

Podemos realizar el análisis utilizando contrastes.

```
design1 = model.matrix(~ 0+ colData(PRJNA218851)$Stage)
colnames(design1) = levels(colData(PRJNA218851)$Stage)
```

Las comparaciones anteriores las podemos repetir con el siguiente código.

```
AD = makeContrasts(contrast1 = Normal − Cancer,
                   contrast2 = Normal − Metastasis,
                   contrast3 = Cancer − Metastasis,
                   levels=design1)
lrt1 = glmLRT(fit,contrast = AD[,"contrast1"])
topTags(lrt1,n=3)
```

```
Coefficient: 1*Intercept -1*Cancer
                 logFC logCPM LR
ENSG00000136352 -33.10235 1.4445070 673.4680
ENSG00000230316 -33.74969 0.5143565 672.3068
ENSG00000183305 -35.12774 0.9963651 668.7823
                    PValue FDR
ENSG00000136352 1.759600e-148 1.004868e-143
ENSG00000230316 3.147293e-148 1.004868e-143
ENSG00000183305 1.838316e-147 3.196933e-143
```

```
lrt1 = glmLRT(fit,contrast = AD[,"contrast2"])
topTags(lrt1,n=3)
```

```
Coefficient: 1*Intercept -1*Metastasis
                 logFC logCPM LR
ENSG00000221867 -37.32444 1.6769108 722.7515
ENSG00000165471 -34.63815 2.8429305 716.9856
ENSG00000261012 -36.82937 0.9138824 710.8163
                    PValue FDR
ENSG00000221867 3.375538e-159 2.155484e-154
ENSG00000165471 6.055447e-158 1.933383e-153
ENSG00000261012 1.329426e-156 2.212137e-152
```

```
lrt1 = glmLRT(fit,contrast = AD[,"contrast3"])
topTags(lrt1,n=3)
```

```
Coefficient: 1*Cancer -1*Metastasis
                 logFC logCPM LR
ENSG00000110245 -9.614463 7.038351 368.3290
ENSG00000171557 -9.524503 9.679592 367.3941
ENSG00000117601 -9.545839 6.802969 364.5240
                   PValue FDR
ENSG00000110245 4.325325e-82 2.206719e-77
ENSG00000171557 6.911548e-82 2.206719e-77
ENSG00000117601 2.914104e-81 4.403676e-77
```

10.3.3 edgeR utilizando cuasiverosimilitud

En esta sección tratamos el método propuesto en [88] e implementado en [38]. Esta sección es más técnica pero de gran interés. Modificamos algo la notación que proponen los autores.

Denotamos por Y_{ijk} el conteo observado para el gen i-ésimo en la condición j-ésima y la réplica k-ésima donde $i = 1, \ldots, I$, $j = 1, \ldots, J$ y $k = 1, \ldots, n_j$. Por lo tanto estamos considerando I posibles genes y J posibles condiciones donde el número de réplicas por condición no tienen porqué ser igual. Consideramos un factor de normalización (offset) para la muestra (j, k) como c_{jk}. Denotamos $E(Y_{ijk}|c_{jk}) = \mu_{ijk}$. Realmente la notación anterior no ha de interpretarse como una esperanza condicionada en la medida en que el factor de normalización no lo estamos asumiendo como aleatorio. Asumimos que $\mu_{ijk} = \lambda_{ij} c_{jk}$. Una vez corregida por el factor de normalización la media del conteo lo que nos

queda es λ_{ij}. Si se considera que el gen i no tiene expresión diferencial entonces debemos de observar que el valor de λ_{ij} no depende del valor j, no depende de la condición, no depende de las posibles variables predictoras. Denotamos el vector de variables predictoras con $\boldsymbol{x}_{jk}$ entonces el modelo que asumimos para la media es

$$\ln \mu_{ijk} = \ln c_{jk} + \ln \lambda_{ij} = \log c_{jk} + \boldsymbol{x}_{jk}^T \boldsymbol{\beta}_{ijk},$$

esto es, un offset (c_{ij}) más la componente sistemática o predictor lineal. No se supone un modelo lineal generalizado por lo que no tenemos que especificar una componente aleatoria, esto es, la distribución de Y_{ijk} no tiene porqué estar completamente especificada (en particular, dentro de la familia de dispersion exponencial). Los autores proponen un modelo de cuasiverosimilitud ([96, páginas 323 y siguientes]). Para este tipo de modelización no necesitamos especificar necesariamente una distribución para la respuesta sino que tenemos que especificar cómo se relaciona la varianza de la variable con la media salvo un factor de proporcionalidad. Estamos asumiendo que

$$var(Y_{ijk}) = \phi_i V_i(\mu_{ijk}),$$

siendo V_i una función que especifica el usuario y ϕ_i un parámetro de dispersión que hemos de estimar a partir de los datos y luego trabajaremos condicionado a este valor. Si asumimos que la variable respuesta Y_{ijk} sigue una distribución binomial negativa entonces $V_i(\mu_{ijk}) = \mu_{ijk} + \omega_i \mu_{ijk}^2$. En [97] tenemos otros ejemplos. Las ecuaciones de cuasiverosimilitud vienen dadas por

$$\frac{\partial l(\mu_{ijk}|y_{ijk})}{\partial \mu_{ijk}} = \frac{y_{ijk} - \mu_{ijk}}{V_i(\mu_{ijk})} = 0,$$

[104] La función de cuasiverosimilitud propuesta por Wedderburn [144] se define como $\int_y^\mu \frac{y-t}{V(t)} d\mu$.

donde l es la función de cuasiverosimilitud.[104] Si asumimos una función varianza de acuerdo con el modelo binomial negativo y utilizamos la función de cuasiverosimilitud entonces nos aparecen dos parámetros de dispersión, ϕ_i y ω_i, el primero de ellos se refiere a la dispersión de la cuasiverosimilud y el segundo a la dispersión de la distribución binomial negativa. Utilizando el método de cuasiverosimilitud asumimos que $var(Y_{ijk} = \phi_i(\mu_{ijk} + \omega_i \mu_{ijk}^2)$ que no es lo mismo que asumir una respuesta binomial negativa en donde asumimos que $\phi_i = 1$. Añade mayor flexibilidad a la modelización de la varianza de la respuesta.

Para cada gen i (con $i = 1, \ldots, I$) tenemos los conteos observados $\boldsymbol{y}_i = (y_{i11}, \ldots, y_{iJn_J})$ y el vector de medias $\boldsymbol{\mu}_i = (\mu_{i11}, \ldots, \mu_{iJn_J})$ y la correspondiente función de cuasiverosimilitud

$$\boldsymbol{l}_i(\boldsymbol{\mu}_i; \boldsymbol{y}_i) = \sum_{j=1}^{J} \sum_{k=1}^{n_j} l_i(\mu_{ijk}; y_{ijk}), \tag{10.12}$$

siendo l_i la función de cuasiverosimilitud correspondiente a la función varianza que se ha elegido para el i-ésimo gen.

La comparación de dos modelos puede realizarse utilizando un test del cociente de cuasiverosimilitudes cuyo estadístico se define como

$$LRT_i = 2(\boldsymbol{l}_i(\hat{\mu}_i; \boldsymbol{y}_i) - \boldsymbol{l}_i(\tilde{\mu}_i; \boldsymbol{y}_i) \tag{10.13}$$

siendo $\hat{\boldsymbol{\mu}}_i$ y $\tilde{\boldsymbol{\mu}}_i$ los estimadores cuasi máximo verosímiles (los que maximizan la función de cuasiverosimilitud) bajo la hipótesis nula y la alternativa respectivamente. Se prueba que si la función varianza ha sido

correctamente especificada entonces

$$LRT_i \sim \phi_i \chi^2_q + O_p(n^{-1/2}), \tag{10.14}$$

siendo q la diferencia de dimensiones del espacio paramétrico completo y el restringido en la hipótesis nula y n el total de muestras. El parámetro de dispersión se puede estimar con

$$\hat{\phi}_i = \frac{2(\boldsymbol{l}_i(\boldsymbol{y}_i;\boldsymbol{y}_i) - \boldsymbol{l}_i(\hat{\boldsymbol{\mu}}_i;\boldsymbol{y}_i)}{n-p} \tag{10.15}$$

donde p es la dimensión del espacio paramétrico (la dimensión del vector de coeficientes $\boldsymbol{\beta}$). Se sugiere utilizar el siguiente cociente

$$F_{QL} = \frac{LRT_i}{\hat{\phi}_i} \tag{10.16}$$

utilizando como distribución aproximada una F de Fisher con q y $n-p$ grados de libertad.

En este momento tenemos un parámetro de dispersión estimado para cada una de los genes $\hat{\phi}_i$. Ahora los autores proponen utilizar una idea de [129] intentando combinar los estimadores de los parámetros de dispersión de distintos genes con la idea de mejorar la estimación de los mismos. Es una aproximación empírica bayesiana por lo que asume una distribución de probabilidad sobre los parámetros de dispersión. En concreto se asume que

$$d_0 \frac{\phi_0}{\phi_i} \sim \chi^2_{d_0} \tag{10.17}$$

y que

$$\frac{(n-p)\hat{\phi}_i}{\phi_i} | \phi_i \chi^2_{n-p} \tag{10.18}$$

donde esto último hay que interpretarlo como que $\frac{(n-p)\hat{\phi}_i}{\phi_i}$ *condicionado* a ϕ_i sigue una distribución χ^2_{n-p}. Bajo las hipótesis expuestas en (10.17) y 10.18 se tiene que la distribución a posteriori sobre $1/\phi_i$ es una distribución gamma, es decir,

$$\frac{1}{\phi_i} | \hat{\phi}_i \sim Gamma(\frac{1}{2}(d_0 + n - p), \frac{1}{2}(d_0\phi_0 + (n-p)\hat{\phi}_i). \tag{10.19}$$

Los hiperparámetros d_0 y ϕ_0 los podemos estimar a partir de los valores estimados $\hat{\phi}_i$ utilizando un método de los momentos propuesto en [129] que el mismo utilizado en el método limma (§ 9.2). Se propone sustituir el estimador $\hat{\phi}_i$ obtenido directamente a partir de la función de cuasiverosimilitud por la media de la distribución a posteriori (10.19). Por tanto el estimador sería

$$\hat{\phi}^s_i = \frac{1}{\widehat{E(\phi_i^{-1}|\hat{\phi}_i)}} = \frac{\hat{d}_0\hat{\phi}_0 + (n-p)\hat{\phi}_i}{\hat{d}_0 + (n-p)}. \tag{10.20}$$

Como vemos el nuevo estimador de la dispersión del i-ésimo gen no es más una media ponderada entre la estimación de cuasiverosimilitud y la obtenida por el método de los momentos a partir de **todos** los estimadores $\hat{\phi}_i$. Podemos considerarlo como una versión suavizada del estimador de cuasiverosimilitud. Ahora la distribución aproximada será

$$\frac{LRT_i}{q\hat{\phi}^s_i} \sim F_{q,\hat{d}_0+(n-p)} \tag{10.21}$$

TCGA-COAD con cuasiverosimilitud

Utilizamos los datos de la sección § 10.3.2 así como la matriz de modelo allí definida en 202 con las variables fenotípicas `tissue_or_organ_of_origin` y `age_at_diagnosis`.

Ajustamos los modelos lineales generalizados por el método de cuasiverosimilitud.

```
qlfit = glmQLFit(dge, design=design0)
```

Veamos si influye en los conteos observados la variable `age_at_diagnosis`. Si observamos la matriz de modelo `design0` definida previamente corresponde con la columna 10 de la matriz de modelo. Se realiza un test del cociente de verosimilitudes.

```
Ftest1 = glmQLFTest(qlfit,coef="age_at_diagnosis")
```

```
Error in glmQLFTest(qlfit, coef = "age_at_diagnosis"): need to run
    ↪ glmQLFit before glmQLFTest
```

```
Ftest1 = glmQLFTest(qlfit,coef=10) ## Equivalente a la línea anterior
```

```
Error in glmQLFTest(qlfit, coef = 10): need to run glmQLFit before
    ↪ glmQLFTest
```

```
topTags(Ftest1)
```

```
Error: objeto 'Ftest1' no encontrado
```

Podemos evaluar toda la variable `tissue_or_organ_of_origin`.

```
Ftest2 = glmQLFTest(qlfit,coef=1:9)
```

```
Error in glmQLFTest(qlfit, coef = 1:9): need to run glmQLFit before
    ↪ glmQLFTest
```

```
topTags(Ftest2)
```

```
Error: objeto 'Ftest2' no encontrado
```

Podemos evaluar los contrastes que queramos. Mostramos una comparación entre dos grupos.

```
AD = makeContrasts(contrast1 = Ascending_colon − Descending_colon,
                   levels=design0)
lrt3 = glmLRT(qlfit,contrast = AD)
topTags(lrt3)
```

10.4 SAMseq

Este método propuesto en [82] está implementado en [**R-samr**]. Supongamos el caso con dos grupos. Denotamos como siempre por y_{ij} el conteo de la i-ésima característica en la j-ésima muestra ($i = 1, \ldots, N$ y $j = 1, \ldots, n$). El tamaño de la librería será $m_j = \sum_{j=1}^{N} y_{ij}$. Como siempre el valor observado es y_{ij} mientras que el valor aleatorio (antes de observar los datos) lo denotamos con Y_{ij}. Los índices correspondientes a los dos grupos de muestras los denotamos por C_1 y C_2. Estos conjuntos

de índices constituyen una partición de $\{1, \ldots, n\}$. Suponemos que las clases C_1 y C_2 la componen n_1 y n_2 elementos (con $n_1 + n_2 = n$).

Supongamos, en un primer momento, que todos los tamaños de las librerías son iguales $m_1 = \ldots = m_n = m$. Fijamos la característica i. Ordenamos (de menor a mayor) los conteos $y_{i1}, \ldots, y_{in}$. La posición de y_{ij} en $\boldsymbol{y}_{i*} = (y_{i1}, \ldots, y_{in})$ la denotamos por $r_{ij}(\boldsymbol{y}_{i*})$. Si consideramos el valor aleatorio Y_{i*} entonces tenemos la variable aleatoria $R_{ij}(\boldsymbol{Y}_{i*})$. El estadístico de Mann-Whitney-Wilcoxon[105] sería

$$T_i = \sum_{j \in C_1} R_{ij}(\boldsymbol{Y}_{i*}) - \frac{n_1(n+1)}{2}. \tag{10.22}$$

[105] A veces se le llama de Mann-Whitney, otras de Wilcoxon y otras con los tres nombres.

Se supone que no tenemos empates en $\boldsymbol{Y}_{i*}$. Si T_i es muy grande significa que en la característica i los valores en la primera clase (C_1) tienden a ser mayores que en la segunda (C_2). Un valor muy pequeño indica lo contrario. Notemos que T_i puede tomar valores negativos. En general, si el valor absoluto $|T_i|$ es muy grande, lo que tenemos es que ambos grupos de valores son claramente distintos, uno mayor que el otro. Valores próximos a cero indican que no hay diferencias entre los grupos. Hasta aquí no hay problema ninguno. Simplemente se propone utilizar un test de Mann-Whitney-Wilcoxon para comparar los órdenes en los dos grupos que comparamos. El problema aparece cuando los tamaños de las librerías son distintas que es lo habitual. De hecho, suelen ser muy distintos. Una primera idea que no da buenos resultados es dividir los conteos originales por los correspondientes tamaños de las librerías.

Suponemos fijos los tamaños de las librerías. Esto es, aunque suponemos conteos aleatorios la suma de las correspondientes columnas de conteos las asumimos como fijas.[106] Denotamos el mínimo tamaño de las librerías con $m_0 = \min_{j=1,\ldots,n} m_j$ y la librería donde se da este mínimo suponemos que es la j_0. Supongamos que, para la librería j, del total de lecturas elegimos al azar m_0 (correspondiente a la librería con menor tamaño). Esto supone que cada lectura original de esta librería es conservada con probabilidad $\frac{m_0}{m_j}$ y eliminada con probabilidad $1 - \frac{m_0}{m_j}$. Si denotamos por X_{ij}^T el número de lecturas que nos quedan después de este proceso de muestreo tendremos que

[106] Una hipótesis necesaria.

$$X_{ij}^T | Y_{ij} \sim Binomial(Y_{ij}, \frac{m_0}{m_j}).$$

Los autores llaman a esto un **muestreo hacia abajo**.[107] Este procedimiento de muestreo nos produce unos nuevos conteos y un nuevo valor del estadístico dado en 10.22 que podemos denotar por T_i^T. Notemos que si el tamaño mínimo es mucho menor que el resto de tamaños estamos descartando muchas lecturas.

[107] Down sampling.

Una segunda opción es igualar los tamaños de las librerías a un valor intermedio como la media geométrica $m^* = \left(\prod_{j=1}^n m_j\right)^{1/n}$. Generamos conteos aleatorios con

$$X'_{ij} | Y_{ij} \sim Poisson\left(\frac{m^*}{m_j} Y_{ij}\right). \tag{10.23}$$

Sería el muestreo Poisson. Para evitar empates entre los valores generados se sustituyen estos valores por

$$X'_{ij} + \epsilon_{ij}$$

siendo ϵ_{ij} variables aleatorias independientes y con distribución común uniforme en el intervalo $[0, 0.1]$.[108] Como vemos estamos generando valores por lo que lo que obtenemos también es aleatorio. Supongamos que en un muestreo obtenemos obtenemos $T_i'(b)$. Repetimos este muestreo B veces. Consideramos el estadístico

[108] Un pequeño truco para evitar empates.

$$T_i^* = \frac{1}{B}\sum_{b=1}^{B} T_i'(b). \tag{10.24}$$

Este es el estadístico con el que finalmente se trabaja para contrastar. No conocemos la distribución de este estadístico. Hemos indicado que la región crítica natural, la zona en donde rechazamos la hipótesis nula de que no hay diferencia significativa entre muestras vendría dada por $T_i^* > c$ siendo c un valor a determinar. Dado un valor positivo c consideraremos como significativa a la característica i si $T_i^* > c$. Lo que vamos a hacer es elegir un conjunto de posibles valores para c y **estimaremos** la tasa de hipótesis nulas falsamente rechazadas si utilizamos ese valor de c. El procedimiento[109] es el siguiente:

[109] Ver [132].

1. Calculamos $T_1^*, \ldots, T_N^*$ a partir de nuestros datos.
2. Permutamos S veces las columnas de X manteniendo la asignación de los grupos (las etiquetas de clase) y calculamos $T_1^*(s), \ldots, T_N^*(s)$ utilizando los datos permutados con $s = 1, \ldots, S$.
3. Tomamos una serie de valores para c y calculamos:

$$\hat{V} = \frac{1}{S}\sum_{i=1}^{N}\sum_{s=1}^{S} 1_{|T_i^*(s)|>c} \tag{10.25}$$

   que estima la cantidad de hipótesis nulas falsamente rechazadas y

$$\hat{R} = \sum_{i=1}^{N} 1_{|T_i^*|>c} \tag{10.26}$$

   que estima la cantidad de hipótesis nulas rechazadas.
4. El valor de FDR asociado a cada valor c es estimado con

$$\widehat{FDR}(c) = \frac{\hat{\pi}_0 \hat{V}}{\hat{R}}. \tag{10.27}$$

   donde π_0 es la proporción de hipótesis nulas ciertas.[110] El estimador de π_0 utilizado es

[110] O proporción de características que no son significativamente distintas entre los dos grupos que comparamos.

$$\hat{\pi}_0 = \frac{2\sum_{i=1}^{N} 1_{|T_i^*|\leq q}}{N},$$

   siendo q la mediana de $\{|T_i^*(s)| : i = 1, \ldots, N; s = 1, \ldots, S\}$.

10.4.1 PRJNA297664

Analizamos los datos `tamidata::PRJNA297664` utilizando el método `SAMseq` visto en § 10.4.

```
pacman::p_load("SummarizedExperiment","samr")
```

```
Warning: dependency 'impute' is not available
```

```
Warning in utils::install.packages(package, ...): installation of
    ↪ package 'samr' had non-zero exit status
```

```
Warning in p_install(package, character.only = TRUE, ...):
```

```
Warning in library(package, lib.loc = lib.loc, character.only = TRUE,
    ↪ logical.return = TRUE, : no hay paquete llamado 'samr'
```

```
Warning in pacman::p_load("SummarizedExperiment", "samr"): Failed to
    ↪ install/load:
samr
```

```
data(PRJNA297664,package="tamidata")
```

```
PRJNA297664samseq = SAMseq(x = assay(PRJNA297664),
    y = colData(PRJNA297664)[,"treatment"],
    resp.type="Two class unpaired",geneid = rownames(PRJNA297664),
    genenames=rownames(PRJNA297664),nperms = 1000,nresamp = 40,
    fdr.output = 0.20)
```

Podemos ver los resultados con[111]

[111] No mostrado.

```
print(PRJNA297664samseq)
```

Un dibujo con los resultados que podemos ver en figura 10.4

```
plot(PRJNA297664samseq)
```

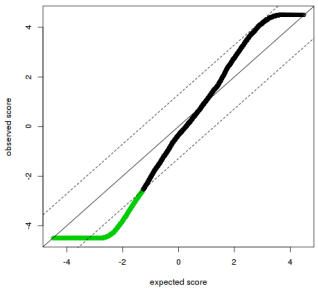


Figura 10.4: Genes significativos con los datos PRJNA297664.

10.5 DESeq2

Este método se propuso en [86].

10.5.1 Método

Denotamos la matriz de conteos aleatoria con $\boldsymbol{Y}$ donde el elemento en posición (i, j) es el conteo aleatorio Y_{ij} correspondiente al i-ésimo gen y a la j-ésima muestra. Se asume que Y_{ij} sigue una distribución binomial negativa con media μ_{ij} y dispersión ϕ_i. Se supone que la media de este conteo aleatorio la podemos expresar como

$$\mu_{ij} = s_{ij} q_{ij}, \tag{10.28}$$

siendo s_{ij} un factor de normalización que tiene en cuenta los distintos tamaños de las librerías así como otros posibles sesgos como contenido GC del gen o su longitud.[112] La dependencia de las covariables (variables fenotípicas) se introduce en q_{ij}. En concreto si asumimos para la muestra j un vector de covariables $\boldsymbol{y}_j = (y_{j1}, \ldots, y_{jp})^T$ y denotamos un vector de coeficientes (dependiente de la fila o gen) con $\boldsymbol{\beta}_i = (\beta_{i1}, \ldots, \beta_{ip})^T$. Se asume el siguiente modelo

$$\log_2 q_{ij} = \boldsymbol{y}_j^T \boldsymbol{\beta}_i = \sum_{k=1}^{p} y_{jk} \beta_{ik}.$$

[112] En la propia modelización incorporar el proceso de normalización que, en principio, puede depender del gen y de la muestra. Por defecto se utilizan s_{ij} que no dependen de i y están definidos en § 3.9.3.

[113]

[113] Un detalle importante sobre la codificación de un factor. No se utilizan las habituales variables dummy que codifican uno para una categoría y cero para las demás dejando la media como el valor en

Los **contrastes** vendrán dados por $\boldsymbol{c}^T\boldsymbol{\beta}_i$. El error estándar (su desviación típica) del contraste $\boldsymbol{c}^T\boldsymbol{\beta}_i$ viene dada por $SE(\boldsymbol{c}^T\boldsymbol{\beta}_i) = \sqrt{\boldsymbol{c}^T\boldsymbol{\Sigma}_i\boldsymbol{c}}$, donde $\boldsymbol{\Sigma}_i$ denota la matriz de covarianzas de $\boldsymbol{\beta}_i$. Cuando se contraste si estos contrastes[114] son nulos se utiliza la matriz de diseño estándar en donde no se introduce el parámetro adicional que hemos comentado anteriormente. Esta sí es una matriz de rango completo.

Estimando las dispersiones

Sobre las distintas dispersiones ϕ_i se asume una distribución lognormal.

$$\log \phi_i \sim N(\log \tilde{\phi}(\bar{\mu}_i), \sigma_d^2) \tag{10.29}$$

donde la media es una función $\tilde{\phi}$ que evaluamos en

$$\bar{\mu}_i = \frac{1}{n}\sum_{j=1}^{n}\frac{y_{ij}}{s_{ij}}. \tag{10.30}$$

¿Qué papel desempeña la función $\tilde{\phi}$? Nos describe cómo depende la media de la distribución a priori (sobre el parámetro de dispersion) de la media muestral de los conteos normalizados observados que denotamos $\bar{\mu}_i$. La varianza σ_d^2 modeliza la variabilidad de las dispersiones alrededor de este comportamiento medio que acabamos de indicar. Se utiliza la siguiente función $\tilde{\phi}$.[115]

$$\tilde{\phi}(\bar{\mu}) = \frac{a_1}{\bar{\mu}} + \phi_0. \tag{10.31}$$

En resumen, tenemos tres parámetros en la distribución a priori. [116] Para tener especificada esta distribución a a priori tenemos que estimar a_1, ϕ_0, σ_d^2. Se utiliza un método empírico bayesiano.[117] ¿Cómo los estimamos?

Estimación de las dispersiones ϕ_i's Se empieza ajustando un modelo lineal generalizado con componente aleatoria binomial negativa. Se realiza una primera estimación en donde aplicando el método de los momentos y los conteos estimamos las medias y los parámetros de dispersion. De este modo se tiene una primera estimación de las medias que denotamos $\hat{\mu}_{ij}^0$. Una vez estimadas las medias podemos estimar las dispersiones. Se considera la verosimilitud ajustada de Cox-Reid. Sea $l(\phi)$ la logverosimilitud dada por (eliminamos la referencia a la fila i pero obviamente esta función se considera para cada gen)

$$l(\phi) = \sum_{j=1}^{n} \log f_{NB}(y_j|\mu_j, \phi)$$

donde $f_{NB}(y_j : \mu_j, \phi)$ es la función de probabilidad de la binomial negativa con media μ_j y dispersión ϕ. Denotamos por D la matriz de diseño. La verosimilitud ajustada de Cox-Reid viene dada por

$$l_{CR}(\phi|\boldsymbol{\mu}, , \phi) = l(\phi) - \frac{1}{2}\log(det(D'WD)), \tag{10.32}$$

siendo $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)$ y W la matriz de pesos diagonal de los mínimos cuadrados iterativamente reponderados. Dado que la función de enlace es $g(\mu) = \log(\mu)$ y la varianza viene dada por $V(\mu|\phi) =$

[114] Siento esta redundancia necesaria para ser precisos.

[115] En [86, pág. 15] hay una interesante discusión sobre la elección de esta función.

[116] Por ser parámetros sobre una distribución de probabilidad para un parámetro realmente hemos de hablar de hiperparámetros en terminología bayesiana.

[117] Los propios datos son utilizados para estimar los hiperparámetros de la distribución a priori.

$\mu + \phi\mu^2$ entonces el j elemento de la diagonal es

$$w_{jj} = \frac{1}{g'(\mu_j)^2 V(\mu|\phi)} = \frac{1}{\frac{1}{\mu_j} + \phi}.$$

Se maximiza la función dada en 10.32 y obtenemos un estimador que denotamos por ϕ^{gw} para cada gen. Para el i-ésimo gen tenemos $(\bar{\mu}_i, \phi_i^{gw})$.

Tendencia de la dispersión Pretendemos estimar la función propuesta en 10.31. Se aplica iterativamente un modelo lineal generalizado utilizando la familia de distribuciones gamma. ¿Por qué iterativamente? Porque se van eliminando en cada iteración genes donde el ajuste es malo. ¿En qué sentido? En una iteración dada ajustamos el modelo lineal generalizado. Con el modelo ajustado en esa iteración tendremos una predicción y el valor original (ϕ_i^{gw}). Si el cociente del primero respecto del segundo no está en el intervalo $[10^{-4}, 15]$ es eliminado del estudio. El proceso continúa iterativamente hasta que la suma de los logaritmos de los cocientes de los nuevos coeficientes respecto de los antiguos es menor que 10^{-6}.

Distribución a priori sobre la dispersión Consideremos los residuos logarítmicos dados por

$$\log \phi_i^{gw} - \log \tilde{\phi}(\bar{\mu}_i).$$

La desviación estándar de estos residuos es estimada (evitando observaciones anómalas) utilizando el estimador basado en MAD (mediana de las desviaciones absolutas respecto de la mediana). Es decir con

$$s_{lr} = \frac{1}{\Phi^{-1}(\frac{3}{4})} MAD(\{\log \phi_i^{gw} - \log \tilde{\phi}(\bar{\mu}_i) : i = 1, \ldots, N\}). \quad (10.33)$$

Finalmente, la varianza de la distribución a priori es estimada con

$$\sigma_d^2 = \max\{s_{lr}^2 - \psi_1((n-p)/2), 0.25\}, \quad (10.34)$$

donde ψ_1 es la función trigamma; n, el número de muestras y p el número de covariables por muestra.

El estimador 10.34 no es adecuado cuando los grados de libertad residuales (número de muestras n menos número de parámetros a estimar p) son de tres o inferior. En este caso se propone un método basado en simulación que se puede consultar en [86, pág. 16].

Estimaciones de la dispersión. El estimador final de la dispersión ϕ_i se obtiene

$$\phi_i^{MAP} = argmax_\phi \left(l_{CR}\left(\phi; \bar{\mu}_{i\cdot}^0, y_{i\cdot}\right) + \Lambda_i(\phi) \right) \quad (10.35)$$

siendo

$$\Lambda_i(\phi) = \frac{-(\log \phi - \log \tilde{\phi}(\bar{\mu}_i))^2}{2\sigma_d^2}.$$

Outliers de dispersion. En el punto anterior hemos indicado cómo estimamos la dispersión para cada gen. No obstante, si un gen verifica que

$$\log \phi_i^{gw} > \log \tilde{\phi}(\bar{\mu}_i) + 2s_{lr}$$

entonces consideramos el gen como anómalo (outlier) en términos de dispersión. Para estos genes utilizamos como estimador de su dispersión no el propuesto en el apartado anterior sino ϕ_i^{gw}. Además estos genes no son utilizados cuando ajustamos $\tilde{\phi}$.

Estimación de los parámetros de la distribución a priori

Como asumimos que tienen medias nulas hemos de estimar las varianzas de las distribuciones normales asumidas.

Asumimos la siguiente distribución a priori sobre los coeficientes.

$$\beta_{ir} \sim N(0, \sigma_r^2). \tag{10.36}$$

Para el i-ésimo gen aplicamos el algoritmo (habitual) de los mínimos cuadrados iterativamente reponderados y obtenemos los estimadores máximo verosímiles β_{ir}^{MLE}. Consideramos un coeficiente r (excepto el correspondiente a la constante). Tendemos un valor estimado (máximo verosímil) de este coeficiente para cada gen. Ajustamos a esta distribución empírica una normal con media nula. Con objeto de obtener un estimador robusto (frente a estimadores anómalos) no utilizamos (como sería lo habitual) la varianza muestral observada sino que consideramos el cuantil empírico de orden $1-p$ de los $|\beta_{ir}^{MLE}|$, $q_{1-p}(|\beta_r|)$. Si considerados $Z_{1-p/2}$ el cuantil $1-p/2$ de una normal estándar entonces estimamos con $\sigma_r = \frac{q_{1-p}(|\beta_r|)}{Z_{1-p/2}}$. Se toma por defecto $p = 0.05$. Cuando se calculan los percentiles se eliminan aquellos que verifican $|\beta_{ir}| > \log_2 10$.

Estimadores de $\boldsymbol{\beta}_i$

. Vamos a estimar los coeficientes como los valores que maximizan la distribución a posteriori conocidos como MAP (maximum a posteriori).

Consideramos el logaritmo de la distribución a posteriori del vector $\boldsymbol{\beta}_i = (\beta_{i1}, \ldots, \beta_{ip})$. Obviamente es igual a la suma del logaritmo de la logverosimilitud del modelo lineal generalizado (que estamos asumiendo) más el logaritmo de la distribución a priori. Se estima $\boldsymbol{\beta}_i$ como el valor que maximiza

$$\sum_{j=1}^{n} f_{NB}(y_{ij}|\mu_j(\boldsymbol{\beta}), \phi_i) + \Lambda(\boldsymbol{\beta}) \tag{10.37}$$

siendo

$$\mu_j(\boldsymbol{\beta}) = s_{ij} e^{\sum_{r=1}^{p} y_{jr}\beta_r},\ \Lambda(\boldsymbol{\beta}) = \sum_{r=1}^{p} \frac{-\beta_r^2}{2\sigma_r^2},$$

y $\phi_i = \phi_i^{MAP}$ excepto para los genes que son outliers de dispersión en donde $\phi_i = \phi_i^{gw}$. La función 10.37 es maximizada utilizando el algoritmo de regresión ridge iterativamente reponderada ya que el término de penalización es del tipo que nos aparece en regresión ridge. En concreto en una iteración dada actualizamos los valores de los coeficientes con

$$\boldsymbol{\beta} \leftarrow (D^T W D + \boldsymbol{\lambda} I)^{-1} D^T W \boldsymbol{z},$$

donde $\lambda_r = \frac{1}{\sigma_r^2}$ y

$$z_j = \log \frac{\mu_j}{s_j} + \frac{y_j - \mu_j}{\mu_j}$$

siendo $\mu_j(\boldsymbol{\beta}) = s_j e^{\sum_{r=1}^{p} y_{jr}\beta_r}$ donde los valores de β_r son la estimación en la iteración actual.

Test de Wald

En un test de Wald se comparan los estimadores de los coeficientes β_{ir} con sus errores estándar estimados $SE(\beta_{ir}$ que corresponden con el elemento en posición (r, r) de la matriz de covarianzas de $\boldsymbol{\beta}_i$. Esta matriz viene dada por

$$cov(\boldsymbol{\beta}_i) = (D^T W D + \boldsymbol{\lambda} I)^{-1}(D^T W D)(D^T W D + \boldsymbol{\lambda} I)^{-1}.$$

El cociente del estimador con el error estándar tiene aproximadamente una distribución normal estándar bajo la hipótesis nula de que el coeficiente es nulo. Aplicando el test de Wald tendremos p-valores para el test bilateral de que el coeficiente no es nulo. Finalmente estos p-valores pueden ser ajustados mediante algún procedimiento como el método de Benjamini-Hochberg.

En el trabajo se discute la integración de un método de filtrado independiente basado en la media de los conteos normalizados y la adaptación a hipótesis nulas compuestas así como detalles adicionales que no comentamos. En esencia este es el método propuesto.

10.5.2 PRJNA218851

Vamos a analizar los datos `tamidata2::PRJNA218851`.

```
pacman::p_load(SummarizedExperiment,DESeq2)
```

Leemos los datos.

```
data(PRJNA218851,package="tamidata2")
```

Tenemos la covariable `Stage` que indica el tipo de tejido analizado. Tenemos tres muestras por individuo. Por tanto tenemos un factor intra sujeto con tres niveles. No vamos a utilizar este diseño. Simplemente vamos a considerar que tenemos tres grupos independientes de muestras.

Empezamos generando el objeto de clase `DESeqDataSet` a partir de nuestro objeto de clase `RangedSummarizedExperiment`.

```
dds = DESeqDataSet(PRJNA218851, design = ~Stage)
```

Eliminamos genes con conteos bajos.

```
keep = rowSums(counts(dds)) >=10
dds = dds[keep,]
```

Fijamos el nivel `control` como categoría de referencia.

```
dds$Stage <-relevel(dds$Stage, ref = "Normal")
```

Hacemos el estudio de la expresión diferencial.

```
dds = DESeq(dds)
```

Podemos evaluar el coeficiente correspondiente a la comparación entre el nivel `cancer` y el de referencia `control`.

```
resLFC2 = lfcShrink(dds, coef=2)
```

Ahora evaluamos los tests correspondientes al coeficiente que compara el nivel `metastasis` con el nivel de referencia `control`.

```
resLFC3 = lfcShrink(dds, coef=3)
```

10.6 Ejercicios

Ex. 19 — Para los datos `tamidata::PRJNA297664` se pìde:

1. Aplicar el procedimiento de comparación de proporciones utilizando edgeR::binomTest a las muestras 1 y 3.
2. Ajustar los p-valores obtenidos en el punto anterior por el método de Benjamini-Hochberg.
3. Una vez ajustados los p-valores determinar cuantos y cuales tienen un p-valor ajustado menor a 0.01.
4. Repetir los tres pasos anteriores comparando las muestras 2 y 5.
5. ¿Cuántos genes han sido detectados como significativos simultáneamente en los puntos 3 y 4?

Ex. 20 — Repetir el análisis realizado en 104 intentando comparar todos los pares dos a dos de la variable `colData(tcga_coad)[,"tissue_or_organ_of_origin"]`.

Ex. 21 — Utilizar el análisis contenido en 104. Se pide comparar los conteos originales con los normalizados. Para ello hacer la diferencia de los mismos y al vector resultante aplicarle el método `summary()`.

Parte III

Análisis de grupos de genes

Capítulo 11

Grupos de genes

11.1 Introducción

Hasta ahora hemos trabajado habitualmente a nivel de gen. Cada gen en cada muestra nos da un nivel de expresión. Obviamente esto supone que por muestra tenemos una muy alta dimensión. Un análisis de expresión diferencial supone que tenemos un contraste por gen. Muchos miles de contrastes supone esto. Si pretendemos clasificar la muestra en distintos fenotipos entonces la dimensión alta con la que trabajamos produce un sobreajuste y modelos que no son generalizables a otros datos. En ambos problemas no es bueno tener una dimensión tan alta. Podemos utilizar componentes principales para resumir el perfil de expresión de la muestra (§ 16). Otra opción puede ser trabajar no con genes individuales sino con *grupos de genes*. Si nos interesa la expresión diferencial entre condiciones entonces podemos valorar si el grupo de genes que nos interese, como grupo, se expresa diferencialmente. Si vamos a clasificar en fenotipos, podemos resumir la expresión de los genes en una muestra dentro de cada grupo. Reducimos el número de hipótesis a contrastar (expresión diferencial) o bien la dimensión del vector que utilizamos para clasificar (clasificación en fenotipo).

Obviamente podemos construir el grupo de genes que nos apetezca y evaluarlo. No parece algo muy lógico. Es de sobra conocido en la literatura estadística que un contraste de hipótesis es útil en la misma medida en que está correctamente formulado. Y formular correctamente el contraste supone conocimiento sobre el problema que abordamos. En este caso la comunidad científica ha propuesto y propone y propondrá clasificaciones que se han de utilizar para definir estos grupos.

Tres fuentes vamos a utilizar a la hora de considerar grupos de genes.

Gene Ontology: Gene Ontology Podemos utilizar las tres ontologías consideradas en Gene Ontology. El grupo de genes viene definido por todos aquellos genes que tienen este término.

KEGG: Kyoto Encyclopedia of Genes and Genomes Tenemos las rutas de señalización para muchos organismos.

MSigDB: Molecular Signatures Database En MSigDB [118] consideran hasta ocho formas distintas de construir estos grupos.

[118] The Molecular Signatures Database

DO: Disease ontology

DisGeNET: Disease Gene Network

wikiPathways

En http://www.genesetdb.auckland.ac.nz/haeremai.html tenemos una herramienta web que contiene a su vez a otras muchas bases de datos. Realiza análisis de sobre representación y está orientada a humanos, ratones y ratas.

Hay muchas formas de definir estos grupos. En cualquier caso en su definición interviene un conocimiento previo. No utilizamos los propios datos de expresión con los que posteriormente vamos a trabajar. No son grupos obtenidos por un análisis cluster previo.

En este tema tratamos de cómo construir grupos de genes utilizando R/Bioconductor. Es un tema de carácter muy técnico previo al que sigue en donde analizaremos la posible expresión diferencial de estos grupos.[119] Vamos a definir conjuntos de genes o colecciones de conjuntos de genes utilizando el paquete [101, GSEABase].

[119] Siempre nuestro interés principal.

```
library(GSEABase)
```

11.2 Homo sapiens

La opción más simple sería definirnos nuestro propio conjunto de genes. Para ello podemos utilizar la función `GSEAbase::GeneSet()`. Tomamos como ejemplo GSE20986 (§ 2.7.7).

```
data(gse20986,package="tamidata")
```

Supongamos que queremos construir un conjunto de genes formado por aquellos que ocupan las filas de la 345 a la 405 (sin ningún sentido práctico). Los índices los podemos obtener con

```
345:405
```

```
 [1] 345 346 347 348 349 350 351 352 353 354 355
[12] 356 357 358 359 360 361 362 363 364 365 366
[23] 367 368 369 370 371 372 373 374 375 376 377
[34] 378 379 380 381 382 383 384 385 386 387 388
[45] 389 390 391 392 393 394 395 396 397 398 399
[56] 400 401 402 403 404 405
```

Construimos el grupo y le damos nombre.

```
(egs = GeneSet(gse20986[345:405, ], setName = "Burjasot"))
```

Los genes tienen sus identificadores obtenidos de su anotación (mostramos los primeros).

```
head(geneIds(egs))
```

```
[1] "1552739_s_at" "1552740_at" "1552742_at"
[4] "1552743_at" "1552745_at" "1552747_a_at"
```

En este caso, como se obtuvo con Affymetrix GeneChip, los identificadores son los proporcionados por el fabricante y corresponde a los nombres de los conjuntos de sondas (§ 2). Podemos tener más información con

```
details(egs)
```

```
setName: Burjasot
geneIds: 1552739_s_at, 1552740_at, ..., 1552822_at (total: 61)
geneIdType: Annotation (hgu133plus2)
collectionType: ExpressionSet
setIdentifier: debian:5667:Sat May 24 08:06:49 2025:1
description:
organism: Homo sapiens
pubMedIds:
urls:
contributor:
setVersion: 0.0.1
creationDate:
```

En el ejemplo que acabamos de ver hemos creado un conjunto de genes a partir de un ExpresionSet pero podemos hacerlo de otros modos.

```
showMethods("GeneSet", inherited = FALSE)
```

```
Function: GeneSet (package GSEABase)
type="BroadCollection"
type="character"
type="ExpressionSet"
type="GeneIdentifierType"
type="GOCollection"
type="missing"
```

Podemos obtener la correspondencia de nuestros identificadores con otros tipos de identificadores, por ejemplo, los EntrezId:

```
mapIdentifiers(egs, EntrezIdentifier(), verbose = TRUE)
```

También podemos tener una colección de conjuntos, `GSEABase::GeneSetCollection()`. Quizás la mejor manera de definir conjuntos utilizando un `ExpresionSet` con su anotación es hacerlo utilizando Gene Ontology u otra base de datos. Por ejemplo, en el siguiente ejemplo construimos los conjuntos para nuestro ExpressionSet.

```
gsc = GeneSetCollection(gse20986, setType = GOCollection())
```

Podemos ver un resumen de los conjuntos.

```
gsc
```

```
GeneSetCollection
  names: GO:0000002, GO:0000012, ..., GO:2001070 (18488 total)
  unique identifiers: 1555591_at, 201917_s_at, ..., 216045_at (38483
      ↪ total)
  types in collection:
    geneIdType: AnnotationIdentifier (1 total)
    collectionType: GOCollection (1 total)
```

El correspondiente a `GO:0000122` sería

```
gsc[["GO:0000122"]]
```

```
setName: GO:0000122
geneIds: 1316_at, 1552338_at, ..., AFFX-HUMISGF3A/M97935_MB_at (total:
    ↪ 2340)
geneIdType: Annotation (hgu133plus2)
collectionType: GO
  ids: GO:0000122 (1 total)
  evidenceCode: EXP IDA IPI IMP IGI IEP HTP HDA HMP HGI HEP ISS ISO ISA
      ↪ ISM IGC IBA IBD IKR IRD RCA TAS NAS IC ND IEA
  ontology: CC MF BP
details: use 'details(object)'
```

Ahora vamos convertir estos mismos conjuntos a identificadores Entrez.

```
gsc = mapIdentifiers(gsc, EntrezIdentifier())
```

Podemos ver el primero de ellos.

```
head(geneIds(gsc),n=1)
```

```
$`GO:0000002`
 [1] "80119" "55186" "291" "4358" "1890"
 [6] "4205" "9361" "4976" "10000" "84275"
[11] "92667"
```

¿Cuántos elementos tiene cada uno de los conjuntos que hemos construido?

```
head(sapply(geneIds(gsc), length))
```

Podemos quedarnos con los dos grupos que tengan un cardinal mínimo, por ejemplo, por encima de 10.

```
gsc.filt = gsc[sapply(geneIds(gsc), length) > 10]
```

Son los siguientes grupos.

```
geneIds(gsc.filt)
```

11.3 Grupos con levadura

Veamos cómo construir grupos de genes utilizando los términos Gene Ontology en la Saccharomyces cerevisiae. Empezamos cargando el fichero de anotación.

```
pacman::p_load(org.Sc.sgd.db,GSEABase)
```

```
frame = toTable(org.Sc.sgdGO)
goframeData = data.frame(frame$go_id, frame$Evidence, frame$systematic_name)
goFrame = GOFrame(goframeData, organism = "Saccharomyces cerevisiae")
goAllFrame = GOAllFrame(goFrame)
gscSc_ORF = GeneSetCollection(goAllFrame, setType = GOCollection())
```

Habitualmente cuando queramos utilizar estos grupos tendremos que hacer dos cosas:

1. Quedarnos con aquellos genes que aparecen en nuestra plataforma.
2. Quedarnos posiblemente con grupos de genes con un tamaño mínimo.

Para ello vamos a utilizar la siguiente función.

```
subsettingGeneSet = function(gs0, fn0){
    geneIds(gs0) = geneIds(gs0)[is.element(geneIds(gs0), fn0)]
    gs0
}
```

Y ahora podemos modificar el GeneSetCollection de modo que nos quedamos con los genes que tenemos en nuestros datos `tamidata::gse6647` ↪ .

```
data(gse6647,package="tamidata")
```

```
gsc1 = sapply(gscSc_ORF, subsettingGeneSet, fn0 = featureNames(gse6647))
gsc2 = GeneSetCollection(gsc1)
```

Vamos a expresar la base de datos en identificadores ENTREZID. Empezamos considerando el total de genes que pertenecen a alguno de los grupos.

```
idsORF = unique(unlist(geneIds(gscSc_ORF)))
head(idsORF)
```

```
[1] "YHR194W" "YOR147W" "YAL048C" "YDL239C"
[5] "YDR150W" "YLL001W"
```

Obtenemos para todos estos identificadores `ORF` la correspondencia con los identificadores `ENTREZID`.

```
pacman::p_load(AnnotationDbi)
df = AnnotationDbi::select(org.Sc.sgd.db,keys=idsORF,
                           columns=c("ENTREZID"),
                           keytype="ORF")
```

Eliminamos posibles correspondencias múltiples entre los identificadores que hemos elegido: `ORF` de origen y `ENTREZID` de destino. Para ello definimos la siguiente función.

```
#' Choosing a pairs of correspondences
#' @description
#' Different gene identifiers (or proteins or ...) and we require
#' a unique correspondence
#' @param x \code{data.frame} with the different identifiers
#' @param coln \code{coln[,1]} is the original identifier and
#' \code{coln[,2]} is the new identifier
#' @export
multcorrespond = function(x,coln = c("PROBEID","ENTREZID")){
    x = x[,coln]
    x = na.omit(x)
    x = x[match(unique(x[,coln[1]]),x[,coln[1]]),]
    x = x[match(unique(x[,coln[2]]),x[,coln[2]]),]
    x
}
```

La aplicamos a nuestras correspondencias.

```
uc = multcorrespond(x=df,coln= c("ORF","ENTREZID"))
```

Y ahora, para cada grupo de genes, transformamos los identificadores.

```
coln= c("ORF","ENTREZID")
gscSc = lapply(1:length(gscSc),function(i)
    uc[match(geneIds(gscSc)[[i]],uc[,coln[1]]),coln[2]])
names(gscSc) = names(gscSc_ORF)
```

```
load(paste0(dirTamiData,"gscSc.rda"))
```

Podemos ver el primer grupo.

```
gscSc[1]
```

```
$`GO:0000001`
 [1] "851727" "856601" "850686" "854153" "854318"
 [6] "855412" "851249" "851359" "850887" "855318"
[11] "854079" "854504" "856249" "851558" "851532"
[16] "852825" "853033" "853146" "854748" "854867"
[21] "855094" "856198" "854668" "855645"
```

11.4 Grupos para GSE1397

Leemos los datos.

```
data(gse1397,package="tamidata")
```

Cargamos la anotación del `ExpressionSet` para poder formar grupos.

```
library(annotate)
annotation(gse1397)
```

```
[1] "hgu133a"
```

```
library(hgu133a.db)
```

Buscamos los conjuntos de genes utilizando Gene Ontology.

```
library(GSEABase)
```

```
gse1397.gsc = GeneSetCollection(gse1397,setType=GOCollection())
names(gse1397.gsc) = unlist(lapply(gse1397.gsc,setName))
```

¿Cuántos grupos tenemos definidos?

```
gsc = gse1397.gsc
length(gsc)
```

```
[1] 17556
```

Como vemos muchos. Sin embargo, la mayor parte de ellos tiene muy pocos genes. Podemos ver información del primer grupo.

```
(g1 = gsc[[1]])
```

```
setName: GO:0000002
geneIds: 201917_s_at, 201918_at, ..., 219393_s_at (total: 20)
geneIdType: Annotation (hgu133a)
collectionType: GO
  ids: GO:0000002 (1 total)
  evidenceCode: EXP IDA IPI IMP IGI IEP HTP HDA HMP HGI HEP ISS ISO ISA
      ↪  ISM IGC IBA IBD IKR IRD RCA TAS NAS IC ND IEA
  ontology: CC MF BP
details: use 'details(object)'
```

Y sus identificadores Affymetrix son

```
geneIds(g1)
```

```
 [1] "201917_s_at" "201918_at" "201919_at"
 [4] "202591_s_at" "202825_at" "203466_at"
 [7] "204858_s_at" "208328_s_at" "209017_s_at"
[10] "212213_x_at" "212214_at" "212535_at"
[13] "212607_at" "212609_s_at" "214060_at"
[16] "214306_at" "214684_at" "214821_at"
[19] "217497_at" "219393_s_at"
```

Sus identificadores Entrez o Ensembl son

```
unlist(mget(geneIds(g1),hgu133aENTREZID))
```

```
201917_s_at 201918_at 201919_at 202591_s_at
   "55186" "55186" "55186" "6742"
  202825_at 203466_at 204858_s_at 208328_s_at
     "291" "4358" "1890" "4205"
209017_s_at 212213_x_at 212214_at 212535_at
    "9361" "4976" "4976" "4205"
  212607_at 212609_s_at 214060_at 214306_at
   "10000" "10000" "6742" "4976"
  214684_at 214821_at 217497_at 219393_s_at
    "4205" "291" "1890" "10000"
```

```
unlist(mget(geneIds(g1),hgu133aENSEMBL))
```

```
     201917_s_at 201918_at
"ENSG00000114120" "ENSG00000114120"
       201919_at 202591_s_at1
"ENSG00000114120" "ENSG00000106028"
    202591_s_at2 202825_at
"ENSG00000262771" "ENSG00000151729"
       203466_at 204858_s_at
"ENSG00000115204" "ENSG00000025708"
     208328_s_at 209017_s_at
"ENSG00000068305" "ENSG00000196365"
     212213_x_at 212214_at
"ENSG00000198836" "ENSG00000198836"
       212535_at 212607_at1
"ENSG00000068305" "ENSG00000117020"
      212607_at2 212609_s_at1
"ENSG00000275199" "ENSG00000117020"
    212609_s_at2 214060_at1
"ENSG00000275199" "ENSG00000106028"
      214060_at2 214306_at
"ENSG00000262771" "ENSG00000198836"
       214684_at 214821_at
"ENSG00000068305" "ENSG00000151729"
       217497_at 219393_s_at1
"ENSG00000025708" "ENSG00000117020"
    219393_s_at2
"ENSG00000275199"
```

La mayor parte de estos grupos son muy pequeños. Lo siguiente nos da el tamaño de estos grupos.

```
head(table(sapply(geneIds(gsc),length)))
```

```
   1 2 3 4 5 6
2914 2045 1649 1332 1003 788
```

De tamaño uno tenemos

```
sum(sapply(geneIds(gsc),length) == 1)
```

```
[1] 2914
```

Nos quedamos con aquellos grupos que, al menos, tienen 50 genes.

```
gse1397.gsc.filt = gsc[which(sapply(geneIds(gsc),length) > 50)]
```

11.5 Grupos utilizando anotación

11.5.1 Grupos GO para levadura

```
library(org.Sc.sgd.db)
frame = toTable(org.Sc.sgdGO)
goframeData = data.frame(frame$go_id, frame$Evidence, frame$systematic_name)
goFrame = GOFrame(goframeData, organism = "Saccharomyces cerevisiae")
goAllFrame = GOAllFrame(goFrame)
gscSc = GeneSetCollection(goAllFrame, setType = GOCollection())
save(gscSc,file=paste0(dirTamiData,"gscSc.rda"))
```

En lo que sigue será frecuente que no queramos utilizar todos los grupos que hemos construido. ¿Cómo quedarnos con aquellos que tienen al menos un número dado de genes? Por ejemplo, quedarnos con los grupos con 10 o más genes. El siguiente código lo hace.

```
gruposGrandes = which(sapply(gscSc,length) > 10)
gsc1 = gscSc[gruposGrandes]
```

Podemos ver el primer grupo.

```
gsc1[[1]]
```

```
 [1] "851727" "856601" "850686" "854153" "854318"
 [6] "855412" "851249" "851359" "850887" "855318"
[11] "854079" "854504" "856249" "851558" "851532"
[16] "852825" "853033" "853146" "854748" "854867"
[21] "855094" "856198" "854668" "855645"
```

11.5.2 Grupos GO para humanos

En esta sección construimos los grupos de genes según Gene Ontology para humanos. Para otros organismos sería similar reemplazando el paquete [**R-org.Hs.eg.db**] por el correspondiente al organismo. El código es el siguiente.

```
library("org.Hs.eg.db")
frame = toTable(org.Hs.egGO)
goframeData = data.frame(frame$go_id, frame$Evidence, frame$gene_id)
goFrame = GOFrame(goframeData, organism = "Homo sapiens")
goAllFrame = GOAllFrame(goFrame)
gscHs = GeneSetCollection(goAllFrame, setType = GOCollection())
save(gscHs,file = paste0(dirTamiData,"gscHs.rda"))
```

11.5.3 Grupos para Arabidopsis thaliana

Utilizamos el paquete [28].

```
pacman::p_load("ath1121501.db")
frame = toTable(org.At.tairGO)
goframeData = data.frame(frame$go_id, frame$Evidence, frame$gene_id)
goFrame = GOFrame(goframeData, organism = "Arabidopsis")
goAllFrame = GOAllFrame(goFrame)
gscAt = GeneSetCollection(goAllFrame, setType = GOCollection())
gscAt = geneIds(gscAt)
save(gscAt,file = paste0(dirTamiData,"gscAt.rda"))
```

11.6 Utilizando EnrichmentBrowser

Vamos a utilizar para obtener colecciones de genes la función `EnrichmentBrowser` ↪ `::getGenesets()`.

```
pacman::p_load(EnrichmentBrowser)
```

Podemos bajarnos todos las rutas de Gene Ontology para un organismo dado. Empezamos con humanos.

```
hsaGO = getGenesets(org="hsa",onto="BP")
save(hsaGO,file=paste0(dirTamiData,"hsaGO.rda"))
```

¿Qué tenemos?

```
class(hsaGO)
```

```
[1] "list"
```

Podemos ver sus nombres.

```
head(names(hsaGO))
```

```
[1] "GO:0000002_mitochondrial_genome_maintenance"
[2] "GO:0000012_single_strand_break_repair"
[3] "GO:0000017_alpha-glucoside_transport"
[4] "GO:0000018_regulation_of_DNA_recombination"
[5] "GO:0000019_regulation_of_mitotic_recombination"
[6] "GO:0000022_mitotic_spindle_elongation"
```

Y los genes que componen uno determinado con su código Entrez.

```
hsaGO[[3]]
```

```
[1] "6523" "6523" "6523" "6524"
```

```
names(hsaGO[3])
```

```
[1] "GO:0000017_alpha-glucoside_transport"
```

O bien con

```
hsaGO$"GO:0000012_single_strand_break_repair"
```

```
 [1] "1161" "2074" "3981"
 [4] "7141" "7374" "7515"
 [7] "23411" "54840" "54840"
[10] "54840" "55247" "55775"
[13] "55775" "200558" "100133315"
```

```
hsaGO[["GO:0000012_single_strand_break_repair"]]
```

```
 [1] "1161" "2074" "3981"
 [4] "7141" "7374" "7515"
 [7] "23411" "54840" "54840"
[10] "54840" "55247" "55775"
[13] "55775" "200558" "100133315"
```

11.7 Utilizando DOSE

```
pacman::p_load(clusterProfiler,DOSE,org.Hs.eg.db)
data(geneList, package="DOSE")
gene = names(geneList)[abs(geneList) > 2]
head(gene)
ggo = groupGO(gene = gene,
              OrgDb = org.Hs.eg.db,
              ont = "CC",
              level = 3,
              readable = TRUE)
```

```
head(ggo)
```

```
                ID Description
GO:0000133 GO:0000133 polarisome
GO:0000417 GO:0000417 HIR complex
GO:0000796 GO:0000796 condensin complex
GO:0000808 GO:0000808 origin recognition complex
GO:0000930 GO:0000930 gamma-tubulin complex
GO:0000939 GO:0000939 inner kinetochore
         Count GeneRatio geneID
GO:0000133 0 0/207
GO:0000417 0 0/207
GO:0000796 2 2/207 NCAPH/NCAPG
GO:0000808 0 0/207
GO:0000930 0 0/207
GO:0000939 2 2/207 CENPM/CENPN
```

11.8 Ejercicios

Ex. 22 — Construir los grupos basados en Gene Ontology para el ratón (Mus musculus).

Ex. 23 — Construir los grupos basados en Gene Ontology y en KEGG para las cepas de la E. coli que puedas.

Ex. 24 — Pretendemos ver cómo construir un informe en html para un grupo de genes. Se pide:

1. Utilizando `EnrichmentBrowser::getGenesets` construir los grupos correspondientes a humanos en la ontología de Gene Ontology correspondiente a funciones moleculares.
2. Utilizando los nombres de la lista que nos devuelve construir un **`data.frame`** que tenga por primera columna los identificadores del grupo y por segunda columna la descripción.
3. Generar las direcciones URL a partir de los identificadores de grupo.
4. Utilizando [71] generar un informe en html en donde la primera columna del informe contenga los URL correspondientes a los grupos y la segunda columna sea la descripción de los mismos.
5. Repetir los apartados del 1 al 4 para las ontologías de Gene Ontology correspondientes a procesos biológicos y a componentes celulares.

Ex. 25 — Repetir los apartados del 1 al 4 del ejercicio 25 para los grupos KEGG.

Capítulo 12

Test de Fisher unilateral

En este tema mostramos la aplicación del test de Fisher a lo que se conoce como análisis de sobre representación. Una presentación más técnica la tenemos en § 15.5.4.

12.1 Test de Fisher

Por alguna razón (que desconozco) a este procedimiento se le llama en numerosas publicaciones de perfil biológico *test hipergeométrico*. [1] En lo tratado hasta este momento hemos obtenido una ordenación de los genes. Esta lista la hemos estudiado pretendiendo que cuando mayor sea la expresión diferencial del gen este aparezca antes en la lista. De este modo el primer gen es el *marginalmente* (o si se prefiere individualmente) tiene una mayor expresión diferencial y así sucesivamente. De hecho, el p-valor no es más que una medida de esa diferenciación. Una expresión muy utilizada en la literatura es que que hay una asociación entre el gen (su expresión) y el fenotipo.[120] De alguna forma el p-valor que obtenemos en cada test es una cuantificación de la asociación gen-fenotipo. Luego modificamos estos p-valores de forma que se tiene en cuenta todos los genes que simultáneamente se están estudiando. Obtenemos de este modo unos p-valores ajustados. Finalmente, tanto los p-valores originales como los ajustados no dejan de ser cuantificaciones marginales de la asociación gen-fenotipo. Cuando hemos fijado una tasa de error (FDR o FWER) lo que hacemos es fijar un punto de corte. En esa lista que hemos construido decidimos (con algún criterio de error) en qué punto de la lista cortamos. Los genes que están antes del punto de corte se consideran significativos y los que siguen no. Reducimos nuestra información a un si (gen significativo o que tiene expresión diferencial o que hay asociación gen-fenotipo[2] o un no (no es significativo, no hay expresión diferencial o no hay asociación gen-fenotipo).

[120] La expresión fenotipo se usa de un modo muy amplio porque podemos estar hablando de características fenotípicas o simplemente un diseño experimental con factores temporales o diferentes temperaturas.

Ya tenemos ese conjunto de genes significativos. Incluso hemos visto cómo generar unos enlaces para que gen a gen examinemos alguna base de datos online. Es claro que el investigador puede ir generando hipótesis sobre qué indica esta lista. Pero claramente no es una labor simple. Una posible ayuda puede ser utilizar información previamente generada por la comunidad científica sobre grupos de genes. Grupos que indican que

[1] La distribución del estadístico de contraste es la distribución hipergeométrica. Hasta ahí llego. Lo que no entiendo es el cambio de nombre ya que la denominación test de Fisher está más que consolidada y debiera de utilizarse.

[2] A gusto del consumidor.

Tabla 12.1: S_0 indica el grupo de genes significativos (grupo predefinido) y $G \setminus S_0$ su complementario respecto del universo de genes considerado G. S_1 indica el conjunto de genes contra el cual comparamos.

	S_1	$S_1^c = G \setminus S_1$	
S_0	n_{11}	n_{12}	$n_{1\cdot}$
$S_0^c = G \setminus S_0$	n_{21}	n_{22}	$n_{2\cdot}$
	$n_{\cdot 1}$	$n_{\cdot 2}$	N

tienen relación con una misma función, o que están localizados próximos en un mismo cromosoma. En fin, grupos con sentido (biológico).

Un poco de notación que nunca es mala. Sea G es el conjunto de genes considerado. ¿Quién es este conjunto? En un estudio con microarrays uno diría que es el conjunto de genes que se está explorando. Sin embargo, un chip suele estar diseñado para observar tanto genes como se pueda y no hay una selección previa. Quizás no sea muy razonable considerar el conjunto total de genes este. En otros casos no es así. Lo fundamental es darse cuenta que lo que hacemos depende de un modo esencial del conjunto total de genes considerado o universo. S_0 ($\subset G$) el conjunto de genes que *nuestro* estudio ha indicado como significativo y S_1 un conjunto de genes predefinido (misma función, misma localización). Un primer problema es definir los conjuntos S_1 con los que comparar.[121] Una vez definidos *nuestro conjunto* (S_0), el conjunto con el que queremos comparar (S_1) y el conjunto total de genes considerado (G) la situación que se presenta la tenemos reflejada en la tabla 12.1. En esta tabla n_{11} indica el número de genes que están tanto en S_0 como en S_1, también tendremos n_{12} genes que están en S_0 pero no en S_1, n_{21} en S_1 pero no en S_0 y, finalmente, n_{22} que no están ni en S_0 ni en S_1. Suponemos que el total de genes (nuestro universo de genes) G tiene un total de N genes.

[121] § 11.

En la tabla 12.1 consideramos dados los totales de la fila y la columna, en otras palabras, consideramos fijos los valores de $n_{1\cdot}$ y $n_{2\cdot}$ (totales de fila) así como los valores de $n_{\cdot 1}$ y $n_{\cdot 2}$ (totales de columna). Asumiendo fijos estos totales de fila y columna: ¿cuál es la probabilidad de observar la tabla 12.1? Utilizando argumentos combinatorios la respuesta es la siguiente: Si denotamos N_{11} el número *aleatorio* de genes en común entonces, bajo la hipótesis de que no hay ningún tipo de asociación entre fila y columna, entonces la probabilidad sería

$$P(N_{11} = n_{11}) = \frac{\binom{n_{1\cdot}}{n_{11}}\binom{n_{2\cdot}}{n_{\cdot 1}-n_{11}}}{\binom{N}{n_{\cdot 1}}}.$$

Supongamos que estamos contrastando la posible sobrerepresentación entonces, bajo la hipótesis de independencia (condicionada a las marginales), rechazaríamos la hipótesis de independencia para un valor mayor o igual al observado por lo que el p-valor sería la suma de las probabilidades siguientes

$$p = P(N_{11} \geq n_{11}) = \sum_{t=n_{11}}^{\min\{n_{1\cdot}, n_{\cdot 1}\}} \frac{\binom{n_{1\cdot}}{t}\binom{n_{2\cdot}}{n_{\cdot 1}-t}}{\binom{N}{n_{\cdot 1}}}.$$

Supongamos que hemos observado la tabla 12.2 y pretendemos saber si el solapamiento entre ambos conjuntos de genes es mayor que el esperable por el puro azar aunque no estén asociados ambos conjuntos.

Podemos utilizar el test exacto de Fisher direccional (o unilateral o de una cola).

Tabla 12.2: S_0 (S) indica el grupo de genes significativos (grupo predefinido) y S_0 (S^c) su complementario. S_1 indica el conjunto de genes contra el cual comparamos.

	S_1	S_1^c	
S_0	30	40	70
S_0^c	120	156	276
	150	196	346

```
conteos = matrix(c(30,120,40,156),ncol=2)
fisher.test(conteos,alternative = "greater")
```

```
        Fisher's Exact Test for Count Data

data:  conteos
p-value = 0.589
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 0.6018802       Inf
sample estimates:
odds ratio
 0.9750673
```

¿Y si valoramos una baja representación?

```
fisher.test(conteos,alternative = "less")
```

```
        Fisher's Exact Test for Count Data

data:  conteos
p-value = 0.5179
alternative hypothesis: true odds ratio is less than 1
95 percent confidence interval:
 0.000000 1.571751
sample estimates:
odds ratio
 0.9750673
```

Observamos que en ambas salidas nos muestra el cociente de los odds. Un valor del cociente de odds mayor que la unidad indica sobre expresión mientras que un valor menor a la unidad indica una expresión menor a la esperable bajo independencia.

12.2 Sobre la elección del universo de genes

Pretendemos evaluar el efecto que tiene la elección del universo de genes. ¿Qué efecto tiene en nuestro análisis el universo de genes, el conjunto G total de genes que estamos utilizando? Consideremos los datos de la tabla 12.3. ¿Qué describe la tabla? Supongamos que tenemos dos conjuntos de genes dados, S_0 y S_1. Dado este par de grupos de genes tenemos el número de genes en los dos y en uno de ellos pero no en el otro. Si miramos la tabla 12.3 significa que tenemos perfectamente definidas tres entradas de la tabla. Pero si suponemos que vamos incrementando nuestro universo de genes (sin incluir ningún gen adicional en S_0 o S_1) entonces la entrada n_{22} en la tabla 12.3 será cada vez mayor. En concreto hemos supuesto que $n_{22} = 156 + k$, es decir, el conteo original más los

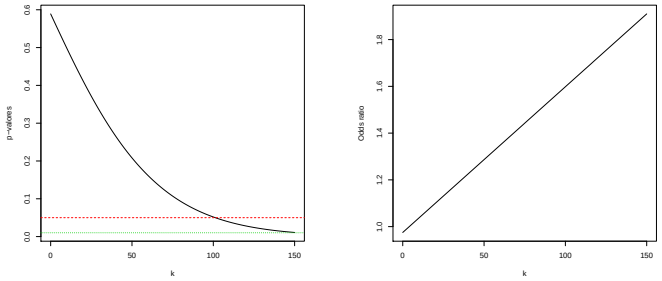
Tabla 12.3: S_0 (respectivamente S_1) indica el grupo de genes significativos (el grupo predefinido) y $G \setminus S_0$ ($G \setminus S_1$) su complementario. El universo de genes crece y suponemos que añadimos k genes.

	S_1	$G \setminus S_1$	
S_0	30	40	70
$G \setminus S_0$	120	156 + k	276 + k
	150	196 + k	346 + k

k genes que vamos incorporando al universo de genes. Vamos a tomar un valor de k creciente y representaremos el p-valor del test de Fisher unilateral. Vemos cómo conforme el valor de k crece el p-valor decrece (figura 12.1(a)) y el cociente de odds crece (figura 12.1(b)). En definitiva la interpretación tanto del p-valor como del cociente de odds depende del universo de genes que estamos considerando.

En la figura 12.1(a) (respectivamente en la figura 12.1(b)) tenemos el p-valor del test de Fisher unilateral correspondiente a la tabla 12.3 (respectivamente el cociente de odds) cuando el valor de k va 0 a 100. En trazo rojo discontinuo representamos la línea horizontal para un valor de ordenada igual a 0.05. La línea verde punteada corresponde con una ordenada de 0.01.

Vemos cómo *el incremento del universo de genes tiene como consecuencia que el p-valor del test de Fisher unilateral decrezca*.



(a) (b)

Figura 12.1: a) p-valores del test de Fisher unilateral. b) Cocientes de odds.

12.3 Utilizando Category y GOstats

En esta sección utilizamos los paquetes [56, GOstats] (ver también [49]) y [55, Category].

12.3.1 GSE1397

Leemos los datos normalizados.

```
pacman::p_load(Biobase)
data(gse1397,package = "tamidata")
eset = gse1397
y = pData(eset)[,"type"]
remove(gse1397) ## Ahorramos memoria
```

Determinamos grupo de genes significativos.

```
pacman::p_load(genefilter,multtest)
tt = rowttests(eset,y)
p0 = tt$p.value
p1 = mt.rawp2adjp(p0, "BH")
orden.original = order(p1$index)
p.BH = p1$adjp[orden.original,2]
significativos = which(p.BH < 0.05)
```

¿Qué anotación tienen?[122]

[122] Es una base de datos asociada a un chip.

```
annotation(eset)
```

```
[1] "hgu133a"
```

Cargamos el paquete de anotación correspondiente.

```
library(hgu133a.db)
```

¿Tenemos un solo identicador Gene Ontology para cada gen?

```
G1.entreizd = unlist(mget(featureNames(eset), hgu133aENTREZID))
anyDuplicated(G1.entreizd)
```

```
[1] 30
```

Hay duplicados. Nos quedamos, para cada gen, con el conjunto de sondas que nos da la máxima variabilidad, por ejemplo, el valor más grande del rango intercuartílico.

```
eset.iqr = apply(exprs(eset), 1, IQR)
uniqGenes = findLargest(featureNames(eset), eset.iqr, "hgu133a")
eset1 = eset[uniqGenes, ]
```

Y ahora construimos el universo de genes y comprobamos que no hay duplicidades.

```
G2.entrezid = unlist(mget(featureNames(eset1), hgu133aENTREZID))
anyDuplicated(G2.entrezid)
```

```
[1] 0
```

Determinamos la identificación en la misma base de datos de los genes declarados significativos.

```
seleccionados = unlist(mget(featureNames(eset[significativos,]),
    hgu133aENTREZID))
```

Vamos a aplicar un test de Fisher unilateral para los grupos definidos de acuerdo con Gene Ontology. Cargamos paquetes necesarios.

```
pacman::p_load(GO.db,Category,GOstats)
```

Y realizamos los tests.[123]

[123] En http://geneontology.org/ tenemos una aplicación en línea que nos realiza un análisis similar.

```
params = new("GOHyperGParams", geneIds = seleccionados,
             universeGeneIds = G2.entrezid,
             annotation = annotation(eset), ontology = "BP",
             pvalueCutoff = 0.001,conditional = FALSE,
             testDirection = "over")
overRepresented = hyperGTest(params)
```

Finalmente para visualizar los resultados guardamos los resultados en un fichero y lo vemos con el navegador.

```
htmlReport(overRepresented, file = "GSE1397overRepresented.html")
browseURL("GSE1397overRepresented.html")
```

También podemos ver un resumen.

```
head(summary(overRepresented))
```

Con el siguiente código obtenemos un grafo donde los vértices corresponden a los grupos definidos por la categorías Gene Ontology significativas. Las aristas nos muestran la estructura de la base de datos que es un grafo acíclico dirigido.

```
library(Rgraphviz)
plot(goDag(overRepresented))
```

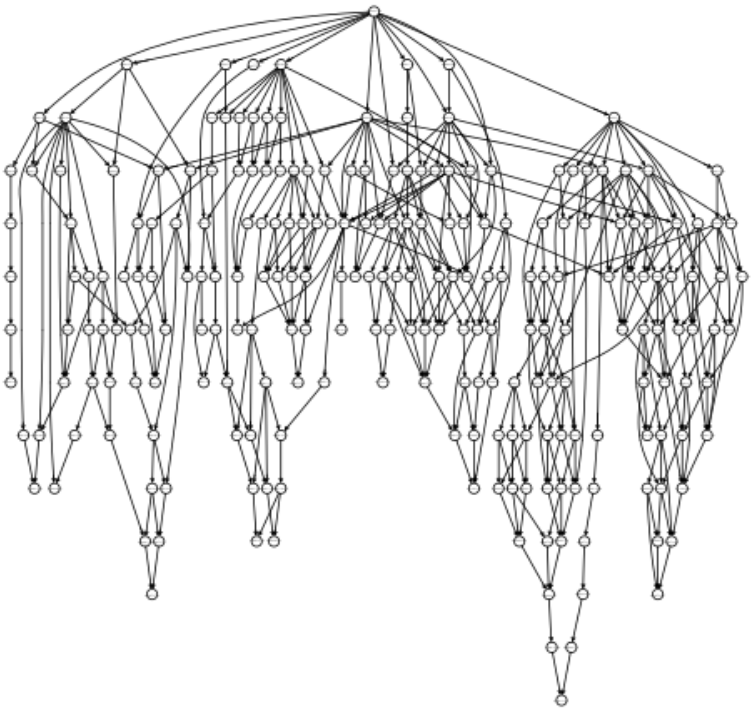


Figura 12.2: Grafo Gene Ontology con las grupos significativos.

12.3.2 GSE20986

Leemos los datos normalizados y lo renombramos.

```
data(gse20986, package = "tamidata")
```

En lo que sigue necesitamos la anotación de este `ExpressionSet`.

```
Biobase::annotation(gse20986)
```

```
[1] "hgu133plus2"
```

Cargamos el paquete con la base de datos correspondiente.

```
library("hgu133plus2.db")
```

Lo primero: **definir el universo de genes**. ¿Cómo? Una propuesta razonable[3] podría ser aplicar un filtrado no específico y que el universo de

[3] Y nada más que esto razonable y tan razonable como muchas otras.

genes sean los restantes.

```
library(genefilter)
gse.filt = genefilter::nsFilter(gse20986, var.func = IQR, var.cutoff = 0.6,
                                require.GOBP = TRUE)$eset
```

¿Cuántos genes nos quedan?

```
dim(gse.filt)
```

```
Error: objeto 'gse.filt' no encontrado
```

¿Tenemos un solo identicador Gene Ontology para cada gen?

```
G1.entreizd = unlist(mget(featureNames(gse.filt), hgu133plus2ENTREZID))
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'mget': error al
    ↪ evaluar el argumento 'object' al seleccionar un método para la
    ↪ función 'featureNames': objeto 'gse.filt' no encontrado
```

```
anyDuplicated(G1.entreizd)
```

```
[1] 30
```

Vemos que no hay duplicidades en los datos filtrados. Sin embargo, si consideramos como universo de genes todos los del chip entonces sí que nos encontramos con duplicidades.

```
G2.entrezid = unlist(mget(featureNames(gse20986), hgu133plus2ENTREZID))
anyDuplicated(G2.entrezid)
```

```
[1] 19
```

La idea sería quedarnos para cada gen con el conjunto de sondas que nos da la máxima variabilidad. Por ejemplo, con el grupo de sondas con el mayor rango intercuartílico.

```
gse.iqr = apply(exprs(gse20986), 1, IQR)
uniqGenes = findLargest(featureNames(gse20986), gse.iqr, "hgu133plus2")
gse.filt2 = gse20986[uniqGenes, ]
```

Y ahora construimos el universo de genes y comprobamos que no hay duplicidades.

```
G2.entrezid = unlist(mget(featureNames(gse.filt2), hgu133plus2ENTREZID))
anyDuplicated(G2.entrezid)
```

```
[1] 0
```

Aplicamos el procedimiento de expresión diferencial utilizado en el ejemplo 6.13.

```
library(multtest)
gse.aov = rowFtests(gse.filt2, pData(gse20986)[,"tissue"])
p.originales = gse.aov[, 2]
p.BH = mt.rawp2adjp(p.originales, "BH")
pvalores = p.BH$adjp[p.BH$index, 2]
sig.1234 = which(pvalores < 0.001)
selected = unlist(mget(featureNames(gse.filt2[sig.1234,]), hgu133plus2ENTREZID))
```

Vamos a aplicar un test de Fisher unilateral para los grupos definidos de acuerdo con Gene Ontology. Cargamos paquetes necesarios.

```
pacman::p_load(GO.db,Category,GOstats)
```

Y realizamos los tests.

```
params = new("GOHyperGParams", geneIds = selected,
             universeGeneIds = G2.entrezid,
             annotation = annotation(gse.filt2), ontology = "BP",
             pvalueCutoff = 0.01,
             conditional = FALSE, testDirection = "over")
overRepresented = hyperGTest(params)
```

Finalmente para visualizar los resultados guardamos los resultados en un fichero y lo vemos con el navegador.

```
fl = tempfile()
htmlReport(overRepresented, file = fl)
browseURL(fl)
```

12.3.3 ALL

Esta sección se reproduce el análisis propuesto en http://www.bioconductor.org/help/course-materials/2009/SSCMay09/gsea/HyperG_Lecture.pdf. Utilizamos los datos [83, ALL]. En § 2.7.2 se indica cómo conseguir los datos `bcrneg` en donde hemos seleccionado algunas muestras. Los datos los tenemos en `bcrneg`. Veamos número de genes y muestras.

```
dim(bcrneg)
```

```
Features Samples
  12625 79
```

Realizamos un filtrado no específico quitando aquellos cuyo rango intercuartílico esté por debajo de la mediana (de los rangos intercuartílicos observados). Además se requiere que el gen tenga anotación en Gene Ontology. Esto lo pedimos con el argumento `require.GOBP = TRUE`.[4]

```
bcrneg.filt = nsFilter(bcrneg, var.cutoff = 0.5, require.GOBP = TRUE)$eset
```

Veamos qué nos queda.

```
dim(bcrneg.filt)
```

```
Features Samples
   4158 79
```

Vamos a generar una lista de genes significativos de un modo muy básico[5]. Calculamos el p-valor del test de la t y nos quedamos con los genes con un p-valor por debajo de 0.05.

```
fac0 = pData(bcrneg.filt)[, "mol.biol"]
fac0 = factor(fac0) ## Quitamos categorías vacías
rtt = rowttests(bcrneg.filt, fac0)
rttPrb = rtt$p.value
tThresh = rttPrb < 0.05
```

Le damos nombres a las componentes del vector utilizando los identificadores de Affymetrix GeneChip.

[4] En concreto se pide su anotación en la ontología *Biological Process*.
[5] Y tampoco muy recomendable.

Tabla 12.4: Tabla de contingencia para el término GO:0006468.

	InGO	
Selected	FALSE	TRUE
FALSE	3101	162
TRUE	658	30

```
names(rttPrb) = featureNames(bcrneg.filt)
```

Guardamos estos identificadores en `ids`.

```
ids = featureNames(bcrneg.filt)
```

Guardamos también las correspondencias con Entrez.

```
map = hgu95av2ENTREZID
```

Definimos quién es nuestro universo. En nuestro caso todos los genes que intervenían en nuestro estudio y de los cuales teníamos su anotación (hay sondas de control que desaparecen con el filtrado que acabamos de hacer).

```
universe = unlist(mget(ids, map))
selected = unlist(mget(ids[tThresh], map))
```

Elegimos un grupo utilizando un término de Gene Ontology. Primero cargamos el paquete [29, GO.db].

```
library(GO.db)
```

Nos fijamos por ejemplo en el término siguiente

```
GOTERM[["GO:0006468"]]
```

```
GOID: GO:0006468
Term: protein phosphorylation
Ontology: BP
Definition: The process of introducing a
   phosphate group on to a protein.
Synonym: protein amino acid phosphorylation
```

La tabla 12.4 muestra los conteos observados.

Cargamos los paquetes [55, Category] y [56, GOstats].

```
pacman::p_load(Category,GOstats)
```

```
params = new("GOHyperGParams", geneIds = selected,
             universeGeneIds = universe,
             annotation = annotation(bcrneg.filt),
             ontology = "BP", pvalueCutoff = 0.001,
             conditional = FALSE, testDirection = "over")
overRepresented = hyperGTest(params)
```

Veamos el resumen.

```
head(summary(overRepresented), n = 3)
```

Guardamos los resultados en un fichero HTML y lo vemos.

```
fl = tempfile()
htmlReport(overRepresented, file = fl)
browseURL(fl)
```

12.4 EnrichmentBrowser::sbea

Vamos a realizar un análisis de sobre representación utilizando `EnrichmentBrowser ↪ ::sbea`.

```
pacman::p_load(EnrichmentBrowser,Biobase,SummarizedExperiment)
data(gse21942,package="tamidata")
```

Hemos transformar el `ExpressionSet` en un `SummarizedExperiment`.

```
se21942 = makeSummarizedExperimentFromExpressionSet(gse21942)
```

```
Error in SummarizedExperiment(assays = assays, rowRanges = rowRanges,
    ↪ : the rownames and colnames of the supplied
  assay(s) must be NULL or identical to those of
  the RangedSummarizedExperiment object (or
  derivative) to construct
```

Podemos ver que los genes vienen identificados por los correspondientes al fabricante `AffyID` o `PROBEID`.

```
head(rowData(se21942),n=1)
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'head': error al
    ↪ evaluar el argumento 'x' al seleccionar un método para la
    ↪ función 'rowData': objeto 'se21942' no encontrado
```

```
se21942=probe2gene(se21942)
```

```
Error: objeto 'se21942' no encontrado
```

¿Qué hemos hecho? Modificar los identificadores de los genes sustituyéndolos por los `ENTREZID`.

```
head(rowData(se21942))
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'head': error al
    ↪ evaluar el argumento 'x' al seleccionar un método para la
    ↪ función 'rowData': objeto 'se21942' no encontrado
```

Introducir una variable fenotípica `GROUP` con valores 0 y 1. Vamos a realizar un análisis de expresión diferencial utilizando el modelo Limma.

```
se21942 = deAna(expr = se21942) ## t-test moderados
```

```
Error: objeto 'se21942' no encontrado
```

```
head(rowData(se21942))
```

```
Error in h(simpleError(msg, call)): error al evaluar el argumento 'x'
    ↪ al seleccionar un método para la función 'head': error al
    ↪ evaluar el argumento 'x' al seleccionar un método para la
    ↪ función 'rowData': objeto 'se21942' no encontrado
```

Utilizamos los grupos de genes Gene Ontology.

```
hsaGO = getGenesets("hsa",onto="BP") ## Biological processes
save(hsaGO,file=paste0(dirTamiData,"hsaGO.rda"))
```

Realizamos un análisis de sobre representación con el test de Fisher unilateral.

```
se21942.oraGO= sbea(method="ora", se=se21942, gs=hsaGO,
                    perm=0, alpha=0.05)
```

```
save(se21942.oraGO,file=paste0(dirTamiData,"se21942.oraGO.rda"))
```

```
load(paste0(dirTamiData,"se21942.oraGO.rda"))
```

Los resultados obtenidos son

```
gsRanking(se21942.oraGO)
```

```
DataFrame with 379 rows and 4 columns
                GENE.SET NR.GENES NR.SIG.GENES
              <character> <numeric> <numeric>
1 GO:1902600_proton_tr.. 161 68
2 GO:0006120_mitochond.. 42 25
3 GO:0007042_lysosomal.. 24 17
4 GO:0042776_proton_mo.. 59 30
5 GO:0006974_DNA_damag.. 291 99
... ... ... ...
375 GO:1901857_positive_.. 10 5
376 GO:1904380_endoplasm.. 10 5
377 GO:2001241_positive_.. 10 5
378 GO:0009306_protein_s.. 52 17
379 GO:0030218_erythrocy.. 56 18
        PVAL
    <numeric>
1 4.97e-09
2 1.43e-07
3 4.09e-07
4 9.42e-07
5 1.08e-06
... ...
375 0.0469
376 0.0469
377 0.0469
378 0.0474
379 0.0493
```

12.5 ORA con clusterProfiler

Vamos a analizar los datos `tamidata::gse21942`. Preparamos los datos con el siguiente código (más detalles en 13.10).

```
data(gse21942,package="tamidata")
tt = genefilter::rowttests(gse21942,
                           pData(gse21942)$FactorValue..DISEASE.STATE.)
```

Vamos a considerar significativos aquellos genes cuyo p-valor ajustado por el método de Bonferroni sea menor o igual a 0.01.

```
sel = which(p.adjust(tt$p.value,method="bonferroni")<.01)
gene =fData(gse21942)$ENTREZID[sel]
gene = na.omit(gene)
gene = unique(gene)
```

El universo de genes que vamos a considerar será los contemplados en la plataforma.

```
universe = fData(gse21942)$ENTREZID
universe = na.omit(universe)
universe = unique(universe)
```

Hacemos un análisis de sobre representación con Gene Ontology.

```
pacman::p_load(clusterProfiler,org.Hs.eg.db)
ego = clusterProfiler::enrichGO(gene = gene,
                universe = universe,
                OrgDb = org.Hs.eg.db,
                ont = "CC",
                pAdjustMethod = "BH",
                pvalueCutoff = 0.01,
                qvalueCutoff = 0.05,
                readable = TRUE)
```

Para la base de datos KEGG.

```
kk = clusterProfiler::enrichKEGG(gene = gene,
                organism = 'hsa',
                pvalueCutoff = 0.05)
```

Para la base de datos KEGG Module.

```
mkk = enrichMKEGG(gene = gene,
                organism = 'hsa',
                pvalueCutoff = 1,
                qvalueCutoff = 1)
```

12.6 tamidata::gse21942 con tami::ora

Utilizamos los datos `tamidata::gse21942` y vamos a realizar un análisis de sobre representación utilizando la función `tami::ora()`.

Empezamos determinando un grupo de genes significativos. Tenemos dos grupos definidos por la variable fenotípica `FactorValue..DISEASE.STATE.` que nos indica si el individuo tiene esclerosis múltiple o no. Utilizamos un test de la t moderado, esto es, utilizamos el método limma. Lo hacemos con la función `tami::dema()` indicando la función `rowtmod`.

```
pacman::p_load(tami)
data(gse21942,package="tamidata")
gse21942_deo = dema(x=gse21942,
                y="FactorValue..DISEASE.STATE.",
                test = rowtmod,correction = "BH",
                fdr = .0001,foutput = "gse21942")
```

A partir del objeto `gse21942_deo` podemos producir un informe en .

```
browseURL(glimpse(gse21942_deo))
```

Podemos obtener un **`data.frame`** con toda la información relevante.

```
gse21942.tidy = tidy(gse21942_deo)
```

Obtenemos los genes significativos una tasa de falso rechazo de 0,05.

```
sel0 = which(gse21942.tidy[,"adjp"] < .05)
set0 = gse21942.tidy[sel0,"ENTREZID"]
```

Utilizamos la colección de grupos de genes de humanos en Gene Ontology correspondiente a procesos biológicos.

```
hsaGO = EnrichmentBrowser::getGenesets("hsa",onto="BP")
```

Finalmente realizamos el análisis de sobre representación.

```
gse21942_gsao = ora(set0,gsc=hsaGO)
```

12.7 Ejercicios

Ex. 26 — Utilizamos los datos `tamidata::gse20986`. En el problema 15 hemos determinado los genes significativos con un FDR de 0.05 para las comparaciones entre las muestras obtenidas en el iris, retina y coroides con las muestras huvec. Tenemos pues tres grupos de genes significativos que podemos denotar S_{iris}, S_{retina} y $S_{coroides}$.

1. Realizar, para cada uno de los tres grupos de genes seleccionados, un análisis de un sobre solapamiento, utilizando el test de Fisher unilateral, con los grupos definidos en Gene Ontology. En concreto hay que utilizar las tres bases de bases de Gene Ontology: BP (procesos biológicos), CC (componentes celulares) y MP (función molecular).

Ex. 27 —

1. Construir colecciones de grupos de genes utilizando [101] para el mus musculus. Para ello vamos a utilizar el paquete de anotación del organismo `org.Mm.eg.db`. Posiblemente ha de instalarse previamente.
2. Realiza un análisis de sobre representación con el paquete [51] utilizando la variable fenotípica `type`.

Ex. 28 —

1. Realizar el ejercicio 27 utilizando el paquete [163, clusterProfiler].

Capítulo 13

Análisis de conjuntos de genes

13.1 Introduction

[1]

El problema abordado en este capítulo corresponde a lo que se conoce como análisis de conjunto de genes.[2] Estudiamos si hay relación entre conjuntos de genes **previamente definidos** y fenotipo. Donde la expresión fenotipo tiene un sentido amplio como en todo el texto. Estos grupos vendrán definidos según distintos criterios. Por ejemplo, corresponder a una ruta metabólica, o localizados en un mismo cromosoma o bien definidos utilizando términos de Gene Ontology. Un conjunto de genes que represente algo interpretable desde un punto de vista biológico.

En este punto no estamos interesados en un análisis *marginal* o *gen a gen* de los datos de expresión. En un análisis de expresión diferencial el resultado final es una lista ordenada de genes de modo que una mayor asociación con la covariable fenotípica produce una posición más alta en la lista. Un procedimiento de comparaciones múltiples nos produce una clasificación en genes significativos y no significativos. En definitiva un valor de corte de la lista ordenada de los genes. En esta aproximación *no usamos ningún conocimiento previo sobre relaciones conocidas entre genes* que podrían ser esenciales a la hora de determinar la asociación no ya gen-fenotipo sino conjunto de genes-fenotipo.

[1] Lo tratado en este capítulo tiene mucho que ver con la obra teatral de Lope de Vega, Fuenteovejuna. Se recomienda el video http://www.youtube.com/watch?v=-IcuFn57nAo.
- ¿Quién mató al Comendador?
- Fuenteovejuna, señor.
-¿Quién es Fuenteovejuna?
- Todo el pueblo, a una. Esta frase aparece repetida en la obra y es básica en este tema. Este tema va de esto, de la acción conjunta de un conjunto de genes. Unos pocos habitantes no pueden pero todos los habitantes de Fuenteovejuna sí que pueden.

[2] En la literatura se refieren a este problema como *gene set analysis*, *gene set enrichment analysis*, *set based enrichment analysis*.

13.2 Sobre la distribución de la matriz de expresión

Denotaremos la matriz de expresión (aleatoria) como

$$\boldsymbol{Y} = [Y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$$

donde Y_{ij} es la expresión aleatoria del i-ésimo gen en la j-ésima muestra. Las expresiones *observadas* serán

$$\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,n}$$

Y_{ij} es una variable aleatoria mientras que y_{ij} es un valor observado. Las columnas de la matriz de expresión Y son vectores aleatorios (de dimensión alta) independientes pero no sus filas.[3] Las distintas muestras son realizaciones de vectores independientes aunque no con la misma distribución ya que son observados bajo distintas condiciones experimentales. Si nos fijamos en las filas de la matriz de expresión, esto es, en los perfiles de expresión entonces tenemos realizaciones de vectores aleatorios dependientes y que no tienen la misma distribución.

13.3 Conjuntos de genes

Tenemos distintos conjuntos de genes previamente definidos utilizando información previa: un grupo(s) definido por el grupo de investigación, grupos definidos utilizando Gene Ontology, ... Lo fundamental, estos grupos de genes no han de estar definidos en función de *nuestros* datos de expresión. Si denotamos por $G = \{1, \ldots, N\}$ el conjunto total de genes considerado (o universo de genes) entonces los conjuntos de genes serán $S_1, \ldots, S_K$. En particular, no es extraño considerar el caso $K = 1$, es decir, estar interesados en un conjunto de genes dado. Supondremos que el conjunto S_k tiene cardinal $|S_k|$. Los distintos conjuntos S_k **no** son una partición de G: no son necesariamente disjustos ($S_i \cap S_j \neq \emptyset$ para $i \neq j$) y su unión no tiene porqué ser todo el universo considerado de genes.

La cuestión básica podemos formularla como: *¿Hay asociación entre un conjunto de genes y el fenotipo?*. Es la misma pregunta que nos formulábamos para cada gen pero ahora referido a un conjunto dado de genes. Es una pregunta muy vaga. Se requiere una formulación más precisa. Caben distintas interpretaciones de la pregunta. En [137] formulan las siguientes dos hipótesis nulas que concretan de dos modos distintos la cuestión previa. Reproducimos las hipótesis nulas.[4]

Hipótesis Q1: Los genes de un conjunto muestran el mismo patrón de asociación con el genotipo comparado con el resto de genes.

Hipótesis Q2: El conjunto de genes no tiene ningún gen cuyo nivel de expresión está asociado con el fenotipo de interés.

[3] Es claro que el preprocesado de la información introduce dependencias entre valores observados para distintas muestras pero las ignoramos. Con todo no se puede.

[4]

Hypothesis Q1: The genes in a gene set show the same pattern of associations with the phenotype compared with the rest of the genes.

Hypothesis Q2: The gene set does not contain any genes whose expression levels are associated with the phenotype of interest.

No tenemos la misma hipótesis nula. La hipótesis nula **Q1** se centra en la comparación entre (la asociación entre) un conjunto dado de genes (con el fenotipo) y (la asociación entre) los otros (con el fenotipo). En cambio, la hipótesis **Q2** se centra en el estudio de la expresión diferencial de los genes que pertenecen a un conjunto dado de genes.

Un planteamiento similar lo podemos encontrar en [62]. En concreto un test que se plantea si un conjunto de genes tiene una asociación con el fenotipo, la hipótesis **Q2**, recibe el nombre de *test autocontenido* (*self-contained test*) mientras que si nos ocupamos de la hipótesis nula **Q1** hablamos de un *test competitivo* (*competitive test*). De hecho el formula las hipótesis nulas del siguiente modo:

Hipótesis nula competitiva H_0^{comp}**:** Los genes en un grupo dado S están como mucho tan frecuentemente expresados de un modo diferencial como los genes en $S^c = G \setminus S$.

Hipótesis nula autocontenida H_0^{auto}**:** Ningún gen en S está diferencialmente expresado.

[5]

Muchos procedimientos estadísticos propuestos para contrastar estas hipótesis no formulan expresamente cuales son las hipótesis nulas que están contrastando. De hecho, la reflexión sobre el propio procedimiento de contraste propuesto es el que nos ha de indicar la hipótesis nula. No tenemos un modelo (estocástico) preciso y esto ocasiona esta indeterminación.

En lo que sigue veremos procedimientos que asumen una distribución conocida (o al menos asintóticamente conocida) para los estadísticos de contraste. Sin embargo, la opción más utilizada es evaluar la significación del estadístico asociado al contraste utilizando como distribución nula (o distribución bajo la hipótesis nula) una distribución de aleatorización o una distribución bootstrap. Qué distribución es adecuada dependerá de la hipótesis que estemos contrastando. De hecho, desde el punto de vista estadístico, este es el núcleo del problema. ¿Cómo estimamos la distribución nula?

13.4 Ejemplos

Es conveniente tener ejemplos artificiales para ilustrar y probar los procedimientos. En esta sección los comentamos.

13.4.1 Un ejemplo de Efron y Tibshirani

En [46] se proponen un par de ejemplos con datos simulados. Son los ejemplos 13.1 y 13.2.

Ejemplo 13.1. *Se consideran 1000 genes y 50 muestras. Se supone que las 25 primeras muestras son controles y las últimas 25 es el grupo de tratamiento. Definimos los grupos de genes como: las 20 primeras filas (de la matriz de expresión) corresponden al primer grupo, de la 21 a la 40 el segundo*

[5] La formulación original de las hipótesis es la siguiente:

Competitive null hypothesis H_0^{comp}**:** The genes in S are at most as often diferentially expressed as the genes in S^c.

Self-contained null hypothesis H_0^{self}**:** No genes in S are differentially expressed.

y así sucesivamente. Los niveles de expresión los generamos aleatoriamente independientemente y con la misma distribución. La distribución común es una normal estándar ($Y_{ij} \sim N(0,1)$). En el primer grupo añadimos un valor constante de 2.5 a los primeros 10 genes en las muestras correspondientes a tratamiento (últimas 25 columnas de la matriz de expresión). En consecuencia, lo que hacemos es que solamente el primer grupo tiene la mitad de los genes asociados con el fenotipo y la otra mitad sin ningún tipo de asociación. El resto de grupos no tiene ninguna asociación con fenotipo. Generamos estos datos.

```
N = 1000; n = 50
set.seed(280562) ## Para obtener los mismos valores generados
et1 = matrix(rnorm(N*n),nrow = N,ncol = n)
et1[1:10,26:50] = et1[1:10,26:50] + 2.5
```

En este ejemplo el vector que nos indica la clasificación de las muestras será

```
y.et = factor(rep(1:2,rep(25,2)),levels = 1:2,
              labels = c("Normal","Tratamiento"))
```

En lo que sigue utilizaremos como medida de asociación fenotipo-expresión el estadístico del t-test.

```
et1.tt = genefilter::rowttests(et1,y.et)$statistic
```

En figura 13.1 tenemos una estimación de la densidad. Podemos ver cómo se aprecia, por debajo de 8, el valor que corresponde al primer grupo.

```
pacman::p_load("ggplot2")
df = data.frame(et1.tt)
p = ggplot(df,aes(x=et1.tt))+geom_density()
ggsave(paste0(dirTamiFigures,"GeneSetAnalysis5b.png"),p)
```

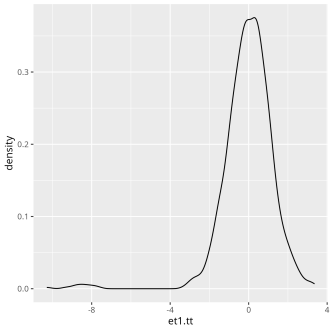


Figura 13.1: Densidad estimada de et1.tt.

Ejemplo 13.2. *Este ejemplo se define exactamente como el ejemplo 13.1 excepto lo que se hacía solamente para el primer grupo lo hacemos para todos: sumamos a los 10 primeros genes de cada grupo 2.5 unidades en las muestras correspondientes al tratamiento (últimas 25 columnas de la matriz de expresión). Generamos los datos.*

```
N = 1000; n = 50
set.seed(280562)
indices.temp = (0:49)*20 + 1
indices = NULL
for(i in indices.temp) indices = c(indices,i:(i+9))
et2 = matrix(rnorm(N*n),nrow = N,ncol = n)
et2[indices,26:50] = et1[1:10,26:50] + 2.5
```

Notemos que la covariable indicando el grupo es la misma que en ejemplo 13.1. Definimos los grupos para su uso posterior.

```
gen.name = function(i) paste0("g",((i-1)*20 +1):(i*20))
gsc.et = lapply(as.list(1:50),gen.name)
names(gsc.et) = paste0("set",as.character(1:50))
gsnames.et = paste0("set",as.character(1:50))
genenames.et = paste0("g",1:1000)
```

```
et2.tt = genefilter::rowttests(et2,y.et)$statistic
```

En figura 13.2 tenemos una estimación de la densidad.

```
df = data.frame(et2.tt)
p = ggplot(df,aes(x=et2.tt))+geom_density()
```

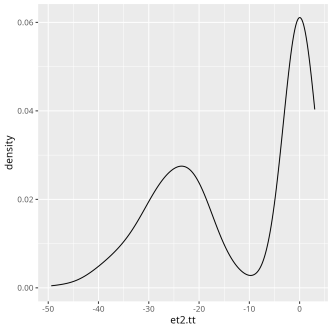


Figura 13.2: Densidad estimada de

13.4.2 gse21942

Realizaremos un análisis de grupo de genes con los datos `tamidata` ↪ `::gse21942`. Utilizaremos la colección de Gene Ontology utilizando el método visto en § 11.5.2.

```
pacman::p_load(GSEABase)
load(paste0(dirTamiData,"gscHs.rda"))
gscHs = geneIds(gscHs)
```

13.5 Cuantificando asociación gen-fenotipo

El primer paso es cuantificar la asociación entre los niveles de expresión de cada gen y el fenotipo. Esta cuantificación será un estadístico (una función de los datos que son las expresiones en la fila de la matriz de expresión) cuya definición dependerá del tipo de información fenotípica disponible. Por ejemplo, si tenemos dos condiciones entonces este estadístico será (habitualmente pero no siempre) el estadístico t que se utiliza en la comparación de medias de dos poblaciones normales. Pero no necesariamente, por ejemplo, si los tamaños muestras n_1 y n_2 son muy pequeños entonces muy probablemente un estadístico como la diferencia de medias es suficiente (y menos peligroso).

En § 9.1 se estudiaba el método SAM (en el contexto de la expresión diferencial marginal). Este método propone, dependiendo de las descripción fenotípica con la que se trabaja, un estadístico d_i para cada gen. La definición de este estadístico es función del fenotipo. Son perfectamente utilizables como medidas de asociación gen-fenotipo (es lo que son).

En lo que sigue a la medida de asociación gen-fenotipo la denotaremos genéricamente por t_i (aunque no sea un t estadístico).

13.6 Enriqueciendo el conjunto de genes

El vector $\boldsymbol{t} = (t_1, \ldots, t_N)$ constituye, de hecho, una descripción de la asociación *marginal* gen-fenotipo. Pero tenemos unos grupos de genes en los que tenemos interés o que expresan simplemente conocimiento previo de posibles asociaciones entre los genes. Sea S uno de estos conjuntos. Enriquecer el estadístico t para el conjunto consiste en considerar un resumen de t sobre S. Y esto lo podemos hacer de muchas formas. Si denotamos el estadístico de enriquecimiento por $t(S)$ la opción más simple es

$$\boldsymbol{t}(S) = \sum_{i \in S} \frac{t_i}{n_S}, \tag{13.1}$$

es decir, el promedio de los t_i observados sobre el conjunto de genes de interés S. Podemos considerar muchas otras opciones y en las secciones que siguen las veremos.

13.7 Distribuciones condicionadas a los datos

Sea $\boldsymbol{x}$ la matrix de expresión observada. Y supongamos que el vector $\boldsymbol{y}$ (de dimensión n) nos indica la covariable de interés de la muestra

(normalmente la pertenencia a un grupo). Para el universo de genes considerado tendremos el vector $\boldsymbol{t} = (t_1, \ldots, t_N)$.

13.7.1 Distribución de permutación para un gen

Para un gen tenemos su perfil de expresión $u = (u_1, \ldots, u_n)$ y tenemos el vector y asociado a las muestras.

Sea π denota una permutación aleatoria de $(1, \ldots, n)$ de forma que los $\pi(i)$ será el índice que ocupa la posición i-ésima en la permutación π. En particular denotaremos por π_0 la permutación que nos devuelve el orden original: $\pi_0(j) = j$. Si el vector asociado a las muestras es y entonces tendremos el vector (permutado de y) $\boldsymbol{y}_\pi = (y_{\pi(1)}, \ldots, y_{\pi(n)})$. Obviamente $\boldsymbol{y} = y_{\pi_0}$.

La cantidad que describe la asociación entre u y $\boldsymbol{y}_\pi$ es t_π. Utilizando esta notación el valor observado de la asociación gen-fenotipo para u e y será t_{π_0}. Si consideramos B permutaciones aleatorias entonces tendremos los valores $t_{\pi_1}, \ldots, t_{\pi_B}$ para las B permutaciones aleatorias.

Si no hay asociación gen-fenotipo entonces el valor de t_{π_0} debe de ser *como* los valores $t_{\pi_1}, \ldots, t_{\pi_B}$. De hecho, cualquier ordenación de $t_{\pi_0}, t_{\pi_1}, \ldots, t_{\pi_B}$ tiene la misma probabilidad. Podemos considerar dos casos. En el primero una mayor asociación se expresa como un valor mayor de t (normalmente positivo). ¿Cuántos valores t_{π_b} (con $b = 1, \ldots, B$) son mayores que t_{π_0}? Si hay pocos indica que no hay equiprobabilidad y, por lo tanto, rechazamos esa hipótesis. Esto es un test de aleatorización aplicado a las muestras. El p-valor sería la proporción de t_{π_b}s mayores que t_{π_0}, es decir,

$$p_r = \frac{|\{b : t_{\pi_b} > t_{\pi_0}\}|}{B+1}. \tag{13.2}$$

En el caso en que una mayor asociación se exprese como un valor muy grande (positivo) o muy pequeño (negativo) de t entonces el p valor vendría dado por

$$p_r = \frac{|\{b : |t_{\pi_b}| > |t_{\pi_0}|\}|}{B+1}, \tag{13.3}$$

ya que tendríamos un test bilateral.

Ejemplo 13.3. *Consideremos el ejemplo 13.1, en concreto, los valores del primer gen. La covariable y nos indica la pertenencia a control o tratamiento. Como medida de asociación consideramos el estadístico t (con varianzas desiguales). Notemos que una asociación grande supone un valor o muy grande (positivo) o muy pequeño (negativo).*

```
u = et1[1,]
t0 = t.test(u ~y.et)$statistic
t0 = abs(t0)
```

Generamos $B = 100$ permutaciones aleatorias de y utilizando la función `sample`.

```
B = 100
tb = rep(0,B)
for(i in 1:B) tb[i] = t.test(u ~sample(y.et))$statistic
```

Determinamos los valores absolutos de los estadísticos.

```
tb = abs(tb)
```

El valor observado de t_{π_0} es 8.77. Y el p-valor será la proporción de los que su valor absoluto es mayor que el valor absoluto de t_{π_0}), es decir,

```
sum(tb > t0) / (B+1)
```

```
[1] 0
```

Tenemos un p-valor nulo. No parece que todas las ordenaciones de los valores $t_{\pi_0}, t_{\pi_1}, \ldots, t_{\pi_B}$ sean equiprobables.

La distribución nula que hemos considerado en esta sección corresponde con la hipótesis nula autocontenida que veíamos previamente. No asumimos ninguna distribución asintótica para el estadístico enriquecido y generamos una distribución nula en la que intercambiamos las etiquetas de las muestras. Si nuestro interés es contrastar la hipótesis Q2 o hipótesis autocontenida esta sería la distribución nula natural.[124]

[124] Podemos hablar de aleatorización por columnas entendiendo que las distintas muestras corresponden a las distintas columnas en la matriz de expresión.

13.7.2 Distribución de aleatorización

Denotemos por S_0 el conjunto de genes original en el que tenemos interés. Ahora vamos a elegir al azar un grupo S de genes (filas) del mismo cardinal que S_0. Lo aleatorio ahora no es el vector y sino el conjunto S de genes. Si S es aleatorio entonces también lo es $\boldsymbol{t}(S)$ (ahora las muestras tienen su ordenación original). A la distribución de probabilidad de $\boldsymbol{t}(S)$ se le llama **distribución de aleatorización** del estadístico de enriquecimiento. Si tomamos B selecciones aleatorias de n_{S_0} genes tendremos los conjuntos S_b con $b = 1, \ldots, B$ y los valores del estadístico de enriquecimiento observados son $\boldsymbol{t}(S_0)$ (para el grupo original) y $\boldsymbol{t}(S_b)$ con $b = 1, \ldots, B$ para los seleccionados al azar. El p-valor se define análogamente como

$$p = \frac{|\{b : t(S_b) > t(S_0)\}|}{B+1}. \tag{13.4}$$

si solamente rechazamos para valores grandes (positivos). En el caso bilateral tendremos

$$p = \frac{|\{b : |t(S_b)| > |t(S_0)|\}|}{B+1}. \tag{13.5}$$

La distribución nula que hemos considerado en esta sección corresponde con la hipótesis nula competitiva o hipótesis Q1. No asumimos ninguna distribución asintótica para el estadístico enriquecido y generamos una distribución nula en la que seleccionamos al azar grupos del mismo tamaño. Realmente lo que estamos haciendo es generar permutaciones aleatorias de las filas en la matriz de expresión. Si nuestro interés es contrastar la hipótesis Q1 o hipótesis competitiva esta sería la distribución nula natural.[6]

En lo que sigue repasamos distintos paquetes R/Bioconductor que implementan distintas medidas de enriquecimiento y distribuciones nulas en donde permutamos aleatoriamente las columnas (muestras) o las filas (características).

[6] Podemos hablar de aleatorización por filas entendiendo que los distintos genes (o genéricamente características que observamos) corresponden a las distintas filas en la matriz de expresión. También se habla de muestreo de genes o *gene sampling*.

13.8 Usando Limma

13.8.1 genSetTest

En el paquete [128, limma] tenemos la función `limma::geneSetTest` ↪ `()`. En este caso la medida de enriquecimiento que se utiliza es la media muestral de los estadísticos calculados para cada gen. Simplemente se permuta por filas. Contrastamos pues la hipótesis competitiva.

Ejemplo 13.4 (genSetTest y ejemplo 13.1)**.** *Consideremos los datos del ejemplo 13.1 y el t-estadístico para cada gen. El grupo de interés es el primero (primeras 20 filas). Supongamos que tomamos como alternativa si tienden a tomar valores mayores en el primer grupo.*

```
pacman::p_load(limma)
geneSetTest(1:20,et1.tt,alternative ="up")
```

```
[1] 0.9984722
```

El p-valor nos indica que no rechazamos la hipótesis nula. ¿Y si contrastamos la alternativa de valores mayores en el segundo grupo?

```
geneSetTest(1:20,et1.tt,alternative ="down")
```

```
[1] 0.00153168
```

Vamos a realizar el contraste para cada uno de los 50 grupos considerados.

```
pvalores = NULL
for(i in 0:49){
  indices = (i * 20 + 1): (i * 20 + 10)
  p.temp = geneSetTest(indices,et1.tt,alternative ="down")
  pvalores = c(pvalores,p.temp)
}
```

Comprobamos que el p-valor del primer grupo es claramente menor que el p-valor para los restantes grupos.

13.8.2 wilcoxGST

Siguiendo con el paquete [128, limma] una opción simple (no necesariamente muy potente) y que no utiliza la distribución de aleatorización consiste en tomar los valores del estadístico en el grupo de interés y en el resto de genes y compararlos utilizando un test de Wilcoxon. Por lo tanto estamos en el caso en que **conocemos** la distribución nula. No aleatorizamos ni por filas ni por columnas. Es un test exacto. El test de Wilcoxon tampoco asume ninguna hipótesis distribucional sobre los estadísticos por gen que utilizamos lo que es una ventaja. Sin embargo, asume independencia entre estos estadísticos que **no** se verifica ya que las expresiones de los genes son interdependientes.

Ejemplo 13.5 (Datos de ejemplo 13.1)**.** *Utilizamos la función limma::wilcoxGST.*

```
wilcoxGST(1:20,et1.tt,alternative ="down")
```

```
[1] 0.00153168
```

```
wilcoxGST(1:20,et1.tt,alternative ="up")
```

```
[1] 0.9984722
```

```
wilcoxGST(1:20,et1.tt,alternative ="mixed")
```

```
[1] 2.837697e-08
```

Tomemos otro grupo y repitamos el análisis.

```
wilcoxGST(21:30,et1.tt,alternative ="down")
```

```
[1] 0.09546351
```

```
wilcoxGST(21:30,et1.tt,alternative ="up")
```

```
[1] 0.904723
```

```
wilcoxGST(21:30,et1.tt,alternative ="mixed")
```

```
[1] 0.8181643
```

13.8.3 CAMERA

El método fue propuesto en [155].[125] Es un procedimiento para contrastar la hipótesis competitiva. Es un procedimiento que tiene en cuenta la correlación entre las expresiones de los distintos genes en cada muestra. La mayor parte de los procemientos propuestos para el contraste de la hipótesis competitiva suelen asumir (falsamente) la independencia de la expresión observada para distintos genes y su validez depende una hipótesis que sabemos no es cierta. Se ha visto que no tener en cuenta esta dependencia entre genes nos lleva a un incremento de la tasa de error tasa de falso rechazo (FDR).

[125] El nombre es un acrónimo **C**orrelation **A**djusted **ME**an **RA**nk.

Se asume que la expresión está cuantificada en escala logarítmica (base 2 como es habitual en este contexto).

$$E[Y_{ij}] = \mu_{ij} = \sum_{k=1}^{p} \alpha_{ik} y_{jk} \tag{13.6}$$

siendo y_{jk} la k-ésima covariable (o variable fenotípica) de la j-ésima muestra. Se supone que las expresiones aleatorias (su perfil aleatorio de expresión) de un mismo gen tienen una varianza común σ_i^2.[126] Se asume que las expresiones para distintas muestras son independientes pero no entre distintos genes. En particular, denotamos el coeficiente de correlación de Pearson entre las variables aleatorias $Y_{i_1 j}$ y $Y_{i_2 j}$ como $cor(Y_{i_1 j}, Y_{i_2 j}) = \rho_{i_1,i_2}$. Obviamente no asumimos que estos coeficientes sean nulos.[127]

[126] Asumible si hemos aplicado previamente algún procedimiento de normalización.

[127] No asumimos incorrelación. Recordemos que independencia implica incorrelación pero no viceversa.

Un contraste, para el gen i-ésimo viene dado por

$$\beta_i = \sum_{k=1}^{p} c_j \alpha_{ik}.$$

Se está interesado en contrastar la hipótesis nula de que el contraste es nulo. Tenemos interés en los contrastes de hipótesis: $H_0 : \beta_i = 0$ frente a la alternativa de que $H_0 : \beta_i \neq 0$. Denotemos el estadístico del contraste como Z_i.[128] ¿Qué estadísticos Z_i vamos a considerar?

[128] No necesariamente con distribución normal.

13.9 GSA

[129] En el paquete [45, GSA] se implementa el método propuesto en [46].

[129] Se utiliza la distribución de permutación a la que aplican una reestandarización. La medida de enriquecimiento que proponen por defecto [7] es la siguiente: partimos de los valores t_i que miden asociación gen-fenotipo. Definimos $t^+ = \max\{t, 0\}$ y $t^- = -\min\{t, 0\}$. Consideramos un conjunto de genes S. Definimos $\bar{t}_S^+ = \sum_{i \in S} \frac{t_i^+}{n_S}$ y $\bar{t}_S^- = \sum_{i \in S} \frac{t_i^-}{n_S}$. Finalmente la medida de enriquecimiento (que llamaremos el estadístico `maxmean`) es

$$\boldsymbol{t}(S) = \max\{\bar{t}_S^+, \bar{t}_S^-\}. \tag{13.7}$$

Estamos tomando la media de las partes positivas t_i^+, la media de las partes negativas t_i^- y nos quedamos con el máximo de ambos valores.

Ejemplo 13.6. *[46, página 119] La medida de enriquecimento que acabamos de definir es robusta frente al caso en que tengamos una medida extrema. El ejemplo que proponen los autores es el siguiente: supongamos que tenemos 100 genes, 99 de los valores t_i's son -0.5 y el valor restante es 10. Tenemos que $\bar{t}_S^+ = 10/100 = 0.1$ y $\bar{t}_S^- = -99(-0.5)/100 = 0.495$. Vemos que los valores negativos dominan pues son la mayor parte de los datos. Si se tomara la media de las partes positivas t^+ y la media de las partes negativas t^- tendríamos los valores 10 y -0.5 (es decir, solamente consideramos cuando el valor no es nulo) y la medida de enriquecimiento vendría dominada por valores extremos.*

Como medidas de asociación gen-prototipo t_i utilizan las mismas del paquete [**R-samr**] que aparecen en la sección § 9.1.1. Cuando analizan los conjuntos de genes hablan de *grupos de genes positivos* y *grupos de genes negativos*. Un grupo de genes se dice negativo si corresponden con genes que en la clase 2 tiene expresiones menores cuando tenemos dos grupos (1 y 2). Si tenemos una covariable y numérica entonces los negativos corresponden al caso en que expresiones menores se asocian a valores mayores de y. Los grupos positivos se definen de modo contrario a los positivos. Cargamos el paquete.

```
pacman::p_load(GSA)
```

Ejemplo 13.7 (Ejemplo 13.1 con GSA)**.** *Empezamos analizando el ejemplo 13.1.*

```
grupo = c(rep(1,25),rep(2,25)) #El grupo tiene que numerarse 1,2
et1.gsa = GSA(et1,grupo, genenames=genenames.et, genesets=gsc.et,
  resp.type="Two class unpaired", nperms=100)
```

Podemos ver los estadísticos enriquecidos para los grupos

```
head(et1.gsa$GSA.scores)
```

```
[1] 3.99534187 -0.01231818 0.11759218
[4] -0.27081919 0.05557714 -0.07898232
```

El valor observado para el grupo 1 es 4 mientras que una descriptiva de los demás es la siguiente

```
summary(et1.gsa$GSA.scores[-1])
```

[7] Aunque lleva el promedio de los t_i's y el promedio de $|t_i|$ como otras opciones.

```
   Min. 1st Qu. Median Mean 3rd Qu.
-0.44625 -0.12429 -0.04239 -0.05241 0.05558
   Max.
 0.21225
```

Los p-valores obtenidos para lo que ellos llaman genes negativos que corresponden con genes que en la clase 2 tiene expresiones menores. Podemos obtenerlos con

```
head(et1.gsa$pvalues.lo)
```

```
[1] 1.00 0.51 0.63 0.09 0.58 0.42
```

Nuestro grupo 1 no destaca. Si considerados los p-valores de grupos positivos (en grupo 2 expresiones mayores) tenemos

```
head(et1.gsa$pvalues.hi)
```

```
[1] 0.00 0.49 0.37 0.91 0.42 0.58
```

que para el grupo 1 vale 0 y un resumen de los demás es

```
summary(et1.gsa$pvalues.hi[−1])
```

```
  Min. 1st Qu. Median Mean 3rd Qu. Max.
 0.1400 0.3800 0.5800 0.5706 0.7100 0.9700
```

Los valores originales t_i los tenemos con

```
head(et1.gsa$gene.scores)
```

```
[1] 3.485459 3.901185 3.698664 4.143949 3.555439
[6] 3.656246
```

En la figura 13.3 podemos ver un diagrama de cajas que compara el primer grupo con los demás.

```
fac0 = factor(c(rep(1,20),rep(2,980)),levels = 1:2,labels=c("Grupo 1","Resto"))
boxplot(et1.gsa$gene.scores ~fac0)
```

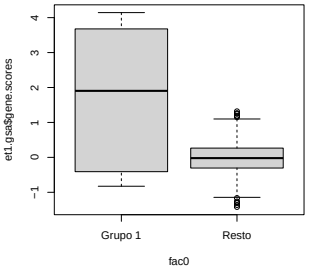


Figura 13.3: Valores t_i para el primer grupo (izquierda) y los demás. El primer grupo tiene diez genes claramente diferenciados mientras que los demás no lo están. De ahí la gran variabilidad que muestra el grupo.

En el primer grupo tenemos la mitad de los genes diferenciados y la otra mitad no. En el resto de los genes no hay diferenciación. De ahí la gran variabilidad del primer grupo y que se solape por la parte inferior con el resto de los genes. El análisis de grupo que hemos realizado lo diferencia sin problemas.

En la figura 13.4 tenemos los grupos de genes que se considerarían significativos (diferenciando grupos up y down) y el valor de FDR para que los declaremos significativos en los datos `et1` *(ejemplo 13.1).*

```
GSA.plot(et1.gsa)
```

Vemos cómo solamente admitimos un grupo con un valor bajo de FDR.

Ejemplo 13.8 (Ejemplo 13.2 con GSA)**.** *En el ejemplo 13.2 todos los grupos considerados tienen el mismo comportamiento. Aunque todos tienen expresión diferencial, no hay un comportamiento diferenciado del grupo respecto de los otros grupos. Los datos son* `et2` *mientras que los grupos los tenemos en* `gsc.et` *con nombres en* `gsnames.et`*. Finalmente* `genenames.tt` *nos da los nombres de los genes.*

```
grupo = c(rep(1,25),rep(2,25))
et2.gsa = GSA(et2,grupo, genenames=genenames.et, genesets=gsc.et,
    resp.type="Two class unpaired", nperms=100)
```

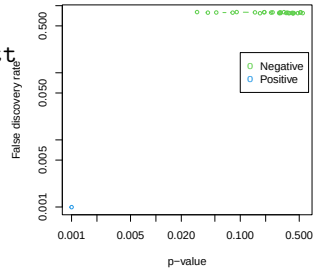


Figura 13.4: Grupos significativos con datos et1 en función de FDR.

La figura 13.5 muestra los grupos significativos en función de la FDR para los datos `et2` *(ejemplo 13.2).*

```
png(paste0(dirTamiFigures,"et2GSAplot.png"))
GSA.plot(et2.gsa)
dev.off()
```

```
pdf
  2
```

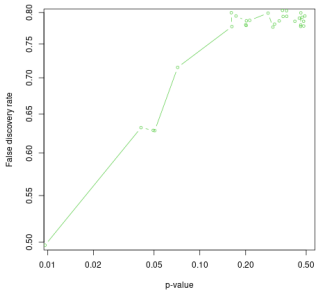


Figura 13.5: Grupos significativos con datos et2 en función de FDR.

Es claro que ningún grupo se diferencia de los demás.

Ejemplo 13.9 (Datos GSE1397 y GSA). *Leemos los datos.*

```
data(gse1397,package="tamidata")
```

En § 11.4 hemos construido la colección de grupos de genes.

```
gsc = gse1397.gsc
gruposGrandes = which(sapply(geneIds(gsc),length) > 50)
gsc = gsc[gruposGrandes]
gse = gse1397
```

```
Error: objeto 'gse1397' no encontrado
```

Realizamos el análisis. Previamente convertimos la variable `tipo` *que es un factor a un vector numérico con valores 1 y 2 (es como lo pide [45, GSA].*

```
tipo.num = as.numeric(pData(gse)[,"type"])
```

Utilizamos como nombre de los genes los AffyId.

```
gse1397.gsa = GSA(exprs(gse),tipo.num, genenames=featureNames(gse),
   genesets=geneIds(gsc),resp.type="Two class unpaired", nperms=1000)
```

En la figura 13.6 podemos ver la representación de la tasa de falsos positivos como función del p-valor.

```
png(paste0(dirTamiFigures,"gse1397GSAplot.png"))
GSA.plot(gse1397.gsa)
dev.off()
```

```
pdf
  2
```

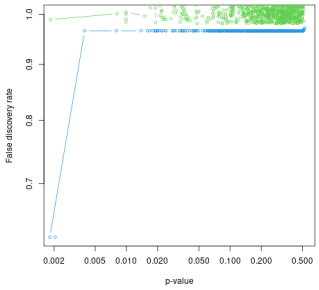


Figura 13.6: Grupos significativos con datos gse1397.

Fijamos una tasa de error de 0.05. Veamos qué grupos son los que o bien con una asociación negativa o bien con asociación positiva son los que presentan una mayor diferenciación entre los dos grupos considerados (con y sin síndrome de Down). Primero determinar los índices de los grupos en nuestra colección.

```
(ind.lo = which(gse1397.gsa$pvalues.lo <.05))
```

```
 [1]  9  45  53  61  105  111  118  135  137
[10]  159  270  273  325  331  345  348  349  360
[19]  376  397  410  420  434  510  512  530  536
[28]  545  564  582  593  675  679  705  733  839
[37]  853  959  1004  1047  1067  1076  1243  1302
```

```
(ind.hi = which(gse1397.gsa$pvalues.hi <.05))
```

```
 [1]  38  46  70  181  218  237  261  309  310
[10]  477  485  492  532  546  560  605  618  624
[19]  638  639  650  655  668  672  692  728  769
[28]  801  823  833  859  865  869  914  997  1001
[37]  1028  1194  1245
```

Podemos ver sus (primeros) identificadores en Gene Ontology. Para unos

```
head(names(gsc[ind.lo]))
```

```
  GO:0000002 GO:0000012 GO:0000017
"GO:0000245" "GO:0001933" "GO:0001974"
  GO:0000018 GO:0000019 GO:0000022
"GO:0002244" "GO:0006493" "GO:0006611"
```

y análogamente

```
head(names(gsc[ind.hi]))
```

```
  GO:0000002 GO:0000012 GO:0000017
"GO:0001819" "GO:0001934" "GO:0006006"
  GO:0000018 GO:0000019 GO:0000022
"GO:0007193" "GO:0007605" "GO:0008333"
```

13.10 GSEA: Gene set enrichment analysis

La referencia básica es [134] que presenta una versión modificada del procedimiento originalmente propuesto en [99]. Nos sirve para contrastar la hipótesis competitiva y utiliza una modificación del test de Kolmogorov-Smirnov para dos muestras.

Método

Empezamos ordenando el universo de genes de acuerdo al grado de asociación gen-fenotipo utilizando algún estadístico Tendremos la lista de genes ordenada, L.

Entradas

1. La matriz de expresión X con N filas y n columnas.
2. Procedimiento de ordenación con objeto de producir la lista ordenada de genes, L.
3. Un valor p
4. Un conjunto de genes S.

Cálculo del enriquecimiento

1. Calculamos medida de asociación gen-fenotipo, t_i.
2. Ordenamos el universo de genes de acuerdo a las medidas de asociación del paso 1. Denotamos los índices ordenados con $r_1 \ldots, r_N$, es decir,

$$t_{r_1} \geq \ldots \geq t_{r_N}.$$

3. Calculamos para cada i con $i = 1, \ldots, N$

$$h_S(i) = \frac{1}{\sum_{r_i \in S} |t_i|^p} \sum_{j \leq i; r_j \in S} |t_j|^p, \tag{13.8}$$

Calculamos también

$$m_S(i) = \sum_{j \leq i; r_j \notin S} \frac{1}{N - n_S}, \tag{13.9}$$

El valor de e_S es la máxima desviación de cero de $h_S(i) - m_S(i)$, es decir,

$$e_S = \max_{1 \leq i \leq N} |h_S(i) - m_S(i)|. \tag{13.10}$$

Si elegimos S de un modo aleatorio (del universo de genes) entonces e_S tendrá un valor pequeño en relación a lo que se observa cuando S no es aleatorio bien concentrándose en la parte superior de la lista o en la inferior o siguiendo algún patrón no aleatorio. Si $p = 0$ entonces e_S es el estadístico del test de Kolmogorov-Smirnov.

Estimación del p-valor

1. Consideramos una asignación aleatoria del fenotipo a las muestras (una permutación aleatoria de las muestras manteniendo fijo el fenotipo), reordenamos los genes y calculamos el valor $E(S)$.
2. Se repite el paso anterior un gran número de veces.
3. Si e_0 es el valor de e_S sobre los datos originales y $e_1, \ldots, e_B$ son los valores de e_S para las asignaciones aleatorias de fenotipo a muestras entonces el p-valor viene dado por

$$p = \begin{cases} 2\frac{|\{e_i : e_i \geq e_0\}|}{B+1}, \text{ for } e_0 > 0, \\ 2\frac{|\{e_i : e_i \leq e_0\}|}{B+1} \text{ for } e_0 < 0. \end{cases}$$

Contrastes múltiples Suponemos que consideramos ahora una colección de conjuntos de genes de interés: $S_1, \ldots S_K$.

1. Calculamos e_{S_k} para $k = 1, \ldots, K$.
2. Para cada S_k y 1000 permutaciones fijas π_i con $i = 1, \ldots, 1000$ (las mismas para todos los conjuntos de genes) del fenotipo, reordenamos los genes y calculamos el enriquecimiento e_{S_k,π_i}.
3. Ajustamos por el tamaño variable de los conjuntos de genes. Para ello consideramos los valores

$$\bar{e}_+ = \sum_{k,i:e_{S_k,\pi_i}>0} \frac{e_{S_k,\pi_i}}{|\{k,i : e_{S_k,\pi_i} > 0\}|},$$

y

$$\bar{e}_- = \sum_{k,i:e_{S_k,\pi_i}<0} \frac{e_{S_k,\pi_i}}{|\{k,i : e_{S_k,\pi_i} < 0\}|}.$$

Consideramos los valores

$$\tilde{e}_0 = \begin{cases} \frac{e_0}{\bar{e}_+}, \text{ si } e_0 > 0, \\ \frac{e_0}{\bar{e}_-}, \text{ si } e_0 < 0. \end{cases}$$

y

$$\tilde{e}_{S_k,\pi_i} = \begin{cases} \frac{e_{S_k,\pi_i}}{\bar{e}_+}, \text{ si } e_{S_k,\pi_i} > 0, \\ \frac{e_{S_k,\pi_i}}{\bar{e}_-}, \text{ si } e_{S_k,\pi_i} < 0. \end{cases}$$

4. Consideremos un valor normalizado (según método de punto anterior) $\tilde{e}$. Se *estima* la tasa de falsos positivos para ese valor como: si $\tilde{e} > 0$

$$q(\tilde{e}) = \frac{|\{k,i : e_{S_k,\pi_i} \geq \tilde{e}\}|/|\{k,i : e_{S_k,\pi_i} \geq 0\}|}{|\{k : e_{S_k} \geq \tilde{e}\}|/|\{k : e_{S_k} \geq 0\}|}$$

Si $\tilde{e} < 0$ entonces

$$q(\tilde{e}) = \frac{|\{k, i : e_{S_k,\pi_i} \leq \tilde{e}\}|/|\{k, i : e_{S_k,\pi_i} \leq 0\}|}{|\{k : e_{S_k} \leq \tilde{e}\}|/|\{k : e_{S_k} \leq 0\}|}$$

Ejemplo 13.10. *Vamos a analizar los datos* `tamidata::gse21942` *con [163, clusterProfiler]. Vamos a construir un vector que almacene los estadísticos por gen y tenga como nombres de sus elementos sus identificadores* `Entrez`.

```
pacman::p_load("SummarizedExperiment")
data(gse21942,package="tamidata")
tt = genefilter::rowttests(gse21942,
                           pData(gse21942)$FactorValue..DISEASE.STATE.)
tt3 = tt$statistic
names(tt3) = fData(gse21942)$ENTREZID
tt3 = na.omit(tt3)
```

Eliminamos duplicidades eligiendo una sonda por gen. Nos quedamos con la primera aparición.

```
tt3 = tt3[match(unique(names(tt3)),names(tt3))]
```

Hay que ordenar en orden decreciente según el estadístico que hemos calculado por gen.

```
tt3 = sort(tt3,decreasing=TRUE)
```

Guardamos los nombre de los genes.

```
gene = names(tt3)
```

La función `clusterProfiler::groupGO()` *nos va construir la colección de grupos de genes Gene Ontology. Lo podemos hacer siempre que tengamos un paquete* `OrgDb` *§ 24.3.*

```
pacman::p_load(clusterProfiler,org.Hs.eg.db)
ggo = clusterProfiler::groupGO(gene = gene,
                               OrgDb = org.Hs.eg.db,
                               ont = "CC",
                               level = 3,
                               readable = TRUE)
```

Podemos ver la información de los dos primeros grupos.

```
head(ggo,n=2)
```

```
                 ID Description Count GeneRatio
GO:0000133 GO:0000133 polarisome 0 0/21358
GO:0000417 GO:0000417 HIR complex 1 1/21358
          geneID
GO:0000133
GO:0000417 HIRA
```

El método GSEA visto en esta sección lo podemos aplicar con la función `clusterProfiler::gseGO()`.

```
ego3 = clusterProfiler::gseGO(geneList = tt3,
                              OrgDb = org.Hs.eg.db,
                              ont = "CC",
                              minGSSize = 100,
                              maxGSSize = 500,
                              pvalueCutoff = 0.05,
                              verbose = FALSE)
```

```
Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize
    ↪ = minSize, : For some of the pathways the P-values were likely
    ↪ overestimated. For such pathways log2err is set to NA.
```

```
Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize
    ↪ = minSize, : For some pathways, in reality P-values are less
    ↪ than 1e-10. You can set the `eps` argument to zero for better
    ↪ estimation.
```

También podemos hacer un análisis GSEA utilizando las rutas KEGG.

```
k3 = clusterProfiler::gseKEGG(geneList = tt3,
                              organism = 'hsa',
                              minGSSize = 120,
                              pvalueCutoff = 0.05,
                              verbose = FALSE)
```

```
Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize
    ↪ = minSize, : For some of the pathways the P-values were likely
    ↪ overestimated. For such pathways log2err is set to NA.
```

```
Warning in fgseaMultilevel(pathways = pathways, stats = stats, minSize
    ↪ = minSize, : For some pathways, in reality P-values are less
    ↪ than 1e-10. You can set the `eps` argument to zero for better
    ↪ estimation.
```

```
head(k3,n=1)
```

```
             ID Description
hsa00190 hsa00190 Oxidative phosphorylation
        setSize enrichmentScore NES pvalue
hsa00190 127 0.6755136 2.770191 1e-10
        p.adjust qvalue rank
hsa00190 8.5e-10 3.508772e-10 2015
                       leading_edge
hsa00190 tags=44%, list=9%, signal=40%




hsa00190 518/1351/4708/4720/29796/374291/4728/126328/4722/27089/4697/
    ↪ 4718/1345/1349/522/4694/523/9114/1329/526/7384/1337/4723/4714/
    ↪ 51079/7388/4726/4513/537/533/8992/4710/528/4716/4713/51606/4717
    ↪ /4696/4712/4707/9296/7386/10312/527/4706/55967/51382/4702/506/
    ↪ 4537/514/4725/155066/4512/1347/515
```

También podemos repetir el análisis para los módulos KEGG (otra colección dentro de KEGG).

```
mkk2 = gseMKEGG(geneList = tt3,
                organism = 'hsa',
                pvalueCutoff = 1)
head(mkk2,n=1)
```

```
          ID Description setSize
M00160 M00160 V-type ATPase, eukaryotes 24
       enrichmentScore NES pvalue
M00160 0.7396864 2.227717 5.077186e-06
          p.adjust qvalue rank
M00160 0.0001142367 8.283829e-05 1883
                       leading_edge
M00160 tags=54%, list=9%, signal=49%
                                                    core_enrichment
M00160 523/9114/526/537/533/8992/528/51606/9296/10312/527/51382/155066
```

Podemos visualizar los conjuntos de genes que hemos visto que tienen un comportamiento diferenciado desde el punto de vista competitivo.[8]

```
clusterProfiler::browseKEGG(k3,'hsa00190')
```

13.11 Estudiando interacciones entre genes individuales y grupos de genes

Suponemos que tenemos bien definido un grupo de genes S y pretendemos estudiar la posible asociación entre un gen g no incluido en S y el grupo S. Se trata de estudiar posibles asociaciones no conocidas previamente entre la actividad del gen individual y la actividad conjunta del grupo S.

En [24] consideran el caso en que las muestras están clasificadas en dos grupos. Se propone el siguiente método. Consideramos la matriz de datos $[y_{ij}]_{i\in S, j\in\{1,\ldots,n\}}$. Se aplican una componentes principales a esta matriz de datos y nos quedamos con la primera componente principal. Por cada muestra tendremos la primera componente del grupo S y la expresión del gen de interés. Se calcula el coeficiente de correlación de los pares de valores indicados en un grupo y en otro (trabajamos en el caso en que las muestras las clasificamos en dos grupos y se el módulo de la diferencia de los coeficientes de correlación. Obviamente un valor muy grande indica que el coeficiente de correlación en cada uno de los grupos considerados es muy diferente y por tanto el tipo de asociación entre el gen y el grupo de genes es distinto.[130]

[130] Indican que ha implementado el método en el paquete R GPCscore pero no se encuentra en ningún repositorio.

13.12 Un método simple pero efectivo

Es un método propuesto en [73]. Insiste mucho en este trabajo en la innecesaria complejidad del método GSEA comentado en sección § 13.10. Su interés está en la detección de grupos de genes que se asocian con el fenotipo (consideran la situación en que comparamos dos grupos) donde algunos de los genes están sobre expresados (up-regulated) y otros infra expresados (down-regulated).

Como siempre t_i es la medida de asociación marginal gen-fenotipo (puede ser el t-estadístico pero no necesariamente). Si consideramos el grupo de genes S con n_S elementos. Asumiendo (lo que no es cierto) que los distintos t_i son valores observados de variables independientes y con varianza igual a 1, entonces aproximadamente (aplicando teorema central del límite) se tiene que

$$E^{(1)} = \sqrt{n_S}\bar{t}_S \sim N(0,1), \tag{13.11}$$

siendo $\bar{t}_S = \sum_{i\in S} t_i/n_S$. Aunque no hay mayores justificaciones de esta distribución aproximada los autores indican que, sobre los datos, es asumible.[9] Utilizando esta distribución podemos tener fácilmente p-valores que no están basados en la distribución de permutación sino en la distribución normal estándar.

[8] La interpretación de estas rutas se salen completamente de los limitados conocimientos biológicos del que escribe.

[9] Si la cosa funciona: ¿por qué no?

El estadístico considerado en 13.11 detecta cambios en la media pero no cambios de escala. Si un grupo de genes tiene la mitad que están sobre expresados y la otra mitad infra expresados entonces no detectaríamos cambios en la media pero la variabilidad de los valores t_i sí que se modificaría claramente. Proponen utilizar el siguiente estadístico con la siguiente distribución aproximada (con $n_S \geq 20$ proponen los autores)

$$E^{(2)} = \frac{\sum_{i \in S}(t_i - \bar{t}_S)^2 - (n_S - 1)}{\sqrt{2(n_S - 1)}} \sim N(0,1) \tag{13.12}$$

[10] Tendremos un p-valor asociado a cada grupo de genes. Una vez tenemos todos los grupos tenemos un problema de comparaciones múltiples que resolvemos utilizando lo visto en el capítulo § 8. En particular, los autores utilizan los q-valores de Storey. Esencialmente el trabajo se centra mucho en la comparación con el método GSEA destacando su simplicidad frente al otro y que los resultados obtenidos son esencialmente los mismos. Algunos comentarios propios (discutibles):

1. En mi opinión hay un uso de resultados asintóticos que no siempre son asumibles.

2. En cualquier caso, el tamaño de los grupos de genes ha de ser suficientemente grande para que se verifiquen.

3. Es una buena opción a probar.

13.13 Un método basado en poblaciones finitas

Veamos el método propuesto en [108]. Consideremos el conjunto de genes S con m genes.[11] Tenemos t_i la medida de asociación gen-fenotipo para el i-ésimo gen. Consideremos la media muestral como estadístico de enriquecimiento dado por

$$\bar{t}_S = \sum_{i \in S} \frac{t_i}{n_S}.$$

En el resto de aproximaciones lo que se considera aleatorio es la medida de asociación t_i y *fijo* el conjunto de genes S. En este trabajo se adopta otro punto de vista. Tenemos $G = \{1, \ldots, N\}$ un universo de genes y nuestro conjunto de genes S es un *subconjunto aleatorio* de G. Por tanto nos planteamos cuál es la distribución de probabilidad de $\bar{t}_S$ cuando S es un subconjunto aleatorio de tamaño n_S de G o, dicho de otro modo, extraemos al azar y sin reemplazamiento n_S genes del total de N que componen nuestro universo. Los autores llaman a esta distribucion de probabilidad el modelo de conjunto aleatorio (random-set model). Notemos que se plantea la distribución de probabilidad de $\bar{t}_S$ **condicionada**

[10] Bajo la hipótesis de que los t_i son independientes, con una distribución común con varianza unitaria entonces el estadístico tiene una distribución aproximadamente normal estándar. En el trabajo original dividen por $2(n_S - 1)$ y no la raiz cuadrada. Entiendo que debe ser una errata.

[11] Estamos denotando el cardinal de S por n_S. Aquí denotamos por m porque queremos destacar que es un número fijo y dado una vez tenemos el grupo de genes S.

a los valores t_i. Este es el punto básico, los t_i están dados y *no se consideran aleatorios sino fijos, dados previamente*. Utilizando resultados de muestreo en poblaciones finitas se tiene que la media de $\bar{t}_S$ es

$$\mu = E(\bar{t}_S) = \sum_{i \in G} \frac{t_i}{N}, \tag{13.13}$$

es decir, la media de todos los t_i observados sobre todo el universo de genes. La varianza es

$$\sigma^2 = var(\bar{t}_S) = \frac{1}{m}\frac{N-m}{N-1}\left[\sum_{i \in G}\frac{t_i^2}{N} - \left(\sum_{i \in G}\frac{t_i}{N}\right)^2\right]. \tag{13.14}$$

En este procedimiento no recurrimos a ninguna distribución de aleatorización del estadístico. Estamos utilizando una distribución asintótica en el sentido en que se asume un tamaño de S suficientemente grande para que (aplicando el teorema central del límite) podamos considerar que $\bar{t}_S$ tiene una distribución aproximadamente normal. Como estadístico de enriquecimiento utilizan la versión estandarizada de $\bar{t}_S$, es decir, utilizan

$$Z = \frac{\bar{t}_S - \mu}{\sigma} \tag{13.15}$$

con μ y σ dados en las ecuaciones 13.13 y 13.14. Bajo la hipótesis nula de que no hay ningún gen diferencialmente expresado en el conjunto de genes S entonces Z tiene aproximadamente una distribución normal estándar.

$$Z \sim N(0,1). \tag{13.16}$$

13.14 Análisis de grupos de genes por muestra

[131]

¿Qué es lo que se pretende en este tipo de procedimientos? Partiendo de una matriz de expresión y una colección de grupos de genes se obtiene una cuantificación **para cada muestra y grupo de genes** de lo diferente que es la expresión en esa muestra del grupo de genes considerado respecto del resto de grupos de genes. De alguna forma lo que estamos atendiendo es a la hipótesis competitiva pero para cada muestra. Pasamos a otra matriz en la que en filas tenemos los grupos de genes considerados mientras que el número de muestras se conserva. Trasladamos el problema de expresión diferencial a nivel de gen[132] a un problema de expresión diferencial a nivel de grupos de genes. Y el valor que asociamos a cada grupo cuantifica lo diferente que es, en la muestra considerada, la expresión de los genes del grupo de la expresión de los genes que no pertenecen a este grupo.

Con esta nueva matriz podemos aplicar análisis de expresión diferencial donde en lugar de genes trabajamos con un grupo de genes. De hecho, con una colección de grupos de genes. Esto, obviamente, no tiene ninguna importancia desde el punto de vista estadístico. Podemos aplicar todo lo visto para expresión diferencial marginal.[133]

[131] Single sample gene set analysis

[132] Expresión diferencial marginal en este texto.

[133] El adjetivo marginal ahora se refiere a grupos de genes.

13.14.1 GSVA

Este método fue propuesto en [68] y está implementado en el paquete [**R-GSVA**].

Suponemos $\boldsymbol{y} = [y_{ij}]_{i=1,\ldots,N;j=1,\ldots,N}$ la matriz de expresión obtenida después de normalización previa. Tenemos los grupos de genes $\{S_1 \ldots, S_K\}$ entendidos como subconjuntos de las filas de $\boldsymbol{x}$ ($S_k \subset \{1,\ldots,N\}$. El perfil de expresión del gen (en fila) i será $\boldsymbol{y}_i = (y_{i1},\ldots,y_{in})$. El método trabaja con datos de expresión de microarrays ($\log_2$) o bien con los conteos en datos de RNA-seq.

Estimación kernel de las funciones de distribución Utilizando el perfil de expresión $\boldsymbol{x}_i$ proponen realizar una estimación kernel de la función de distribución. Para datos continuos (microarrays) utilizan

$$\hat{F}_{h_i}(y_{ij}) = \frac{1}{n}\sum_{k=1}^{n}\int_{-\infty}^{\frac{y_{ij}-y_{ik}}{h_i}} \frac{1}{\sqrt{2\pi}} e^{-\frac{t^2}{2}}\, dt. \tag{13.17}$$

El ancho de banda h_i lo fijan en $h_i = \frac{s_i}{4}$ siendo s_i la desviación estándar de $\boldsymbol{x}_i$.

Para datos de conteo (RNA-seq) utilizan un kernel de Poisson. En concreto el estimador kernel de la función de distribución propuesto es

$$\hat{F}_r(y_{ij}) = \frac{1}{n}\sum_{k=1}^{n}\sum_{y=0}^{y_{ij}} e^{-(y_{ik}+r}\frac{y_{ik}+r}{y!}, \tag{13.18}$$

tomando $r = 0.5$ para hacer que la moda del kernel Poisson coincida con la propia observación. Denotamos por z_{ij} el valor $\hat{F}_{h_i}(y_{ij})$ o $\hat{F}_r(y_{ij})$ dependiendo de que trabajemos con microarrays o con conteos.

Órdenes simétricos Para cada muestra j ordenamos los datos z_{ij} con j constante. Los órdenes (no los valores ordenados) los denotamos por r_{ij}, esto es, $z_{r_1,j} \leq \ldots \leq z_{r_N,j}$. Finalmente los valores r_{ij} son a su vez transformados con el fin de conseguir que sean simétrico alrededor del origen. En concreto tomamos: $\tilde{r}_{ij} = |\frac{N}{2} - r_{ij}|$.

Agregación de puntuaciones por muestra Fijamos j (una muestra dada) y un grupo de genes S_k. y se define el estadístico

$$\nu_{jk}(s) = \frac{\sum_{i=1}^{s}|\tilde{r}_{ij}|^{\tau}1_{S_k}(r_{ij})}{\sum_{i=1}^{N}|\tilde{r}_{ij}|^{\tau}1_{S_k}(r_{ij})} - \frac{\sum_{i=1}^{s}1_{S_k^c}(r_{ij})}{N-|S_k|}, \tag{13.19}$$

donde $1_{S_k}(r_{ij}) = 1$ si $r_{ij} \in S_k$ y 0 en otro caso. S_k^c es el complementario de S_k y $|S_k|$ el cardinal de S_k.

El valor calculado depende del grupo de genes S_k y de la muestra j.

Resumen En el apartado anterior tenemos una función de s que hemos de resumir. La primera propuesta es tomar

$$E_{jk}^{max} = \nu_{jk}(s*). \tag{13.20}$$

con

$$\nu_{jk}(s*) = max\{\nu_{jk}(s) : s = 1,\ldots,N\}. \tag{13.21}$$

Una segunda opción para resumir es considerar

$$E_{jk}^{diff} = \max_{s=1,\ldots,N}\{0,\nu_{jk}(s)\} - \min_{s=1,\ldots,N}\{0,\nu_{jk}(s)\}. \tag{13.22}$$

Los autores proponen los estadísticos en 13.20 y 13.22.

13.15 gse21942

Analizamos estos datos con distintos paquetes.

13.15.1 Utilizando tami

Utilizamos t-test moderado (Limma) para el análisis de expresión diferencial marginal: `test = rowtmod`. Como medida de asociación fenotipo-expresión utilizamos el estadístico del contraste: `association="statistic''`. Contrastamos la distribución nula de aleatorización por columnas: `GeneNullDistr = "randomization"`, `GeneSetNullDistr = "self-contained"`. ¿El test es bilateral o unilateral? Utilizamos la opción bilateral: `alternative="two-sided"` (otras opciones son `"less"` o `"greater"`). El número máximo de combinaciones consideradas: `nmax = 100`. ¿Qué identificador se ha utilizado en la colección de genes?: `id = "ENTREZID"` . Utilizamos la colección de genes vista en § 13.4.2 y como medida de enriquecimiento tenemos la media muestral.

```
pacman::p_load(Biobase,tami)
data(gse21942,package="tamidata")
load(paste0(dirTamiData,"gscHs.rda"))
gse21942_self = tami::GeneSetTest(x = gse21942,y="FactorValue..DISEASE.STATE.",
                                  test = rowtmod,gsc=geneIds(gscHs),
                                  association="statistic",correction="BH",
                                  GeneNullDistr = "randomization",
                                  GeneSetNullDistr = "self-contained",
                                  alternative="two-sided",nmax = 100,
                                  id = "ENTREZID",descriptive=mean,
                                  foutput = "gse21942_self")
```

Podemos contrastar la hipótesis competitiva manteniendo todas las demás opciones.

```
gse21942_comp = tami::GeneSetTest(x = gse21942,y="FactorValue..DISEASE.STATE.",
        test = rowtmod,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "competitive",
        alternative="two-sided",nmax = 100,
        id = "ENTREZID",gsc=geneIds(gscHs),descriptive=mean,
        foutput = "gse21942_comp")
```

Los resultados los podemos tener como un **`data.frame`**.

```
gse21942_self_df = tami::tidy(gse21942_self)
gse21942_comp_df = tami::tidy(gse21942_comp)
```

Las primeras filas del **`data.frame`** las tenemos con

```
head(gse21942_self_df)
```

```
                GO statistic rawp
GO:0000002 GO:0000002 6.4281305 0.00000000
GO:0000012 GO:0000012 1.1343923 0.24242424
GO:0000014 GO:0000014 4.6320273 0.00000000
GO:0000019 GO:0000019 2.5586664 0.02020202
GO:0000022 GO:0000022 -0.4690394 0.74747475
GO:0000027 GO:0000027 4.5651419 0.00000000
              adjp
GO:0000002 0.00000000
GO:0000012 0.42851559
```

```
GO:0000014 0.00000000
GO:0000019 0.05893287
GO:0000022 0.86750086
GO:0000027 0.00000000
```

Un informe en un fichero lo tenemos con

```
glimpse(gse21942_self)
glimpse(gse21942_comp)
```

El fichero podemos generarlo y abrirlo con

```
browseURL(glimpse(gse21942_self))
browseURL(glimpse(gse21942_comp))
```

13.16 PRJNA297664

Leemos los datos.

```
data(PRJNA297664,package = "tamidata")
```

Utilizamos la colección obtenida en § 11.3 utilizando identificadores `ENTREZID` ↪ .

```
load(paste(dirTamiData,"gscSc.rda",sep=""))
```

Utilizamos como estadístico por fila el obtenido en el método descrito en § 10.3.1. Consideramos en primer lugar el test para la hipótesis autocontenida.

```
PRJNA297664_self = GeneSetTest(x = PRJNA297664,y="treatment",gsc=gscSc,
        test = edgercommon,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "self-contained",
        alternative="two-sided",nmax = 100,
        id = "ENTREZID",descriptive=mean,
        foutput = "PRJNA297664_self")
```

Y ahora consideramos la hipótesis competitiva.

```
PRJNA297664_comp = GeneSetTest(x = PRJNA297664,y="treatment",gsc=gscSc,
        test = edgercommon,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "competitive",
        alternative="two-sided",nmax = 100,
        id = "ENTREZID",descriptive=mean,
        foutput = "PRJNA297664_comp")
```

Generamos un **`data.frame`** con los resultados obtenidos.

```
PRJNA297664_self_df = tidy(PRJNA297664_self)
PRJNA297664_comp_df = tidy(PRJNA297664_comp)
```

También podemos general un informe en HTML con

```
glimpse(PRJNA297664_self)
```

```
[1] "./reports/PRJNA297664_self.html"
```

```
glimpse(PRJNA297664_comp)
```

```
[1] "./reports/PRJNA297664_comp.html"
```

Si queremos generar y abrirlo entonces usamos

```
browseURL(glimpse(PRJNA297664_self))
browseURL(glimpse(PRJNA297664_comp))
```

13.17 gse21942

Realizamos un análisis de grupo de genes para `tamidata::gse21942` ↪ . La colección de grupos la podemos obtener con

```
hsa_go = EnrichmentBrowser::getGenesets(org = "hsa",db="go",
                                        onto ="BP")
save(hsa_go,file=paste0(dirTamiData,"hsa_go.rda"))
```

Cargamos los paquetes.

```
pacman::p_load(Biobase,tami)
```

Utilizamos t-test moderado (Limma) para el análisis de expresión diferencial marginal: `test = rowtmod`. Como medida de asociación fenotipo-expresión utilizamos el estadístico del contraste: `association="statistic ''`. Contrastamos la distribución nula de aleatorización por columnas: `GeneNullDistr = "randomization"`, `GeneSetNullDistr = "self -contained"`

¿El test es bilateral o unilateral? `` `alternative="two-sided" `` (otras opciones son `"less"` o `"greater"`). El número máximo de combinaciones consideradas: `nmax = 100`. ¿Qué identificador se ha utilizado en la colección de genes?: `id = "ENTREZID"`. La colección de genes previamente bajada: `gsc=hsa_go`. Como medida de enriquecimiento tenemos la media.

```
data(gse21942,package="tamidata")
gse21942_self = tami::GeneSetTest(x = gse21942,y="FactorValue..DISEASE.STATE.",
        test = rowtmod,gsc=hsa_go,
        association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "self-contained",
        alternative="two-sided",nmax = 100,
        id = "ENTREZID",descriptive=mean,
        foutput = "gse21942_self")
```

Supongamos que queremos contrastar la hipótesis competitiva.

```
gse21942_comp = tami::GeneSetTest(x = gse21942,y="FactorValue..DISEASE.STATE.",
        test = rowtmod,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "competitive",
        alternative="two-sided",nmax = 100,
        id = "ENTREZID",gsc=hsa_go,descriptive=mean,
        foutput = "gse21942_comp")
```

Podemos tener los resultados obtenidos en un `data.frame`.

```
gse21942_self_df = tami::tidy(gse21942_self)
gse21942_comp_df = tami::tidy(gse21942_comp)
```

Un informe en un fichero lo tenemos con

```
glimpse(gse21942_self)
glimpse(gse21942_comp)
```

El fichero podemos generarlo y abrirlo con

```
browseURL(glimpse(gse21942_self))
browseURL(glimpse(gse21942_comp))
```

13.18 PRJNA297664

Son datos de la levadura. Cargamos paquetes necesarios.

```
pacman::p_load(SummarizedExperiment,edgeR,tami)
```

Necesitamos en primer lugar la colección de grupos de genes.

```
sce_go = EnrichmentBrowser::getGenesets(org="sce", onto="BP")
names(sce_go) = sapply(names(sce_go),
                       function(x)unlist(strsplit(x,split="_"))[1])
```

Leemos los datos.

```
data(PRJNA297664,package = "tamidata")
```

Comparamos los dos grupos que define la variable fenotípica `treatment` ↪ . Utilizamos el procedimiento tratado en § 10.3.1 que denominamos **edgeR clásico**. Contrastamos la hipótesis autocontenida.

```
PRJNA297664_self = tami::GeneSetTest(x = PRJNA297664,y="treatment",gsc=sce_go,
        test = edgercommon,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "self−contained",
        alternative="two−sided",nmax = 10000,
        id = "ENTREZID",descriptive=mean,
        foutput = "PRJNA297664_self")
```

Repetimos el análisis para hipótesis competitiva.

```
PRJNA297664_comp = GeneSetTest(x = PRJNA297664,y="treatment",gsc=sce_go,
        test = edgercommon,association="statistic",correction="BH",
        GeneNullDistr = "randomization",
        GeneSetNullDistr = "competitive",
        alternative="two−sided",nmax = 100,
        id = "ENTREZID",descriptive=mean,
        foutput = "PRJNA297664_comp")
```

Podemos generar un informe en con el método `glimpse`.

```
glimpse(PRJNA297664_self)
glimpse(PRJNA297664_comp)
```

Y podemos verlo con

```
browseURL(glimpse(PRJNA297664_self))
browseURL(glimpse(PRJNA297664_comp))
```

Un `data.frame` con los resultados lo tenemos con el método `tidy`.

```
tidy(PRJNA297664_self)
tidy(PRJNA297664_comp)
```

13.19 Material complementario

Un estudio comparativo de distintos métodos de análisis de grupos de genes aparece en [84].

No hemos utilizado en este tema pero es de interés [**R-topGO**].

La bibliografía sobre este tema es (muy) grande. Como ejemplo, podemos citar como bibliografía complementaria [126, 138, 35, 164, 90, 154, 94, 155, 106, 161, 40, 52].

Una revisión sobre distintos procedimientos para análisis de expresión diferencial basada en conjuntos con datos de RNASeq lo tenemos en [118].

Parte IV

Estadística

Capítulo 14

Conceptos fundamentales de Estadística

El contenido tratado en los temas anteriores es intencionadamente muy básico. Se pretende llegar al lector que no tiene ningún conocimiento de Probabilidad y Estadística. Sin embargo, hay procedimientos que se tratan en este manual y que requieren un nivel de formalización mayor. En este tema se aborda este otro nivel. Para una primera lectura del manual no se requiere. Pero sí para una lectura completa. El lenguaje utilizado es más preciso.

14.1 Verosimilitud

Sea $y = (y_1, \ldots, y_n)$ una realización del vector aleatorio $Y = (Y_1, \ldots, Y_n)$. Es habitual asumir que Y tiene una función de densidad conjunta f en una cierta familia $\mathcal{F}$. Para una función dada f, el valor $f(y)$ nos muestra cómo varía la densidad dentro del espacio muestral de valores posibles de y. Y viceversa, si consideramos unos datos y y lo que hacemos variar es la función de densidad entonces estamos viendo cómo de verosímil es cada una de las funciones dados los datos y. Esta función recibe el nombre de **verosimilitud** de f dados los datos y y se suele denotar como

$$L(f; y) = f(y). \tag{14.1}$$

Con frecuencia, es conveniente trabajar con el logaritmo natural de la función anterior y hablaremos de la **log-verosimilitud**.

$$\ell(f; y) = \ln f(y). \tag{14.2}$$

Una simplificación adicional (que es habitual en las aplicaciones) supone que la función de densidad f pertenece a una familia paramétrica $\mathcal{F}$, esto es, cada elemento de la familia es conocido completamente salvo un número finito de parámetros $\theta = (\theta_1, \ldots, \theta_p)$ de modo que denotaremos $f(y; \theta)$, $f_Y(y; \theta)$ o $f(y|\theta)$.

Al conjunto de valores posibles de θ se le llama **espacio paramétrico** y lo denotaremos por Θ. En este caso, la logverosimilitud es una función de θ y denotaremos

$$\ell(\theta; y) = \ln f(y; \theta). \tag{14.3}$$

Supongamos una transformación 1-1 de Y a Z, $Z = g(Y)$. Las densidades de ambos vectores se relacionan según la siguiente relación

$$f_Z(z) = f_Y(y)\left|\frac{\partial y}{\partial z}\right|,$$

donde $\left|\frac{\partial y}{\partial z}\right|$ es el jacobiano de la transformación de z a y. Se tiene la siguiente relación entre las verosimilitudes

$$L_Z(\theta; z) = \left|\frac{\partial y}{\partial z}\right| L_Y(\theta; y).$$

Esto sugiere que es mejor trabajar con el cociente de las verosimilitudes para dos vectores de parámetros θ_1 y θ_2 en lugar de los valores aislados. Si asumimos que los distintos $Y_1, \ldots, Y_n$ son independientes entonces

$$L_Y(\theta; y) = f_Y(y) = \prod_{i=1}^{n} f_{Y_i}(y_i),$$

y

$$\ell_y(\theta; y) = \sum_{i=1}^{n} \ln f_{Y_i}(y_i) = \sum_{i=1}^{n} \ell(\theta; y_i).$$

Veamos algunos ejemplos de verosimilitud con modelos que usamos posteriormente.

Ejemplo 14.1 (Pruebas Bernoulli)**.** *$Y_1, \ldots, Y_n$ son independientes y con la misma distribución (i.i.d.) $P(Y_i = y_i) = \theta^{y_i}(1-\theta)^{1-y_i}$ y*

$$L(\theta; y) = \theta^{\sum_{i=1}^{n} y_i}(1-\theta)^{n-\sum_{i=1}^{n} y_i}$$

Ejemplo 14.2 (Número de éxitos en n pruebas Bernoulli)**.** *Nuestros datos son ahora el número total de éxitos en un número dado de pruebas de Bernoulli, r. Entonces la variable correspondiente R tiene una distribución binomial con n pruebas y una probabilidad de éxito θ. La verosimilitud viene dada por*

$$L(\theta; r) = \binom{n}{r}\theta^r(1-\theta)^{n-r}$$

Ejemplo 14.3 (Binomial negativa)**.** *Nuestros datos son ahora el número total de pruebas necesarias para alcanzar un número previamente especificado de éxitos. La variable aleatoria correspondiente N tendrá una distribución binomial negativa con r éxitos y una probabilidad de éxito θ. La función de verosimilitud correspondiente viene dada por*

$$L(\theta; n) = \binom{n-1}{r-1}\theta^r(1-\theta)^{n-r}$$

Consideremos los tres ejemplos anteriores 14.1, 14.2 y 14.3. Si consideramos dos valores del parámetro θ_1 y θ_2 entonces el cociente de las verosimilitudes calculados en ambos valores tiene el mismo valor en los tres ejemplos.

14.2 Estimación

Denotamos por Θ el espacio formado por los valores que puede tomar θ o espacio paramétrico.

Definición 14.1. *Un **estimador** del parámetros o vector paramétrico θ es cualquier función de la muestra $X_1, \ldots, X_n$ que toma valores en el espacio paramétrico.*

Si $\delta(X_1, \ldots, X_n)$ es un estimador del parámetro θ entonces se define el **error cuadrático medio** como

$$MSE(\delta) = E[\delta(X_1, \ldots, X_n) - \theta]^2 \tag{14.4}$$

En el caso en que se verifique que $E\delta(X_1, \ldots, X_n) = \mu_\delta = \theta$, es decir, que el estimador sea **insesgado** entonces:

$$MSE(\delta) = E[\delta(X_1, \ldots, X_n) - \theta]^2 = E[\delta(X_1, \ldots, X_n) - \mu_\delta]]^2 = var(\delta).$$

Para estimadores insesgados el error cuadrático medio no es más que la varianza del estimador. Consideremos la siguiente cadena de igualdades. Denotamos

$$MSE(\delta) = E[\delta - \theta]^2 = E[\delta - \mu_\delta + \mu_\delta - \theta]^2 = E[\delta - \mu_\delta]^2 + [\mu_\delta - \theta]^2 \tag{14.5}$$

La diferencia entre la media del estimador y el parámetro, $\mu_\delta - \theta$, recibe el nombre de **sesgo**. Finalmente lo que nos dice la ecuación anterior es que el error cuadrático medio $MSE(\delta)$ lo podemos expresar como la suma de la varianza del estimador, $E[\delta - \mu_\delta]^2$, más el sesgo al cuadrado, $[\mu_\delta - \theta]^2$.

A la raíz cuadrada de la varianza de un estimador, es decir, a su desviación típica o estándar se le llama **error estándar**. La expresión error estándar se usa en ocasiones indistintamente para referirse o bien dicha desviación típica o bien al estimador de la misma.

14.2.1 Estimación insesgada de media y varianza

Dada una muestra $Y_1, \ldots, Y_n$ de una variable. Un estimador habitualmente utilizado para estimar $\mu = EY_i$ es la media muestral dada por

$$\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i. \tag{14.6}$$

Notemos que

$$E\bar{Y} = E\left[\frac{1}{n} \sum_{i=1}^{n} Y_i\right] = \frac{1}{n} \sum_{i=1}^{n} EY_i = \frac{1}{n} \sum_{i=1}^{n} \mu = \mu.$$

En definitiva, la media muestral es un estimador que no tiene ningún sesgo cuando estima la media de Y_i (la media poblacional) o, lo que es lo mismo, es un estimador **insesgado**.

Para estimar de un modo insesgado la varianza σ^2 a partir de una muestra $Y_1, \ldots, Y_n$ se utiliza la varianza muestral dada por

$$S^2 = \frac{1}{n-1} \sum_{i=1}^{n} (Y_i - \bar{Y})^2. \tag{14.7}$$

La razón de la división por $n-1$ en lugar de dividir por n viene de las siguientes igualdades.

$$E\sum_{i=1}^{n}(Y_i-\bar{Y})^2 = E\sum_{i=1}^{n}[(Y_i-\mu)-(\bar{Y})^2-\mu)]^2 = \sum_{i=1}^{n}E(Y_i-\mu)^2 - nE(\bar{Y}-\mu)^2, \quad (14.8)$$

pero $E(Y_i-\mu)^2=\sigma^2$ y $E(\bar{Y}-\mu)^2 = var(\bar{Y}) = \sigma^2/n$. En consecuencia,

$$E\sum_{i=1}^{n}(Y_i-\bar{Y})^2 = n\sigma^2 - \frac{\sigma^2}{n} = n\sigma^2-\sigma^2 = (n-1)\sigma^2,$$

de donde,

$$ES^2 = E\frac{1}{n-1}\sum_{i=1}^{n}(Y_i-\bar{Y})^2 = \sigma^2,$$

es decir, S^2 estima la varianza σ^2 sin sesgo.

14.2.2 Estimación insesgada del vector de medias y la matriz de covarianzas

Ahora consideramos una muestra de un vector de dimensión d, $Y_1,\ldots,Y_n$ i.i.d. con vector de medias $\mu = EY_i$ y matriz de covarianzas $\Sigma = cov(Y_i)$. Los estimadores insesgados de μ y Σ son las versiones multivariantes de la media y varianza muestrales. Si

$$Y_i = \begin{bmatrix} Y_{i1} \\ \vdots \\ Y_{ip} \end{bmatrix}$$

Entonces podemos representar toda la muestra como la siguiente matriz

$$\boldsymbol{Y} = \begin{bmatrix} \boldsymbol{Y}_1^T \\ \vdots \\ \boldsymbol{Y}_n^T \end{bmatrix} = \begin{bmatrix} Y_{11} & \ldots & Y_{1d} \\ \vdots & \vdots & \vdots \\ Y_{n1} & \ldots & Y_{nd} \end{bmatrix}$$

mientras que los datos observados, la matriz de datos, vendría dada por

$$\boldsymbol{y} = \begin{bmatrix} \boldsymbol{y}_1^T \\ \vdots \\ \boldsymbol{y}_n^T \end{bmatrix} = \begin{bmatrix} y_{11} & \ldots & y_{1d} \\ \vdots & \vdots & \vdots \\ y_{n1} & \ldots & y_{nd} \end{bmatrix}$$

El vector de medias muestral viene dado por la siguiente expresión en términos de la matriz $\boldsymbol{Y}$,

$$\bar{\boldsymbol{Y}} = \frac{1}{n}\sum_{i=1}^{n}Y_i = \frac{1}{n}\boldsymbol{Y}^T\mathbf{1}_n. \quad (14.9)$$

siendo $\mathbf{1}_n$ el vector $n\times 1$ con todos los valores iguales a uno. También denotaremos

$$\bar{\boldsymbol{Y}} = \begin{bmatrix} \bar{Y}_{\cdot 1} \\ \vdots \\ \bar{Y}_{\cdot p} \end{bmatrix}$$

El estimador de la matriz de covarianzas (poblacional) Σ sería la matriz de covarianzas muestral que tiene en la posición (j,k) la covarianza muestral entre las componentes j y k,

$$S_{jk} = \frac{1}{n-1}\sum_{i=1}^{n}(Y_{ij} - \bar{Y}_{\cdot j})(Y_{ik} - \bar{Y}_{\cdot k}),$$

de modo que

$$S = \begin{bmatrix} S_{11} & \dots & S_{1d} \\ \vdots & \vdots & \vdots \\ S_{d1} & \dots & S_{dd} \end{bmatrix} = \frac{1}{n-1}\sum_{i=1}^{n}(Y_i - \bar{Y})(Y_i - \bar{Y})^T = \frac{1}{n-1}Q.$$

Es inmediato que $E\bar{\boldsymbol{Y}} = \boldsymbol{\mu}$ porque componente a componente hemos visto que se verifica la igualdad. A partir de los vectores Y_i consideramos $X_i = Y_i - \mu$ de modo que se verifica $\bar{\boldsymbol{X}} = \bar{\boldsymbol{X}} - \mu$. Se sigue que

$$\sum_{i=1}^{n}(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}})(\boldsymbol{Y}_i - \bar{\boldsymbol{Y}})^T = \sum_{i=1}^{n}(\boldsymbol{X}_i - \bar{\boldsymbol{X}})(\boldsymbol{X}_i - \bar{\boldsymbol{X}})^T = \sum_{i=1}^{n}\boldsymbol{X}_i\boldsymbol{X}_i^T - n\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T.$$

Los vectores $\boldsymbol{X}_1, \dots, \boldsymbol{X}_n$ tienen vector de medias nulo y matriz de covarianzas $\boldsymbol{\Sigma}$, la misma que los $\boldsymbol{Y}_i$. En consecuencia, $E\bar{\boldsymbol{X}}\bar{\boldsymbol{X}}^T = \boldsymbol{\Sigma}$ y

$$E\boldsymbol{Q} = \sum_{i=1}^{n} cov(\boldsymbol{Y}_i) - n\, cov(\bar{\boldsymbol{Y}}) = n\boldsymbol{\Sigma} - n\, cov(\bar{\boldsymbol{Y}}) = n\boldsymbol{\Sigma} - n\frac{\boldsymbol{\Sigma}}{n} = (n-1)\boldsymbol{\Sigma}.$$

Tenemos pues que $\boldsymbol{S}$ es un estimador insesgado de la matriz $\boldsymbol{\Sigma}$.

Finalmente, si denotamos por r_{jk} el coeficiente de correlación entre las variables j y k, es decir,

$$r_{jk} = \frac{\sum_{i=1}^{n}(Y_{ij} - \bar{Y}_{\cdot j})(Y_{ik} - \bar{Y}_{\cdot k})}{\sqrt{\sum_{i=1}^{n}(Y_{ij} - \bar{Y}_{\cdot j})^2 \sum_{i=1}^{n}(Y_{ik} - \bar{Y}_{\cdot k})^2}} = \frac{S_{jk}}{\sqrt{S_{jj}S_{kk}}} \tag{14.10}$$

Denotaremos por R la **matriz de correlaciones muestrales** $R = [r_{jk}]$.

14.3 Estimador máximo verosímil

El método de estimación que vamos a utilizar en este curso el **método de máxima verosimilitud**. El estimador máximo verosímil de θ, que denotaremos por $\hat{\theta}$, se obtienen máximizando la función de verosimilitud o, equivalentemente, la transformación monótona de dicha función que es la función de logverosimilitud. Utilizaremos para denotar el estimador máximo verosímil la notación inglesa **MLE**.

$$L(\hat{\theta}) = \max_{\theta\in\Theta} L(\theta), \tag{14.11}$$

o también

$$\hat{\theta} = argmax_{\theta\in\Theta} L(\theta), \tag{14.12}$$

Ejemplo 14.4 (Bernoulli)**.** *Se puede comprobar sin dificultad que* $\hat{p} = \frac{\sum_{i=1}^{n} x_i}{n}$.

Una propiedad importante de los estimadores máximo verosímiles consiste en que si $\theta^* = f(\theta)$ siendo f una biyección entonces el estimador máximo verosímil de θ^* es verifica que

$$\hat{\theta^*} = f(\hat{\theta}). \tag{14.13}$$

Ejemplo 14.5 (Normal)**.** *En este caso se comprueba que* $\hat{\mu} = \hat{X}_n$ *y que* $\widehat{\sigma^2} = \frac{n-1}{n}S^2 = \frac{1}{n}\sum_{i=1}^{n}(X_i - \bar{X}_n)^2$*. Teniendo en que cuenta la propiedad enunciada en 14.13 tendremos que* $\hat{\sigma} = \sqrt{\frac{n-1}{n}S^2}$*.*

En muchas situaciones la función $L(\theta)$ es cóncava y el estimador máximo verosímil $\hat{\theta}$ es la solución de las **ecuaciones de verosimilitud** $\frac{\partial L(\theta}{\partial \theta} = 0$. Si $cov(\hat{\theta})$ denota la matriz de covarianzas de $\hat{\theta}$ entonces, para un tamaño muestral grande y bajo ciertas condiciones de regularidad (ver [119], página 364), se verifica que $cov(\hat{\theta})$ es la inversa de la **matriz de información** cuyo elemento (j,k) viene dado por

$$-E\left(\frac{\partial^2 l(\theta)}{\partial\theta_j \partial\theta_k}\right) \tag{14.14}$$

Notemos que el error estándar de $\hat{\theta}_j$ será el elemento que ocupa la posición (j,j) en la inversa de la matriz de información. Cuanto mayor es la curvatura de la logverosimilitud menores serán los errores estándar. La racionalidad que hay detrás de esto es que si la curvatura es mayor entonces la logverosimilitud cae rápidamente cuando el vector θ se aleja de $\hat{\theta}$. En resumen, es de esperar que θ esté más próximo a $\hat{\theta}$.

Ejemplo 14.6 (Binomial)**.** *Supongamos que una muestra en una población finita y consideremos como valor observado el número de éxitos. Entonces la verosimilitud sería*

$$L(p) = \binom{n}{y} p^y (1-p)^{n-y}, \tag{14.15}$$

y la logverosimilitud viene dada como

$$l(p) = \log\binom{n}{y} + y\log p + (n-y)\log(1-p), \tag{14.16}$$

La ecuación de verosimilitud sería

$$\frac{\partial l(p)}{\partial p} = \frac{y}{p} - \frac{n-y}{1-p} = \frac{y-np}{p(1-p)}. \tag{14.17}$$

Igualando a cero tenemos que la solución es $\hat{p} = \frac{y}{n}$ que no es más que la proporción muestral de éxitos en las n pruebas. La varianza asíntótica sería

$$-E\left[\frac{\partial^2 l(p)}{\partial p^2}\right] = E\left[\frac{y}{p^2} + \frac{n-y}{(1-p)^2}\right] = \frac{n}{p(1-p)}. \tag{14.18}$$

En consecuencia asintóticamente $\hat{p}$ tiene varianza $\frac{p(1-p)}{n}$ lo cual era de prever pues si consideramos la variable Y que nos da el número de éxitos entonces sabemos que $EY = np$ y que $var(Y) = np(1-p)$.

	H_0	H_1
Rechazamos H_0	Error tipo I	
No rechazamos H_0		Error tipo II

14.4 Contraste de hipótesis

Genéricamente vamos a considerar situaciones en donde particionamos el espacio paramétrico Θ en dos conjuntos Θ_0 y Θ_1, es decir, $\Theta_0 \cap \Theta_1 = \emptyset$ (son disjuntos) y y $\Theta_0 \cup \Theta_1 = \Theta$ (cubren todo el espacio paramétrico). Consideramos el contraste de hipótesis siguiente.

$$H_0 : \theta \in \Theta_0 \tag{14.19}$$

$$H_1 : \theta \in \Theta_1 \tag{14.20}$$

Basándonos en una muestra aleatoria $X_1, \ldots, X_n$ hemos de tomar una decisión. Las decisiones a tomar son una entre dos posibles: (i) Rechazar la hipótesis nula o bien (ii) no rechazar la hipótesis nula. Notemos que, una vez hemos tomado una decisión, podemos tener dos posibles tipos de error como recoge la siguiente tabla. En las columnas indicamos la realidad mientras que en las filas indicamos la decisión que tomamos.

Supongamos que $\mathbb{R}^n$ es el conjunto de valores que puede tomar el vector aleatorio $(X_1, \ldots, X_n)$. Entonces el contraste de hipótesis se basa en tomar un estadístico o función de la muestra que denotamos $\delta(X_1, \ldots, X_n)$ de modo que si $\delta(X_1, \ldots, X_n) \in C$ entonces rechazamos la hipótesis nula mientras que si $\delta(X_1, \ldots, X_n) \notin C$ entonces no rechazamos la hipótesis nula. Notemos que simplemente estamos particionando el espacio muestral (que suponemos) $\mathbb{R}^n$ en dos partes, C y C^c, de modo que tomamos una decisión basándonos en si el estadístico δ está en C o bien está en el complementario de C. Al conjunto C se le suele llamar la **región crítica**. La función potencia se define como

$$\pi(\theta) = P(\delta \in C|\theta). \tag{14.21}$$

14.4.1 Test del cociente de verosimilitudes

El cociente de verosimilitudes para contrastar estas hipótesis se define como

$$\Lambda = \frac{\max_{\theta \in \Theta_0} L(\theta)}{\max_{\theta \in \Theta} L(\theta)} \tag{14.22}$$

Es razonable pensar que en la medida en que Λ tome valores menores entonces la hipótesis alternativa sea más plausible que la hipótesis nula y por lo tanto rechacemos la hipótesis nula. Realmente se suele trabajar con $-2\log\Lambda$ pues bajo la hipótesis nula tiene una distribución asintótica ji-cuadrado donde el número de grados de libertad es la diferencia de las dimensiones de los espacios paramétricos $\Theta = \Theta_0 \cup \Theta_1$ y Θ_0. Si denotamos $L_0 = \max_{\theta \in \Theta_0} L(\theta)$ y $L_1 = \max_{\theta \in \Theta} L(\theta)$ entonces $\Lambda = \frac{L_0}{L_1}$ y

$$-2\log\lambda = -2\log\frac{L_0}{L_1} = -2(l_0 - l_1) \tag{14.23}$$

siendo l_0 y l_1 los logaritmos de L_0 y L_1 respectivamente que también corresponden con los máximos de la logverosimilitud sobre Θ_0 y sobre Θ.

14.4.2 Test de Wald

Supongamos que el θ es un parámetro y $\hat{\theta}$ denota su estimador máximo verosímil. Supongamos que queremos contrastar las siguientes hipótesis:

$$H_0 : \theta = \theta_0, \tag{14.24}$$
$$H_1 : \theta \neq \theta_0. \tag{14.25}$$

Denotamos por $SE(\hat{\theta})$ el error estándar bajo la hipótesis alternativa de $\hat{\theta}$. Entonces el estadístico

$$z = \frac{\hat{\theta} - \theta_0}{SE(\hat{\theta})} \tag{14.26}$$

tiene, bajo la hipótesis nula, aproximadamente una distribución normal estándar, $z \sim N(0,1)$. Este tipo de estadísticos donde se utiliza el error estándar del estimador bajo la hipótesis alternativa recibe el nombre de *estadístico de Wald*.

Supongamos que θ es un vector de parámetros y queremos contrastar las hipótesis dadas en 14.24. La versión multivariante del estadístico dado en 14.26 viene dada por

$$W = (\hat{\theta} - \theta_0)^T [cov(\hat{\theta})]^{-1} (\hat{\theta} - \theta_0), \tag{14.27}$$

donde $cov(\hat{\theta})$ se estima como la matriz de información observada en el MLE $\hat{\theta}$. La distribución asintótica de W bajo la hipótesis nula es una distribución ji-cuadrado donde el número de grados de libertad coincide con el número de parámetros no redundantes en θ.

14.4.3 Intervalos de confianza

Empezamos recordando el concepto de intervalo de confianza con un ejemplo muy conocido como es la estimación de la media en poblaciones normales.

Ejemplo 14.7 (Intervalo de confianza para la media de una normal)**.** *Veámoslo con un ejemplo y luego planteamos la situación más general. Tenemos una muestra aleatoria $X_1, \ldots, X_n$ i.i.d. tales que $X_i \sim N(\mu, \sigma 2)$. Entonces es conocido que*

$$\frac{\bar{X}_n - \mu}{S/\sqrt{n}} \sim t_{n-1}. \tag{14.28}$$

Vemos cómo $\frac{\bar{X}_n - \mu}{S/\sqrt{n}}$ depende tanto de la muestra que conocemos como de un parámetro (la media μ) que desconocemos. Fijamos un valor de α (habitualmente tomaremos $\alpha = 0.05$) y elegimos un valor $t_{n-1,1-\alpha/2}$ tal que

$$P(-t_{n-1,1-\alpha/2} \leq \frac{\bar{X}_n - \mu}{S/\sqrt{n}} \leq t_{n-1,1-\alpha/2}) = 1 - \alpha. \tag{14.29}$$

La ecuación anterior la podemos reescribir como

$$P(\bar{X}_n - t_{n-1,1-\alpha/2}\frac{S}{\sqrt{n}} \leq \mu \leq \bar{X}_n + t_{n-1,1-\alpha/2}\frac{S}{\sqrt{n}}) = 1 - \alpha. \tag{14.30}$$

Tenemos una muestra aleatoria $X_1, \ldots, X_n$ y por lo tanto tenemos un intervalo aleatorio dado por $[\bar{X}_n - t_{n-1,1-\alpha/2}\frac{S}{\sqrt{n}}, \bar{X}_n + t_{n-1,1-\alpha/2}\frac{S}{\sqrt{n}}]$. Este

intervalo tiene una probabilidad de $1-\alpha$ *de contener a la verdadera media. Tomemos ahora la muestra y consideremos no los valores aleatorios de* $\bar{X}_n$ *y de* S^2 *sino los valores observados* $\bar{x}_n$ *y* s*. Tenemos ahora un intervalo* $[\bar{x}_n - t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}}, \bar{x}_n + t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}}]$ *fijo. Es posible que* μ *esté en este intervalo y es posible que no lo esté. Sabemos que antes de tomar la muestra teníamos una* probabilidad *de* $1-\alpha$ *de contener a la verdadera media pero después de tomar la muestra tenemos una* ***confianza*** *de* $1-\alpha$ *de contener a la verdadera media. Al intervalo* $[\bar{x}_n - t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}}, \bar{x}_n + t_{n-1,1-\alpha/2}\frac{s}{\sqrt{n}}]$ *se le llama* ***intervalo de confianza*** *para* μ *con* ***nivel de confianza*** $1-\alpha$*.*

Vamos a ver un planteamiento más general del problema. Supongamos que tenemos un test para contrastar la hipótesis simple $H_0 : \theta = \theta_0$ frente a la alternativa $H_1 : \theta \neq \theta_0$. Supongamos que elegimos un nivel de significación α para contrastar las hipótesis anteriores y consideramos el siguiente conjunto formado por todos los θ_0 tales que no rechazamos la hipótesis nula al nivel α. Este conjunto es un **conjunto de confianza** al nivel $1-\alpha$. Cuando el conjunto de confianza es un intervalo hablamos de **intervalo de confianza**.

Supongamos que consideramos el test del cociente de verosimilitudes. Denotemos por $\chi^2_k(1-\alpha)$ el percentil $1-\alpha$ de una distribución ji-cuadrado con k grados de libertad. Entonces el intervalo de confianza al nivel $1-\alpha$ sería el conjunto

$$\{\theta_0 : -2[l(\theta_0) - l(\hat{\theta})] < \chi^2_k(1-\alpha)\} \tag{14.31}$$

Consideremos ahora un test de Wald. En este caso, el intervalo de confianza de Wald vendría dado por el siguiente conjunto:

$$\{\theta_0 : \frac{|\hat{\theta} - \theta_0|}{SE(\hat{\theta})} < Z_{1-\alpha/2}\} \tag{14.32}$$

donde $SE(\hat{\theta})$ es el error estándar estimado de $\hat{\theta}$ bajo la hipótesis alternativa.

Capítulo 15

Datos categóricos

En este tema vemos un breve resumen de análisis estadístico de datos categóricos.[134]

Se entiende que tenemos un problema de datos categóricos cuando nuestra variable relevante, lo que hemos llamado variable respuesta, es una variable de tipo categórico: nominal u ordinal.

En este tema nos ocupamos de lo básico cuando trabajamos con este tipo de datos. Las distribuciones de probabilidad fundamentales en este contexto son la binomial, Poisson, binomial negativa y la hipergeométrica.

[134] Utilizamos fundamentalmente los tres primeros capítulos de [4]. El código de R se obtiene de [136].

15.1 Inferencia con la distribución binomial

La verosimilitud de una Bernoulli viene dada por

$$f(y|\pi) = \pi^y(1-\pi)^{1-y},$$

donde $y = 0, 1$. Si consideramos n observaciones $Y_1, \ldots, Y_n$ i.i.d. esto es una muestra aleatoria y los valores observados los denotamos por $y_1, \ldots, y_n$ entonces la función de verosimilitud viene dada por

$$L(\pi) = L(\pi; y_1, \ldots, y_n) = \prod_{i=1}^{n} \pi^{y_i}(1-\pi)^{1-y_i} = \pi^{\sum_{i=1}^n y_i}(1-\pi)^{n-\sum_{i=1}^n y_i},$$

donde $\boldsymbol{y} = (y_1, \ldots, y_n)$. La logverosimilitud viene dada por

$$\ell(\pi) = \left(\sum_{i=1}^{n} y_i\right)\log\pi + \left(n - \sum_{i=1}^{n} y_i\right)\log(1-\pi).$$

Fácilmente podemos comprobar que el estimador máximo verosímil es

$$\hat{\pi} = \sum_{i=1}^{n} \frac{y_i}{n}. \tag{15.1}$$

Es claro que $\boldsymbol{Y} = \sum_{i=1}^n Y_i \sim Bi(n, \pi)$.[1]

[1] En § 14.1 se incluye un comentario en donde se discute cómo distintos modelos probabilísticos producen verosimilitudes proporcionales.

15.1.1 Contraste de hipótesis

Consideremos la hipótesis nula simple consistente en que la probabilidad de éxito, π, toma un valor dado, esto es, el contraste

$$\begin{aligned} H_0 : \pi &= \pi_0 \\ H_1 : \pi &\neq \pi_0. \end{aligned}$$

El estadístico score sería

$$Z_S = \frac{\hat{\pi} - \pi_0}{\sqrt{\pi_0(1-\pi_0)/n}}. \tag{15.2}$$

Bajo la hipótesis nula Z_S se distribuye aproximadamente como una normal estándar.

El estadístico de Wald (§ 14.4.2) viene dado por

$$Z_W = \frac{\hat{\pi} - \pi_0}{\sqrt{\hat{\pi}(1-\hat{\pi})/n}}, \tag{15.3}$$

y tiene aproximadamente una distribución normal estándar. Finalmente si consideramos el test del cociente de verosimilitudes (§ 14.4.1) adopta la expresión

$$-2\log\Lambda = 2\left(y\log\frac{y}{n\pi_0} + (n-y)\log\frac{n-y}{n-n\pi_0}\right), \tag{15.4}$$

donde Λ denota el cociente de verosimilitudes. La expresión dada en 15.4 no es más que

$$2\sum Observado \log\frac{Observado}{Esperado}.$$

Bajo H_0, esto es que la probabilidad de éxito es π_0, el test del cociente de verosimilitudes tiene una distribución asintótica que es una ji-cuadrado con un grado de libertad, χ^2_1.

15.1.2 Intervalos de confianza

A partir de los estadísticos anteriores se obtiene fácilmente el correspondiente intervalo de confianza, con nivel de confianza $1-\alpha$, como los valores π_0 para los cuales no rechazamos la hipótesis nula con un nivel de significación α. El intervalo de confianza de Wald para π viene dado por

$$\{\pi_0 : |z_W| < Z_{1-\alpha/2}\},$$

donde $Z_{1-\alpha/2}$ denota el percentil $1-\alpha/2$ de la normal estándar. El intervalo adopta la siguiente expresión

$$\hat{\pi} \pm Z_{1-\alpha/2}\sqrt{\frac{\hat{\pi}(1-\hat{\pi})}{n}}.$$

No es un buen intervalo cuando n es pequeño fundamentalmente para valores de π próximos a 0 o a 1.

Si utilizamos el estadístico score entonces los extremos los obtenemos resolviendo las ecuaciones

$$\frac{\hat{\pi} - \pi_0}{\sqrt{\pi_0(1-\pi_0)/n}} = \pm Z_{1-\alpha/2}.$$

La expresión final es bastante compleja pues tenemos una función cuadrática en π_0. Finalmente el intervalo de confianza del cociente de verosimilitudes no tiene una expresión sencilla y viene dado por

$$\{\pi_0 : -2\log\Lambda \leq \chi_1^2(\alpha)\}$$

Ejemplo 15.1 (Vegetarianos)**.** *Preguntamos si se es o no vegetariano. Se considera como éxito ser vegetariano y como fracaso no serlo. Supongamos un tamaño muestral de $n = 55$ y ninguna persona se declara vegetariana, esto es, $y = 0$. Veamos cómo construir los intervalos de confianza de Wald, score y del cociente de verosimilitudes con R. Consideremos el intervalo de Wald para el parámetro de una binomial.*

```
pacman::p_load(Hmisc)
Hmisc::binconf(x = 0, n = 25, method = "asymptotic")
```

```
PointEst Lower Upper
       0 0 0
```

Para el intervalo score lo podemos obtener mediante dos opciones. La primera es

```
res = prop.test(x = 0, n = 25, conf.level = 0.95, correct = FALSE)
res$conf.int
```

```
[1] 0.0000000 0.1331923
attr(,"conf.level")
[1] 0.95
```

La segunda opción sería

```
Hmisc::binconf(x = 0, n = 25, alpha = 0.05, method = "wilson")
```

```
PointEst Lower Upper
       0 0 0.1331923
```

15.2 Inferencia para la multinomial

Los parámetros a estimar son $(\pi_1, \ldots, \pi_I)$ aunque $\pi_I = 1 - \sum_{i=1}^{I-1} \pi_i$. Tomamos una muestra de tamaño n y suponemos que tenemos y_j en la categoría j. La verosimilitud es proporcional a

$$\prod_{j=1}^{I} \pi_j^{y_j} \text{ donde } \pi_j \geq 0, \sum_j \pi_j = 1.$$

Los estimadores máximo verosímiles se obtienen fácilmente igualando las derivadas parciales a cero y son

$$\hat{\pi}_j = \frac{y_j}{n}.$$

15.2.1 Contraste de hipótesis

Consideremos el siguiente contraste.

$$\begin{aligned} H_0 &: \pi_j = \pi_{j0} \text{ con } j = 1, \ldots, I \\ H_1 &: \text{No } H_0. \end{aligned}$$

Bajo H_0 *esperamos* observar de la categoría j, $\mu_j = n\pi_j$ (frecuencia esperada) pero observamos y_j (frecuencia observada). El estadístico de ji-cuadrado de Karl Pearson es el siguiente

$$X^2 = \sum_j \frac{(y_j - \mu_j)^2}{\mu_j} = \sum_j \frac{(y_j - n\pi_{j0})^2}{n\pi_{j0}}.$$

Bajo H_0, y para grandes muestras, se tiene la siguiente distribución asintótica (sin prueba)

$$X^2 \sim \chi^2_{I-1}.$$

El p-valor sería

$$p = P(X^2 \geq X_0^2),$$

donde X_0^2 es el valor observado de X^2.

Ejemplo 15.2 (Mendel)**.** *Cruzamos guisantes amarillos puros con guisantes verdes puros. El carácter dominante es el amarillo. De acuerdo con la teoría de Mendel* $3/4$ *tienen que ser amarillos y* $1/4$ *verdes. En uno de los experimentos de Mendel tenemos* $n = 8023$*, de los cuales* $n_1 = 6022$ *amarillos y* $n_2 = 2001$ *verdes. Aplicamos el test ji-cuadrado:*

```
stats::chisq.test(x = c(6022, 2001), p = c(0.75, 0.25))
```

```
        Chi-squared test for given probabilities

data: c(6022, 2001)
X-squared = 0.014999, df = 1, p-value =
0.9025
```

Fisher (y otros) vieron que tuvo demasiada suerte en su experimentación.

```
stats::chisq.test(x = c(6022, 2001), p = c(0.75, 0.25))
```

```
        Chi-squared test for given probabilities

data: c(6022, 2001)
X-squared = 0.014999, df = 1, p-value =
0.9025
```

Consideremos el test del cociente de verosimilitudes. Tenemos el contraste:

$$\begin{aligned} H_0 &: \pi_j = \pi_{j0} \text{ con } j = 1, \ldots, c \\ H_1 &: \text{No } H_0. \end{aligned}$$

El cociente de verosimilitudes viene dado por

$$\Lambda = \frac{\prod_j \pi_{j0}^{n_j}}{\prod_j (n_j/n)^{n_j}}$$

El estadístico del cociente de verosimilitudes es

$$G^2 = -2\log\Lambda = 2\sum_j y_j \log\frac{y_j}{n\pi_{j0}}.$$

Con n grande, bajo H_0, tenemos

$$G^2 \sim \chi^2_{I-1}.$$

Tabla 15.1: Aspirina y ataque cardíaco.

	Ataque fatal	Ataque no fatal	No ataque
Placebo	18	171	10845
Aspirina	5	99	10933

Tabla 15.2: Aspirina y ataque cardíaco. Consideramos si ha habido ataque frente a que no lo ha habido.

	Ataque	No ataque
Placebo	189	10845
Aspirina	104	10933

15.3 Probabilidad y tablas de contingencia

En esta sección nos ocupamos del estudio de la distribución conjunta de dos variables categóricas. Supondremos dos variables categóricas X e Y con I y J categorías respectivamente. Un individuo puede venir clasificado en una de $I \times J$ categorías.

15.3.1 Distribución conjunta y tabla de contingencia

Consideremos los datos en tabla 15.1. Se trata de estudiar el posible efecto preventivo del uso de la aspirina para prevenir ataques al corazón. Se realizó un estudio aleatorizado durante cinco años ([116]) de modo que se asignó al azar a cada persona en dos posibles grupos. En un grupo los participantes tomaron una aspirina cada día. En el otro tomaron un placebo. Los participantes no sabían si tomaban aspirina o placebo. Todos los participantes eran médicos. Obviamente lo primero que hacemos con los datos es construir una tabla en que aparecen las frecuencias conjuntas donde consideramos la variable X que nos indica si el individuo toma aspirina o placebo ($I = 2$) y la variable Y que nos indica si ha tenido ataque cardíaco fatal, un ataque no fatal o bien no ha tenido ninguno ($J = 3$). Esta tabla recibe el nombre de *tabla de contingencia* o *tabla de clasificación cruzada*. También consideraremos la situación simplificada en que agrupamos los valores de ataque fatal y no fatal de modo que la variable Y pasa a tener dos categorías.

Tenemos dos variables discretas. Su **distribución conjunta** viene dada por

$$\pi_{ij} = P(X = i, Y = j),$$

con $i = 1, \ldots, I$ y $j = 1, \ldots, J$. Las **distribuciones marginales** son, para la variable X,

$$\pi_{i+} = P(X = i) = \sum_{j=1}^{J} P(X = i, Y = j) = \sum_{j=1}^{J} \pi_{ij}$$

y para la variable Y

$$\pi_{+j} = P(Y = j) = \sum_{i=1}^{I} P(X = i, Y = j) = \sum_{i=1}^{I} \pi_{ij}$$

Habitualmente una variable (por ejemplo, Y) es una variable respuesta y la otra (X) es explicativa o predictora. En esta situación no tiene sentido hablar de distribución conjunta. Es natural considerar la distribución

Tabla 15.3: Tabla de contingencia 2×2.

	Test positivo	Test negativo	Total
Enfermo	n_{11}	n_{11}	n_{1+}
No enfermo	n_{21}	n_{22}	n_{2+}
Total	n_{+1}	n_{+2}	n

Tabla 15.4: Tabla de frecuencias relativas sobre el total de la tabla.

$\hat{\pi}_{ij}$	Test positivo	Test negativo	Total
Enfermo	n_{11}/n	n_{11}/n	n_{1+}/n
No enfermo	n_{21}/n	n_{22}/n	n_{2+}/n
Total	n_{+1}/n	n_{+2}/n	1

condicionada de Y a X viene dada por

$$P(Y = j|X = i) = \pi_{j|i} = \frac{\pi_{ij}}{\pi_{i+}}$$

Se dice que las dos variables son **independientes** si

$$\pi_{ij} = \pi_{i+}\pi_{+j},$$

para todo i, j. En particular, la distribución condicionada es igual a la distribución marginal si las variables son independientes, esto es,

$$\pi_{j|i} = \pi_{+j} \text{ con } j = 1, \ldots, J.$$

Si X e Y son variables respuesta entonces hablamos de *independencia*. Si Y es respuesta y X explicativa hablamos de *homogeneidad*.

En la tabla 15.3 recogemos una tabla cruzada con los conteos en donde consideramos las frecuencias absolutas.

Si dividimos los conteos por el total de la tabla estamos **estimando** la distribución conjunta $\pi_{ij} = P(X = i, Y = j)$, $\hat{\pi}_{ij}$. Lo tenemos en la tabla 15.4.

En la tabla 15.5 tenemos las proporciones por fila que estiman la probabilidad condicionada $P(Y = j|X = i)$.

En la tabla 15.6 tenemos las proporciones por columna. Estamos estimando $P(X = i|Y = j)$.

Si volvemos a revisar la tabla 15.5 tenemos los siguientes conceptos de uso habitual en Epidemiología.

Sensibilidad Proporción de enfermos correctamente diagnósticados.

$$\pi_{1|1} = P(Y = 1|X = 1).$$

Especificidad Proporción de no enfermos correctamente diagnósticados.

$$\pi_{2|2} = P(Y = 2|X = 2).$$

Tabla 15.5: Proporciones por fila.

	Test positivo	Test negativo	Total
Enfermo	n_{11}/n_{1+}	n_{12}/n_{1+}	1
No enfermo	n_{21}/n_{2+}	n_{22}/n_{2+}	1

Tabla 15.6: Proporciones por columna.

$\hat{\pi}_{j\|i}$	Test positivo	Test negativo
Enfermo	n_{11}/n_{+1}	n_{12}/n_{+2}
No enfermo	n_{21}/n_{+1}	n_{22}/n_{+2}
Total	1	1

	1	2
1	$\pi_{1\|1}$	$\pi_{2\|1}$
2	$\pi_{1\|2}$	$\pi_{2\|2}$

15.4 Comparación de dos proporciones

Muchos estudios se diseñan para comparar grupos basándonos en una respuesta binaria, Y. Con dos grupos tenemos una tabla de contingencia 2×2.

$$\begin{aligned} \pi_{1|i} &= \pi_i \\ \pi_{2|i} &= 1 - \pi_{1|i} = 1 - \pi_i \end{aligned}$$

Queremos comparar π_1 con π_2.

¿Cómo comparamos? Podemos estudiar la diferencia de las proporciones $\pi_1 - \pi_2$. O bien el riesgo relativo: $\frac{\pi_1}{\pi_2}$. O bien el cociente de odds (**odds ratio**)

$$\theta = \frac{\pi_1/(1-\pi_1)}{\pi_2/(1-\pi_2)}. \tag{15.5}$$

Si π es la probabilidad de éxito entonces los odds se definen como

$$\Omega = \frac{\pi}{1-\pi}.$$

Equivalentemente

$$\pi = \frac{\Omega}{\Omega+1}.$$

En una tabla 2×2 tenemos los odds en la fila i

$$\Omega_i = \frac{\pi_i}{1-\pi_i}.$$

El cociente de los odds de las dos filas será el **odds ratio**

$$\theta = \frac{\pi_1/(1-\pi_1)}{\pi_2/(1-\pi_2)}.$$

Se tiene fácilmente que

$$\theta = \frac{\pi_{11}\pi_{22}}{\pi_{12}\pi_{21}}.$$

Por ello también se le llama el **cociente de los productos cruzados**. Los siguientes puntos son de interés sobre el odds ratio.

	Éxito	**Fracaso**
Grupo 1	$\pi_{1\|1}$	$\pi_{2\|1}$
Grupo 2	$\pi_{1\|2}$	$\pi_{2\|2}$

	Éxito	**Fracaso**
Grupo 1	π_1	$1-\pi_1$
Grupo 2	π_2	$1-\pi_2$

- Puede ser cualquier valor positivo.
- $\theta = 1$ significa que no hay asociación entre X e Y.
- Valores de θ alejados de 1 indican una asociación mayor.
- Se suele trabajar con $\log\theta$ pues entonces el valor que tenemos es simétrico respecto a cero.
- El odds ratio no cambia cuando intercambiamos filas y columnas.

Ejemplo 15.3. *Leemos los datos de la tabla 2.1 en [3].*

```
x = c(104, 189)
n = c(11037, 11034)
prop.test(x, n)
```

```
        2-sample test for equality of proportions
        with continuity correction

data: x out of n
X-squared = 24.429, df = 1, p-value =
7.71e-07
alternative hypothesis: two.sided
95 percent confidence interval:
 -0.010814914 -0.004597134
sample estimates:
   prop 1 prop 2
0.00942285 0.01712887
```

```
asp.ataque = prop.test(x, n)
attributes(asp.ataque)
```

```
$names
[1] "statistic" "parameter" "p.value"
[4] "estimate" "null.value" "conf.int"
[7] "alternative" "method" "data.name"

$class
[1] "htest"
```

```
prop.test(x, n, alt = "less")
```

```
        2-sample test for equality of proportions
        with continuity correction

data: x out of n
X-squared = 24.429, df = 1, p-value =
3.855e-07
alternative hypothesis: less
95 percent confidence interval:
 -1.000000000 -0.005082393
sample estimates:
   prop 1 prop 2
0.00942285 0.01712887
```

```
asp.ataque$estimate
```

```
   prop 1 prop 2
0.00942285 0.01712887
```

La diferencia de proporciones viene dada por

```
asp.ataque$estimate[2] − asp.ataque$estimate[1]
```

```
    prop 2
0.007706024
```

El riesgo relativo lo calculamos con

```
asp.ataque$estimate[2]/asp.ataque$estimate[1]
```

```
  prop 2
1.817802
```

Finalmente el odds ratio sería

```
x[2] ∗ (n[1] − x[1])/(x[1] ∗ (n[2] − x[2]))
```

```
[1] 1.832054
```

15.5 Inferencia en tablas de contingencia

15.5.1 Intervalos de confianza para parámetros de asociación

Odds ratio

Consideremos el intervalo para los odds ratio. El estimador del odds ratio es

$$\hat{\theta} = \frac{n_{11}n_{22}}{n_{12}n_{21}}$$

El estimador puede ser 0, infinito o no estar definido (∞) dependiendo de los conteos. Por ello no existe ni la media ni la varianza de $\hat{\theta}$ ni de $\log\hat{\theta}$. Una posibilidad es trabajar con

$$\tilde{\theta} = \frac{(n_{11}+0.5)(n_{22}+0.5)}{(n_{12}+0.5)(n_{21}+0.5)}$$

y con $\log\tilde{\theta}$. Un estimación del error estándar de $\hat{\theta}$ es

$$\hat{\sigma}(\log\hat{\theta}) = \left(\frac{1}{n_{11}} + \frac{1}{n_{12}} + \frac{1}{n_{21}} + \frac{1}{n_{22}}\right)^{1/2}$$

El intervalo de confianza de Wald sería:

$$\log\hat{\theta} \pm z_{\alpha/2}\hat{\sigma}(\log\hat{\theta})$$

Tomando las exponenciales en los extremos tenemos el correspondiente intervalo para $\log\theta$. El test es algo conservador (la probabilidad de cubrimiento es algo mayor que el nivel nominal).

Ejemplo 15.4 (Aspirina e infarto)**.** *Estudio sueco sobre el uso de la aspirina y el infarto de miocardio.*

	Infarto de miocardio		
	Si	*No*	*Total*
Placebo	*28*	*656*	*684*
Aspirina	*18*	*658*	*656*

Veamos el intervalo de confianza para el odds ratio.

```
Drug <-c("Placebo", "Aspirin")
Infarction <-c("yes", "no")
table.3.1 <-expand.grid(drug = Drug, infarction = Infarction)
datos <-c(28, 18, 656, 658)
table.3.1 <-cbind(table.3.1, count = datos)
```

El código siguiente convierte el data frame en una matriz. La siguiente función calcula todos los intervalos de confianza.

```
## Gives two-sided Wald CI's for odds ratio, difference in proportions
##and relative risk.
## Table is a 2x2 table of counts with rows giving the treatment
    ↪ populations
## aff.response is a string like "c(1,1)" giving the cell of the
    ↪ beneficial
## response and the treatment category alpha is significance level
Wald.ci<-function(Table, aff.response, alpha=.05){
    pow<-function(x, a=-1) x^a
    z.alpha<-qnorm(1-alpha/2)
    if(is.character(aff.response))
        where<-eval(parse(text=aff.response))
    else where<-aff.response
    Next<-as.numeric(where==1) + 1
    ## OR
    odds.ratio<-
        Table[where[1],where[2]]*Table[Next[1],Next[2]]/
        (Table[where[1],Next[2]]*Table[Next[1],where[2]])
    se.OR<-sqrt(sum(pow(Table)))
    ci.OR<-exp(log(odds.ratio) + c(-1,1)*z.alpha*se.OR)
    ## difference of proportions
    p1<-Table[where[1],where[2]]/(n1<-Table[where[1],Next[2]] +
                                    Table[where[1],where[2]])
    p2<-Table[Next[1],where[2]]/(n2<-Table[Next[1],where[2]]+
                                    Table[Next[1],Next[2]])
    se.diff<-sqrt(p1*(1-p1)/n1 + p2*(1-p2)/n2)
    ci.diff<-(p1-p2) + c(-1,1)*z.alpha*se.diff
    ## relative risk
    RR<-p1/p2
    se.RR<-sqrt((1-p1)/(p1*n1) + (1-p2)/(p2*n2))
    ci.RR<-exp(log(RR) + c(-1,1)*z.alpha*se.RR)
    list(OR=list(odds.ratio=odds.ratio, CI=ci.OR),
         proportion.difference=list(diff=p1-p2,
                                    CI=ci.diff),
         relative.risk=list(relative.risk=RR,CI=ci.RR))
}
```

Utilizamos la función que acabamos de definir. Observemos que nos devuelve las estimaciones y los intervalos de confianza para los odds ratio, la diferencia de proporciones y el riesgo relativo.

```
Wald.ci(temp, c(1, 1))
```

Diferencia de proporciones

Veamos el intervalo de confianza para la diferencia de proporciones. Suponemos que tenemos muestras de binomiales independientes. En grupo i tenemos $Y_i \sim Bi(n_i, \pi_i)$. Tenemos

$$\hat{\pi}_i = Y_i/n_i$$

$$E(\hat{\pi}_1 - \hat{\pi}_2) = \pi_1 - \pi_2$$

y el error estándar es

$$\sigma(\hat{\pi}_1 - \hat{\pi}_2) = \left[\frac{\pi_1(1-\pi_1)}{n_1} + \frac{\pi_2(1-\pi_2)}{n_2}\right]^{1/2}$$

Estimamos sustituyendo π_i por $\hat{\pi}_i$. El intervalo de confianza de Wald sería:

$$\hat{\pi}_1 - \hat{\pi}_2 \pm z_{\alpha/2}\hat{\sigma}(\hat{\pi}_1 - \hat{\pi}_2)$$

Usualmente la probabilidad de cubrimiento es menor que el coeficiente de confianza nominal. Especialmente para valores de π_1 y π_2 próximos a 0 o 1.

Intervalo de confianza para el riesgo relativo

El riesgo relativo muestral viene dado por

$$r = \frac{\hat{\pi}_1}{\hat{\pi}_2}$$

Hay una convergencia a la normalidad más rápida trabajando en la escala logarítmica. El error estándar asintótico de $\log r$ es

$$\sigma(\log r) = \left(\frac{1-\pi_1}{n_1\pi_1} + \frac{1-\pi_2}{n_2\pi_2}\right)^{1/2}.$$

Es algo conservador (probabilidad de cubrimiento mayor que el nivel de confianza nominal). El intervalo de confianza de Wald para $\log \pi_1/\pi_2$ es

$$\log r \pm z_{\alpha/2}\hat{\sigma}$$

15.5.2 Contraste de independencia en tablas de doble entrada

Nos planteamos el contraste de:

$$\begin{aligned} H_0 : \pi_{ij} &= \pi_{i+}\pi_{+j} \ \forall\, i,j \\ H_0 : \pi_{ij} &\neq \pi_{i+}\pi_{+j} \text{ para algún } i,j \end{aligned}$$

Los tests que vamos a considerar se pueden aplicar tanto para muestreo multinomial (con $I \times J$ categorías) como para muestreo multinomial independiente (para las distintas filas). En el primer caso contrastamos independencia y en el segundo homogeneidad. Es el test clásico propuesto por K. Pearson. Bajo H_0,

$$En_{ij} = \mu_{ij} = n\pi_{i+}\pi_{+j}.$$

Los MLE son

$$\hat{\mu}_{ij} = n\hat{\pi}_{i+}\hat{\pi}_{+j} = n\frac{n_{i+}}{n}\frac{n_{+j}}{n} = \frac{n_{i+}n_{+j}}{n}$$

Se utiliza el estadístico

$$X^2 = \sum_i \sum_j \frac{(n_{ij} - \hat{\mu}_{ij})^2}{\hat{\mu}_{ij}}. \tag{15.6}$$

Bajo H_0,

$$X^2 \sim \chi^2((I-1)(J-1)). \tag{15.7}$$

El test score produce el estadístico X^2. El test del cociente de verosimilitud produce el test

$$G^2 = 2\sum_i \sum_j n_{ij} \log \frac{n_{ij}}{\hat{\mu}_{ij}}, \tag{15.8}$$

Tabla 15.7: Educación y sentimiento religioso.

	Creencias religiosas			
Educación	Fundamentalista	Moderado	Liberal	Total
Menos que secundaria	178	138	108	424
Secundaria	570	648	442	1660
Graduado	138	252	252	642
Total	886	1038	802	2726

con $\hat{\mu}_{ij} = \frac{n_{i+}n_{+j}}{n}$. Bajo H_0,

$$G^2 \sim \chi^2((I-1)(J-1)). \tag{15.9}$$

Se rechaza para valores grandes de X^2 o G^2. La convergencia a la distribución ji-cuadrado es más rápida para X^2 que para G^2. La aproximación para G^2 es pobre si $n/IJ < 5$. La aproximación para X^2 puede ser razonablemente buena si las frecuencias esperadas son mayores que 1 y la mayor parte son mayores que 5. Si nos los podemos usar hay que utilizar métodos para muestras pequeñas.

Ejemplo 15.5. *Consideremos la tabla de contingencia 15.7.*

Vamos a aplicar los tests X^2 y G^2.

```
religion.counts←c(178,138,108,570,648,442,138,252,252)
table.3.2=cbind(expand.grid(list(Religious.Beliefs=c("Fund", "Mod", "Lib"),
Highest.Degree=c("<HS","HS or JH", "Bachelor or Grad"))),count=religion.counts)
table.3.2.array=tapply(table.3.2$count,table.3.2[,1:2], sum)
(res=chisq.test(table.3.2.array))
```

```
        Pearson's Chi-squared test

data:  table.3.2.array
X-squared = 69.157, df = 4, p-value =
3.42e-14
```

Los conteos esperados los obtenemos con

```
res$expected
```

```
                Highest.Degree
Religious.Beliefs <HS HS or JH
            Fund 137.8078 539.5304
            Mod 161.4497 632.0910
            Lib 124.7425 488.3786
                Highest.Degree
Religious.Beliefs Bachelor or Grad
            Fund 208.6618
            Mod 244.4593
            Lib 188.8789
```

Los tests y frecuencias esperadas pueden obtenerse con el paquete vcd.

```
library(vcd)
assocstats(table.3.2.array)
```

```
                  X^2 df P(> X^2)
Likelihood Ratio 69.812 4 2.4869e-14
Pearson 69.157 4 3.4195e-14

Phi-Coefficient : NA
Contingency Coeff.: 0.157
Cramer's V        : 0.113
```

La función chisq.test lleva test de Montecarlo.

```
chisq.test(table.3.2.array, sim=T, B=2000)
```

```
        Pearson's Chi-squared test with simulated
         p-value (based on 2000 replicates)

data:  table.3.2.array
X-squared = 69.157, df = NA, p-value =
0.0004998
```

El test del cociente de verosimilitud G^2 se obtiene como

```
fit.glm = glm(count~Religious.Beliefs+Highest.Degree, data=table.3.2,
              family=poisson)
fit.glm$deviance
```

```
[1] 69.81162
```

```
temp←predict(fit.glm,type="response")
matrix(temp, nc=3, byrow=T, dimnames=list(c("<HS","HS or JH", "Bachelor or
Grad"),c("Fund", "Mod", "Lib")))
```

```
                    Fund Mod Lib
<HS 137.8078 161.4497 124.7425
HS or JH 539.5304 632.0910 488.3786
Bachelor or\nGrad 208.6618 244.4593 188.8789
```

15.5.3 Más allá del test ji-cuadrado

¿Acaba la vida con el p-valor? Ya sabemos que el test es significativo. Ya tenemos un p-valor maravillosamente pequeño. ¿Y ahora qué? Quizás analizar los residuos. Vamos a comparar las frecuencias observadas con las esperadas. Notemos que, para muestreo de Poisson,

$$\sigma(n_{ij} - \mu_{ij}) = \sqrt{\mu_{ij}}.$$

La desviación estándar de $n_{ij} - \hat{\mu}_{ij}$ es menor que $\sqrt{\mu_{ij}}$ pero todavía proporcional a este valor. Definimos el **residuo de Pearson** como

$$e_{ij} = \frac{n_{ij} - \hat{\mu}_{ij}}{\sqrt{\hat{\mu}_{ij}}}.$$

En particular tenemos que el estadístico X^2 de Pearson es igual a la suma de los cuadrados de los residuos de Pearson.

$$X^2 = \sum_i \sum_j e_{ij}^2.$$

Comparar estos residuos con los percentiles normales da una visión conservadora. Se definen los *residuos de Pearson estandarizados* como

$$\frac{n_{ij} - \hat{\mu}_{ij}}{\left[\hat{\mu}_{ij}(1 - \hat{\pi}_{i+})(1 - \hat{\pi}_{+j})\right]^{1/2}}$$

que sí tienen una distribución normal estándar. Podemos comparar los residuos de Pearson estandarizados con los percentiles de la normal. Valores superiores (en módulo) a 2 o 3 indican falta de ajuste.

Ejemplo 15.6. *Vamos a calcular los residuos de Pearson para el cruce entre educación y fundamentalismo religioso.*

```
religion.counts←c(178,138,108,570,648,442,138,252,252)
table.3.2=cbind(expand.grid(list(Religious.Beliefs=c("Fund", "Mod", "Lib"),
Highest.Degree=c("<HS","HS or JH", "Bachelor or Grad"))),
count=religion.counts)
table.3.2.array=tapply(table.3.2$count,table.3.2[,1:2], sum)
(res=chisq.test(table.3.2.array))
```

```
        Pearson's Chi-squared test

data:  table.3.2.array
X-squared = 69.157, df = 4, p-value =
3.42e-14
```

```
fit.glm = glm(count~Religious.Beliefs+Highest.Degree, data=table.3.2,
                family=poisson)
```

```
resid.pear = residuals(fit.glm, type = "pearson")
```

y los residuos de Pearson estandarizados.

```
ni←rowSums(table.3.2.array) # sumas por filas
nj←colSums(table.3.2.array) # sumas por columnas
n←sum(table.3.2.array) # tamaño muestral total
resid.pear.mat←matrix(resid.pear, nc=3, byrow=T,
                        dimnames=list(c("<HS","HS or JH",
"Bachelor or Grad"),c("Fund", "Mod", "Lib")))
n*resid.pear.mat/sqrt(outer(n−ni,n−nj,"*") ) ## residuos de Pearson
```

```
                    Fund Mod Lib
<HS 4.534918 -3.5921772 -2.086799
HS or JH 1.814033 1.2859224 -3.050211
Bachelor or Grad -6.336271 0.9180244 6.252547
```

```
                                        ## estandarizados
```

15.5.4 Test exacto de Fisher para tablas 2×2

Todos los procedimientos vistos hasta ahora se basan en distribuciones asintóticas. Si tenemos muestras grandes no hay problemas. ¿Y con muestras pequeñas? Rezar es una buena opción. Siempre lo es. La otra es un test exacto. Consideramos una tabla 2×2. La hipótesis nula es de independencia. Condicionamos a los totales marginales de fila y columna. Solamente nos queda libre un conteo (por ejemplo, n_{11}) y

$$p(t) = P(n_{11} = t) = \frac{\binom{n_{1+}}{t}\binom{n_{2+}}{n_{+1}-t}}{\binom{n}{n_{+1}}}$$

donde los valores posibles son

$$m_- \leq n_{11} \leq m_+$$

con $m_- = \max\{0, n_{1+} + n_{+1} - n\}$ y $m_+ = \min\{n_{1+}, n_{+1}\}$. Queremos contrastar independencia. En tablas 2×2 lo podemos formular como

$$H_0 : \theta = 1$$

Tabla 15.8

	Predicción primer servicio		
Primer servicio	Leche	Té	Total
Leche	3	1	4
Té	1	3	4
Total	4	4	

frente a (alternativa unilateral)

$$H_1 : \theta > 1$$

Para el test anterior, si t_0 es el valor observado de n_{11}, entonces el p-valor sería

$$P(n_{11} \geq t_0)$$

Ejemplo 15.7 (La señora Muriel Bristol tomando té)**.** *Sería más extrema la tabla con $n_{11} = 4$. El p-valor sería*

$$P(n_{11} = 3) + P(n_{11} = 4) = 0.243$$

Estamos ordenando las tablas de acuerdo con n_{11}. Podríamos ordenarlas según el odds ratio o la diferencia de las proporciones y obtenemos el mismo test. Esto no será cierto para test bilateral. La definición de p-valor depende de cómo ordenamos las tablas. Lo que suele ir programado en software es (si $p(t) = P(n_{11} = t)$)

$$p = P(p(n_{11}) \leq p(t_0))$$

Sumamos la probabilidad de todas aquellas tablas que son tan probables o menos que la tabla observada. Otra opción es

$$p = P\Big(|n_{11} - E(n_{11})| \geq |t_0 - E(n_{11})|\Big)$$

teniendo en cuenta que para la hipergeométrica

$$E(n_{11}) = n_{1+}n_{+1}/n$$

Este procedimiento equivale con

$$p = P(X^2 \geq X_0^2)$$

siendo X_0^2 el valor observado de X^2.

Ejemplo 15.8. *Vamos a aplicar el test exacto de Fisher para datos de Muriel Bristol.*

```
(test <-fisher.test(matrix(c(3, 1, 1, 3), byrow = T, ncol = 2)))
```

```
        Fisher's Exact Test for Count Data

data:  matrix(c(3, 1, 1, 3), byrow = T, ncol = 2)
p-value = 0.4857
alternative hypothesis: true odds ratio is not equal to 1
95 percent confidence interval:
   0.2117329 621.9337505
sample estimates:
odds ratio
  6.408309
```

Si queremos el p-valor unilateral.

```
test$p.value/2
```

```
[1] 0.2428571
```

Podemos contrastar la alternativa unilateral directamente.

```
(fisher.test(matrix(c(3, 1, 1, 3), byrow = T, ncol = 2),
             alternative = "greater"))
```

```
        Fisher's Exact Test for Count Data

data:  matrix(c(3, 1, 1, 3), byrow = T, ncol = 2)
p-value = 0.2429
alternative hypothesis: true odds ratio is greater than 1
95 percent confidence interval:
 0.3135693       Inf
sample estimates:
odds ratio
  6.408309
```

Podemos contrastar la hipótesis nula de que el odds ratio sea igual a otro valor no necesariamente igual a 1 (por ejemplo 0.2).

```
(fisher.test(matrix(c(3, 1, 1, 3), byrow = T, ncol = 2), or = 0.2))
```

```
        Fisher's Exact Test for Count Data

data:  matrix(c(3, 1, 1, 3), byrow = T, ncol = 2)
p-value = 0.02246
alternative hypothesis: true odds ratio is not equal to 0.2
95 percent confidence interval:
   0.2117329 621.9337505
sample estimates:
odds ratio
  6.408309
```

15.5.5 Distribución condicionada exacta

Consideramos tablas con $I \times J$. Suponemos muestreo multinomial independiente (por filas). En consecuencia los totales de fila (n_{i+}) están fijados. Asumimos la hipótesis de que la distribución condicionada de cada fila es la misma

$$H_0 : \pi_{j|1} = \ldots = \pi_{j|I} = \pi_{+j} \text{ para } j = 1, \ldots, J$$

y condicionamos ahora a los totales de columna. La distribución condicionada de los conteos es

$$\frac{\prod_i n_{i+}! \prod_j n_{+j}!}{n! \prod_i \prod_j n_{ij}!}$$

la *distribución hipergeométrica múltiple*. Si suponemos una única distribución multinomial. Condicionamos al total de la tabla n. Condicionamos a los totales de fila y columna. La independencia se formula como

$$H_0 : \pi_{ij} = \pi_{i+}\pi_{+j} \forall i, j$$

Obtenemos la misma distribución condicionada.

Test exacto de independencia para tablas $I \times J$. Hemos de establecer un orden entre las tablas. La opción más habitual es considerar la probabilidad de la tabla. El p-valor es la probabilidad de las tablas (con las marginales dadas) que no son más probables que la tabla observada. Otra posibilidad es utilizar un estadístico que mida la distancia de una tabla con la hipótesis nula: Como el estadístico X^2.

Ejemplo 15.9.

```
(table.3.9 <-matrix(c(25, 25, 12, 0, 1, 3), byrow = T, ncol = 3))
```

```
     [,1] [,2] [,3]
[1,] 25 25 12
[2,] 0 1 3
```

Ejercicios

* **Ex. 29 —** Demostrar que el estadístico score es el que aparece en la ecuación 15.2.

* **Ex. 30 —** Demostrar que el estadístico de Wald es el que aparece en la ecuación 15.3.

* **Ex. 31 —** Demostrar que el estadístico del cociente de verosimilitudes es el que aparece en la ecuación 15.4.

* **Ex. 32 —** Vamos a considerar los datos `vcdExtra::AirCrash` en los que se tiene información sobre accidentes de aviones comerciales entre 1993 y 2015. Se pide:

1. Leer los datos.
2. Con `base::table()` obtener la distribución de frecuencias absolutas de las variables que indican la fase en que se produjo el accidente y la causa del mismo: `Phase` y `Cause`.
3. Obtener con `base::table` la tabla de contingencia para las variables `Phase` y `Cause`.
4. Repetir el apartado 2 utilizando la tabla de contingencia de apartado 3 y las función `table::margin.table`.
5. Repetir los apartados 2 y 3 utilizando `base::xtabs`.
6. Obtener las proporciones totales, por fila y por columna utilizando la función `base::prop.table`.
7. Utilizando `gmodels::CrossTable` obtener la tabla en que aparezcan, en cada celda: los conteos, las proporciones sobre el total, las proporciones por fila y columna.
8. Constrastar la independencia entre las variables `Phase` y `Cause` con un test ji-cuadrado.

* **Ex. 33 —** Consideremos los datos `vcdExtra::Cancer`. Se refieren a cancer de pecho. Son datos (antiguos) de supervivencia a tres años. En este problema parece lógico considerar la posible asociación (marginal) entre las variables `Grade` y `Center` y la variable **respuesta** `Survival`. Se pide:

1. Leer los datos.

2. Constrastar la dependencia de `Survival` respecto de `Grade` en cada uno de los centros del estudio utilizando un test de Fisher.
3. Repetir el apartado 2 con un test ji-cuadrado.

* **Ex. 34 —** Consideramos los datos `vcdExtra::Titanicp`. Se pide:

1. ¿Existe asociación entre el sexo y si se sobrevivió o no?
2. ¿Qué sexo sobrevivió en mayor proporción?[135]

[135] Recordemos la película Titanic.

Capítulo 16

Componentes principales

16.1 Introducción

En este tema nos ocupamos de problemas de reducción de dimensión. ¿Qué significa reducir la dimensión? Responder a esta pregunta es obvio si nos fijamos en los datos que tenemos. Trabajando con expresión de genes tenemos tantas filas como genes y tantas columnas como muestras. En resumen miles de filas y decenas o centenares de columnas. En el tema En temas anteriores hemos visto como seleccionar filas, esto es, seleccionar genes es una tarea incluso previa. Hemos de quedarnos con genes que tengan una expresión diferencial si consideramos alguna característica fenotípica o bien con genes que tengan una expresión mínima o bien con genes que tengan un cierto nivel de variación. ¿Qué hacemos con las columnas? O de otro modo: ¿qué hacemos con las muestras? Quizás la respuesta natural sería: si tenemos miles de filas, ¿por qué preocuparse de unas decenas de filas? No es una buena respuesta. Realmente tener 50 o 100 columnas son muchas a la hora de visualizar resultados o bien de aplicar tratamientos estadísticos. En este tema tratamos el tema de cómo reducir el número de columnas o el número de filas.

16.2 Definición

En esta sección realizamos una presentación más formal del concepto de componente principal. Su nivel no corresponde con el resto del capítulo y se incluye como material complementario.

Cuando tomamos medidas sobre genes, personas, objetos, empresas, unidades experimentales de un modo genérico, se tiende a recoger el máximo de variables posible. En consecuencia tenemos dimensiones del vector de características grande.

Una opción consiste en sustituir la observación original, de dimensión p, por k combinaciones lineales de las mismas. Obviamente pretendemos que k sea mucho menor que p. El objetivo es elegir k de modo que expresen una proporción razonable de la dispersión o variación total cuantificada como la traza de la matriz de covarianza muestral, $tr(S)$,

Sea $\boldsymbol{X}$ un vector aleatorio de dimensión p con vector de medias $\boldsymbol{\mu}$ y matriz de covarianzas $\boldsymbol{\Sigma}$. Sea $T = (\boldsymbol{t}_1|\boldsymbol{t}_2|\ldots|\boldsymbol{t}_d)$ donde las t_i indican la i-ésima columna de la matriz ortogonal tal que

$$\boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T} = \boldsymbol{\Lambda} = diag(\lambda_1, \ldots, \lambda_d), \tag{16.1}$$

donde $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d \geq 0$ son los valores propios de la matriz definida positiva Σ. Esto se puede hacer como consecuencia del teorema de descomposición espectral.

$$\boldsymbol{Y} = \boldsymbol{T}^T(\boldsymbol{X} - \boldsymbol{\mu}). \tag{16.2}$$

Si denotamos la j-ésima componente de Y como Y_j entonces $Y_j = \boldsymbol{t}_j^T(\boldsymbol{X} - \boldsymbol{\mu})$ con $j = 1, \ldots, d$. A la variable Y_j la llamamos la j-ésima **componente principal** de Y. La variable $Z_j = \frac{Y_j}{\sqrt{\lambda_j}}$ es la j-ésima **componente principal estandarizada** de Y.

Estas componentes tienen algunas propiedades de gran interés. Notemos que el vector $\boldsymbol{t}_j$ tiene longitud unitaria y, por lo tanto, Y_j no es más que la proyección ortogonal de $\boldsymbol{X} - \boldsymbol{\mu}$ en la dirección t_j.

Proposición 16.1. 1. *Las variables Y_j son incorreladas y $var(Y_j) = \lambda_j$.*

2. *Las variables Z_j son incorreladas y con varianza unitaria.*

Prueba. En cuanto al apartado primero tenemos que

$$var(\boldsymbol{Y}) = var(\boldsymbol{T}^T(\boldsymbol{X} - \boldsymbol{\mu}) = \boldsymbol{T}^T var(\boldsymbol{X})\boldsymbol{T} = \boldsymbol{T}^T\boldsymbol{\Sigma}\boldsymbol{T} = \boldsymbol{\Lambda}.$$

El segundo apartado es directo a partir del primero. □

Se verifica el siguiente resultado.

Teorema 16.1. *Las componentes principales $Y_j = \boldsymbol{t}_j^T(\boldsymbol{X} - \boldsymbol{\mu})$ con $j = 1, \ldots, p$ tienen las siguientes propiedades:*

1. *Para cualquier vector $\boldsymbol{a}_1$ de longitud unitaria, $var(\boldsymbol{a}_1^T\boldsymbol{X})$ alcanza su valor máximo, λ_1, cuando $\boldsymbol{a}_1 = \boldsymbol{t}_1$.*

2. *Para cualquier vector $\boldsymbol{a}_j$ de longitud unitaria tal que $\boldsymbol{a}_j^T\boldsymbol{t}_i = 0$ para $i = 1, \ldots, j-1$, se tiene que $var(\boldsymbol{a}_j^T X)$ toma su valor máximo λ_j cuando $\boldsymbol{a}_j = \boldsymbol{t}_j$.*

3. $\sum_{j=1}^p var(Y_j) = \sum_{j=1}^p var(X_j) = traza(\boldsymbol{\Sigma})$.

La versión muestral de las componentes principales la obtenemos sustituyendo en lo anterior $\boldsymbol{\mu}$ y $\boldsymbol{\Sigma}$ por $\bar{\boldsymbol{X}}$ y $\hat{\boldsymbol{\Sigma}}$ respectivamente. Es importante considerar el estimador de $\boldsymbol{\Sigma}$ que estamos utilizando (o bien el estimador insesgado donde dividimos por $n-1$ o bien el estimador en donde dividimos por n).

Si denotamos por $\hat{\lambda}_1 \geq \ldots \geq \hat{\lambda}_p$ los valores propios ordenados de $\hat{\Sigma}$ y la matriz $\hat{T} = (\hat{t}_1, \ldots, \hat{t}_d)$ es la matriz tal que cada columna es el correspondiente vector propio entonces tenemos las componentes principales muestrales dadas por $y_j = \hat{\boldsymbol{T}}^T(\boldsymbol{x}_i - \bar{\boldsymbol{x}})$. La nueva matriz de datos viene dada por

$$\boldsymbol{y}^T = (y_1, \ldots, y_n) = \hat{\boldsymbol{T}}^T(\boldsymbol{x}_1 - \bar{\boldsymbol{x}}, \ldots, \boldsymbol{x}_n - \bar{\boldsymbol{x}}) \tag{16.3}$$

Finalmente, si las variables vienen dadas en unidades muy distintas puede ser conveniente sustituir la matriz de covarianzas (poblacional o muestral) por la correspondiente matriz de correlaciones. De hecho, una de los inconvenientes de las componentes principales como un modo de reducir la dimensión de los datos es precisamente que obtenemos resultados distintos si utilizamos las componentes principales obtenidas a partir de

la matriz de covarianzas o bien las componentes principales obtenidas a partir de la matriz de correlaciones.

A partir de las p variables originales podemos obtener hasta p componentes principales. Sin embargo, hemos dicho que pretendemos reducir la dimensión del vector de datos. La pregunta a responder es: ¿con cuántas componentes nos quedamos?

Supongamos que estamos trabajando con la matriz de covarianzas $\mathbf{\Sigma}$. Hemos de recordar que $var(y_j) = \lambda_j$ y que $\sum_{j=1}^{p} var(x_j) = \sum_{j=1}^{p} var(y_j) = \sum_{j=1}^{p} \lambda_j$. En consecuencia se suelen considerar los siguientes cocientes

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j}, \text{ con } k = 1, \ldots, p,$$

de modo que, cuando para un cierto valor de k, estamos próximos a la unidad nos quedamos con ese valor de k. En la versión muestral trabajaremos o bien con los valores propios de la matriz de covarianzas muestral o la matriz de correlaciones muestrales.

Una referencia muy interesante sobre componentes principales es [1].

16.3 Componentes principales

Para ilustrar los conceptos vamos a considerar unos datos sencillos. Tomamos los datos *golub* y nos fijamos en los genes que tienen que ver con "Cyclin" (tienen esta palabra en su nombre). Vamos a considerar las dos primeras muestras, esto es, las dos primeras columnas.

```
data(golub,package="multtest")
sel = grep("Cyclin",golub.gnames[,2])
golub.red = golub[sel,1:2]
```

Los datos aparecen en la figura 16.1.

```
pacman::p_load(ggplot2)
df = data.frame(golub.red)
names(df) = c("sample1","sample2")
p = ggplot(df,aes(x=sample1,y=sample2)) + geom_point()
ggsave(paste0(dirTamiFigures,"PCA2.png"),p)
```

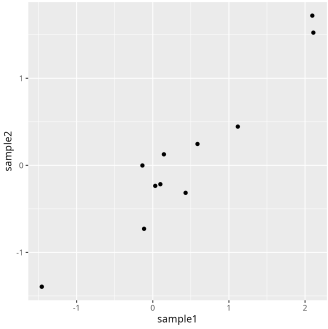


Figura 16.1: Expresión de las dos primeras muestras de los datos golub para aquellos genes que contienen la expresión Cyclin.

Cada punto corresponde con uno de los genes seleccionados.

Para la fila i (para el gen i) denotamos las expresiones observadas en las dos muestras como $x_i = (x_{i1}, x_{i1})$. Tenemos n filas y por lo tanto nuestros datos son x_i con $i = 1, \ldots, n$.

Centramos los datos. Esto es, le restamos a cada columna la media de la columna. Para ello, primero calculamos las medias. El vector de medias lo vamos a denotar por $\bar{x} = (\bar{x}_1, \bar{x}_2)$ donde

$$\bar{x}_j = \sum_{i=1}^{n} \frac{x_{ij}}{n}$$

es decir, cada componente es la media de las componentes. En resumen el primer valor es la expresión media en la primera muestra para todos los genes. Podemos calcular fácilmente el vector de medias. Una función específica es la siguiente.

```
medias = colMeans(golub.red)
```

Un código equivalente al anterior con `apply` es

```
medias = apply(golub.red,2,mean)
```

Le restamos a cada columna su media.

```
golub.red = sweep(golub.red,2,medias)
```

En la figura 16.2 reproducimos los datos centrados y mostramos los ejes de coordenadas en rojo.

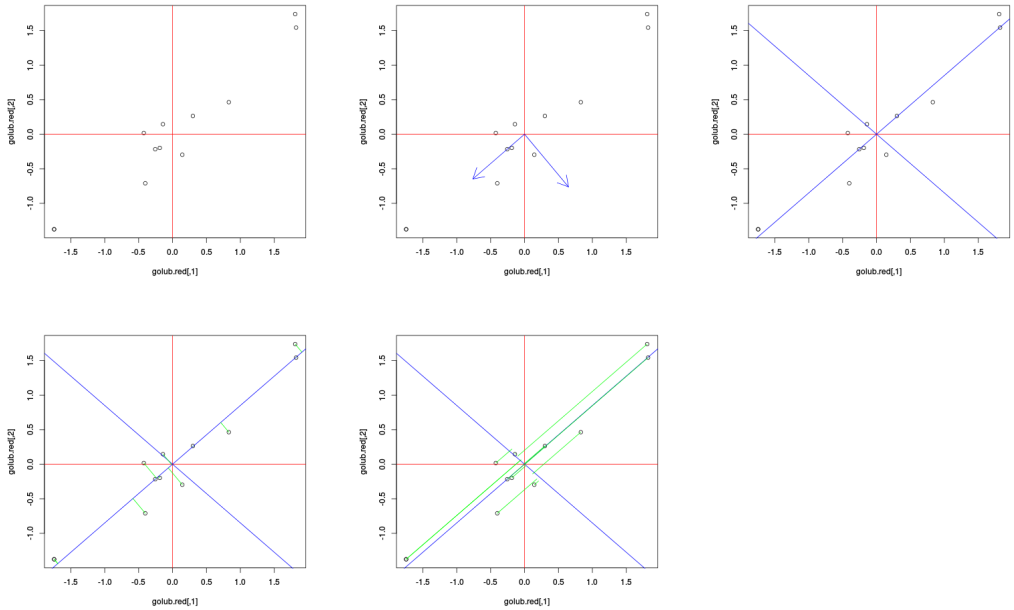


Figura 16.2: a) Centramos los datos golub.red y mostramos el nuevo sistema de coordenadas en rojo. b) Vectores que definen las líneas donde proyectamos. c) Las líneas sobre las que proyectamos. d) Datos golub.red mostrando las líneas sobre las que proyectamos (en azul) para obtener las dos componentes principales y las proyecciones sobre la *primera* componente en verde. e) Datos golub.red mostrando las líneas sobre las que proyectamos (en azul) para obtener las dos componentes principales y las proyecciones sobre la *segunda* componente.

Hemos trasladado los datos de modo que las medias de cada variable valen cero ahora. Esto es lo que se conoce como centrar los datos. Hemos *centrado los datos*. Podemos comprobar que los nuevos datos tienen una media nula.

```
colMeans(golub.red)
```

```
[1] -3.006854e-17 1.069746e-17
```

Nuestros datos (filas) corresponden a las expresiones correspondientes a los genes. Los datos originales tienen dimensión 2 (dos variables correspondientes a las dos muestras) y supongamos que pretendemos reducir la dimensión a solo una, esto es, representar cada gen mediante un único número. La idea de las componentes principales es considerar una combinación lineal de los valores originales. Es decir, se pretende elegir un vector (de dimensión dos) $a_1 = (a_{11}, a_{12})$ de modo que en lugar de utilizar x_i consideremos (el resumen) $u_i = a_{11}x_{i1} + a_{12}x_{i2}$. ¿Qué a_1 elegimos? La idea es lograr que los valores u_i tengan la mayor variabilidad que se pueda con objeto de no perder información. Mantener la

variabilidad original indica que mantenemos la información que los datos originales tienen. En concreto se elige a_1 de modo que maximizamos

$$\frac{1}{n}\sum_{i=1}^{n}(u_i - \bar{u})^2.$$

El vector a_1 nos indica la dirección sobre la cual proyectamos los datos originales. Las proyecciones sobre a_1, los valores u_i son la mejor descripción univariante de los datos. La segunda mejor descripción que sea ortogonal a la anterior serían las proyecciones sobre la línea ortogonal a la primera que pasa por el origen de coordenadas. Obtengamos las componentes principales.

```
a.pca = prcomp(golub.red)
```

Los vectores directores de las líneas sobre las que proyectamos aparecen en la figura 16.2. Estos vectores son

```
a.pca$rotation
```

Las líneas sobre las que proyectamos aparecen en azul en la figura 16.2 Hemos trasladado los datos de modo que las medias de cada variable valen cero ahora. Esto es lo que se conoce como centrar los datos. Hemos *centrado los datos*. Podemos comprobar que los nuevos datos tienen una media nula.

```
colMeans(golub.red)
```

Nuestros datos (filas) corresponden a las expresiones correspondientes a los genes. Los datos originales tienen dimensión 2 (dos variables correspondientes a las dos muestras) y supongamos que pretendemos reducir la dimensión a solo una, esto es, representar cada gen mediante un único número. La idea de las componentes principales es considerar una combinación lineal de los valores originales. Es decir, se pretende elegir un vector (de dimensión dos) $a_1 = (a_{11}, a_{12})$ de modo que en lugar de utilizar x_i consideremos (el resumen) $u_i = a_{11}x_{i1} + a_{12}x_{i2}$. ¿Qué a_1 elegimos? La idea es lograr que los valores u_i tengan la mayor variabilidad que se pueda con objeto de no perder información. Mantener la variabilidad original indica que mantenemos la información que los datos originales tienen. En concreto se elige a_1 de modo que maximizamos

$$\frac{1}{n}\sum_{i=1}^{n}(u_i - \bar{u})^2.$$

El vector a_1 nos indica la dirección sobre la cual proyectamos los datos originales. Las proyecciones sobre a_1, los valores u_i son la mejor descripción univariante de los datos. La segunda mejor descripción que sea ortogonal a la anterior serían las proyecciones sobre la línea ortogonal a la primera que pasa por el origen de coordenadas. Obtengamos las componentes principales.

```
a.pca = prcomp(golub.red)
```

Los vectores directores de las líneas sobre las que proyectamos aparecen en la figura 16.2(b). Estos vectores son

```
a.pca$rotation
```

Las líneas sobre las que proyectamos aparecen en azul en la figura 16.2(c). Y finalmente podemos ver las proyecciones. En verde mostramos las proyecciones sobre la primera componente (figura 16.2(d)). Y ahora consideremos la proyección sobre la segunda componente. En la figura 16.2(e) la mostramos. Los valores de estas proyecciones los obtenemos con

```
predict(a.pca)
```

Las desviaciones estándar de la primera y segunda componente principal son las siguientes

```
a.pca$sdev
```

Y las varianzas son los cuadrados de las desviaciones estándar.

```
a.pca$sdev^2
```

¿Cómo de variables son nuestros datos? Podemos cuantificar el total de la variación de los datos sumando las varianzas de cada una de las dos coordenadas

```
var(golub.red[,1])
```

```
[1] 1.266931
```

```
var(golub.red[,2])
```

```
[1] 0.9245807
```

cuya suma es

```
var(golub.red[,1]) + var(golub.red[,2])
```

```
[1] 2.191512
```

Las nuevas coordenadas tienen la misma varianza total.

```
sum(a.pca$sdev^2)
```

```
[1] 2.191512
```

¿Y qué proporción de la varianza es atribuible a la primera componente? ¿Y a la segunda? Podemos dividir la varianza de cada componente por la suma total.

```
variacion.total = sum(a.pca$sdev^2)
a.pca$sdev^2 / variacion.total
```

```
[1] 0.98439023 0.01560977
```

La primera componente explica un 98.44 % de la variación total. ¿Para qué necesitamos utilizar dos números por gen si con uno tenemos esencialmente la misma información.

16.4 Componentes principales de los datos golub

Hemos visto las componentes principales con dos variables (en nuestro caso dos muestras) con efecto de poder ver el significado geométrico de las componentes principales. Vamos a trabajar con el banco de datos completo: todos los datos golub que tienen 38 muestras (27 de un tipo de leucemia y 11 de otro tipo).

Obtengamos las componentes principales.

```
golub.pca = prcomp(golub,scale=FALSE,center=TRUE)
```

El argumento `center=TRUE` centra los datos restando la media de la columna de modo que las variables tengan medias nulas. El argumento `scale=TRUE` hace que las variables originales sean divididas por su desviación estándar de modo que la varianza (y la desviación estándar) de las nuevas variables sea la unidad.

Diferentes criterios podemos aplicar a la hora de decidir con cuántas componentes nos quedamos.

1. Uno puede ser la proporción total explicada. Fijar un nivel mínimo y quedarnos con el número de componentes necesario para superar este valor mínimo.

2. El segundo puede ser que una componente no puede tener una desviación estándar menor que una de las variables originales. Si hemos escalado cada variable original dividiendo por su desviación estándar entonces la desviación estándar de cada componente ha de ser mayor que uno.

3. Otro criterio puede ser ver en qué momento se produce un descenso de la desviación estándar muy notable. Quedarnos con las componentes previas.

Un resumen de las componentes nos puede indicar con cuántas nos quedamos.

```
summary(golub.pca)
```

En este caso podemos aplicar el criterio 1 o 3 ya que no hemos dividido por la desviación estándar.

Si fijamos, por ejemplo un 80% de la variación total a explicar entonces debemos quedarnos con las cinco primeras componentes.

Atendiendo al tercer criterio quizás (muy subjetivo) tampoco sea una mala solución quedarnos con las cinco primeras. Puede ser una buena elección y una solución intermedia. Los nuevos datos los obtenemos con la función *predict*.

```
a = predict(golub.pca)
```

Podemos ver todas las componentes para el primer gen (primera fila).

```
a[1,]
```

Y ahora nos quedamos con las primeras cinco columnas correspondientes con las cinco primeras componentes principales como hemos decidido previamente.

```
a = a[,1:5]
```

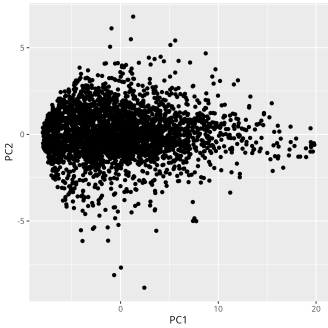


Figura 16.3: Dos primeras componentes principales de los datos golub.

Podemos representar, como es habitual, las dos primeras componentes. Lo tenemos en la figura 16.3

```
df = data.frame(PC1 = a[,1],PC2=a[,2])
p = ggplot(df,aes(x=PC1,y=PC2)) + geom_point()
ggsave(paste0(dirTamiFigures,"PCA28.png"),p)
```

Es interesante observar los valores del vector asociado a la primera componente.

```
golub.pca$rotation[,1]
```

Podemos ver que son coeficientes muy parecidos, todos positivos. Básicamente tenemos la media muestral de todos los niveles de expresión en las 38 muestras. La primera componente es básicamente la media sobre las 38 muestras. ¿Y la segunda componente?

```
golub.pca$rotation[,2]
```

Si observamos los coeficientes vemos que las primeros 27 valores son positivos y los 11 últimos son negativos. Además no hay una gran diferencia entre los 27 primeros y tampoco entre los 11 últimos. Básicamente (aunque no exactamente) estamos comparando, para cada gen, la media de los niveles de expresión sobre los datos ALL (leucemia linfoblástica aguda) con la media sobre los datos AML (leucemia mieloide aguda).

16.5 ¿Muestras o genes?

En la sección anterior hemos realizado un análisis de componentes principales *de las muestras*, esto es, la reducción de dimensión suponía resumir, para gen, su perfil de expresión con menos valores. Sin embargo, también podemos realizar un análisis de componentes principales de los genes. En este caso, las observaciones son las muestras y las variables serían las expresiones sobre los genes. Obviamente podemos realizarlo sobre el conjunto de todos los genes o bien después de aplicar alguna selección previa (bien un filtrado no específico, bien utilizando información sobre los propios genes).

Ejemplo 16.1 (Datos golub). *Pretendemos hacer un análisis de componentes principales donde las observaciones son las muestras y para cada muestra tenemos su perfil de expresión sobre todos los genes. Para ello hemos de considerar la matriz traspuesta de la matriz de expresión original con la función* `t()`. *A los datos que obtenemos le aplicamos la función* `prcomp`.

```
tgolub.pca = prcomp(t(golub),scale=FALSE,center=TRUE)
```

Veamos el resumen del análisis que acabamos de realizar.

```
summary(tgolub.pca)
```

En principio, la dimensión de mis datos es de 3051 que es el número de genes con los que estamos trabajando. Sin embargo, vemos que solamente considera 38 posibles componentes principales. ¿Por qué? Pues porque es el rango de la matriz de covarianzas.[1]

Es interesante representar las componentes principales de las muestras y los genes. En las figuras 16.4 y 16.5 mostramos las dos primeras componentes utilizando la función `plotPCA` *del paquete [89, affycoretools].*

[1] Es un comentario incomprensible siguiendo el material anterior. Lo indico simplemente.

```
library(affycoretools)
png(paste0(dirTamiFigures,"PCA34−a.png"))
affycoretools::plotPCA(golub,legend = FALSE)
dev.off()
png(paste0(dirTamiFigures,"PCA34−b.png"))
affycoretools::plotPCA(t(golub),legend = FALSE)
dev.off()
```

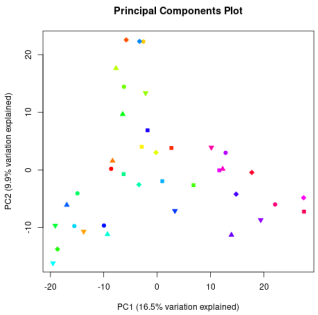


Figura 16.4: Dos primeras componentes principales de los datos golub utilizando como variables las expresiones en los genes.

Cuando realizamos un análisis de los genes sabemos que las observaciones están clasificadas (las muestras son de dos tipos de leucemia). La función `plotPCA` *está preparada para mostrarnos si las componentes nos reproducen los grupos que sabemos que previamente tenemos (aquí no hemos realizado ningún análisis cluster previo). En la figura 16.6 mostramos la misma figura 16.4 pero diferenciando el tipo de muestra (según el tipo de leucemia).*

```
tipo = factor(golub.cl+1,levels = 1:2,
                labels = c("ALL","AML"))
png(paste0(dirTamiFigures,"PCA37.png"))
affycoretools::plotPCA(golub,groups = tipo,
                        groupnames = tipo,legend=FALSE)
dev.off()
```

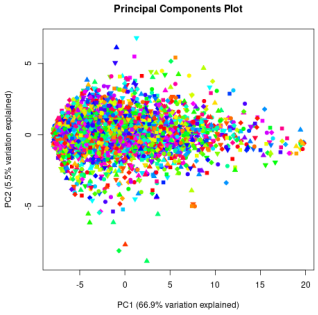


Figura 16.5: Dos primeras componentes principales de los datos golub utilizando como variables las expresiones en las muestras.

Podemos ver cómo la primera componente principal diferencia bastante claramente los dos tipos de leucemia.

16.6 ¿Tipificamos los datos?

Hemos trabajado hasta ahora con los datos centrados (restamos la media muestral a cada variable) pero no hemos tipificado, esto es, no hemos dividido cada variable por su desviación estándar. Si los datos tienen una escala similar esta es la opción razonable. Para matrices de expresión esto es así, estamos midiendo lo mismo en distintas muestras y genes y, sobre el papel, la escala es la misma. Pero: ¿y si no es así? En este caso en lugar de trabajar con las variables originales las tipificamos y realizamos el análisis anterior con estos datos.[136]

Reproducimos (y no mostramos) el análisis de los datos golub pero utilizando datos tipificados. Notemos que la opción fundamental es `scale = TRUE`.

```
golub.pca = prcomp(golub,scale=TRUE,center=TRUE)
summary(golub.pca)
tgolub.pca = prcomp(t(golub),scale=TRUE,center=TRUE)
summary(tgolub.pca)
```

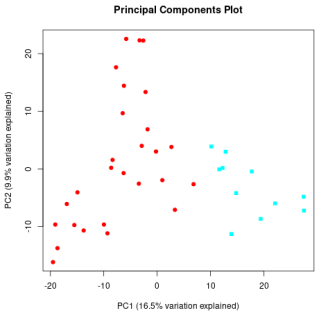


Figura 16.6: Dos primeras componentes principales de los datos golub. donde se aplican las componentes a los genes y diferenciamos la muestra con un color y tipo de punto distintos.

16.7 Ejemplos

Ejemplo 16.2 (GSE20986). *Cargamos los datos.*

```
pacman::p_load(Biobase)
data(gse20986,package="tamidata")
```

La covariable fenotípica que nos indica el tejido de donde se obtuvo la muestra es

```
pData(gse20986)[,"tissue"]
```

En la figura 16.7 vemos representadas las componentes principales de los genes.

[136] Realmente si no tipificamos y simplementamos centramos los datos estamos realizando un análisis de componentes principales sobre la matriz de covarianzas. Si tipificamos las variables entonces estamos realizando un análisis de componentes principales sobre la matriz de correlaciones.

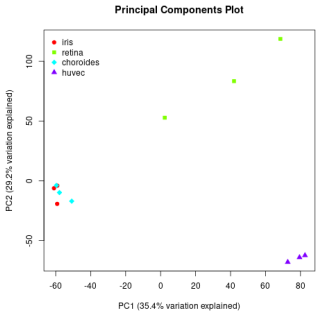


Figura 16.7: Datos GSE20986. Componentes principales de los genes. Se aprecia como el grupo huvec (tejido de vena umbilical) se diferencia claramente de los demás.

```
png(paste0(dirTamiFigures,"PCA44.png"))
plotPCA(gse20986,
        groups = pData(gse20986)[,"tissue"],
        groupnames=c("iris","retina","choroides","huvec"))
dev.off()
```

Llama la atención la clara diferenciación de las tres muestras correspondientes a huvec. Aparecen dispersas pero separadas de los demás las correspondientes a la retina. Finalmente las muestras de iris y coroides no muestran una diferenciación.

Vemos que son informativas pero: ¿qué proporción de la variación total explican? Realizamos unas componentes principales de las muestras. Notemos que tomamos la matriz de expresión y luego aplicamos la traspuesta ya que de lo contrario las componentes se estarían realizando sobre las muestras.

```
a.pca = prcomp(t(exprs(gse20986)))
summary(a.pca)
```

Apreciamos que las dos primeras componentes nos explican un 64.61% de la variación total.

En la representación gráfica de la figura 16.7 hemos utilizado la función affycoretools::plotPCA. Quizás no es buena costumbre ligarnos tanto a un tipo de dato como el ExpressionSet. Podemos utilizar [70, ggfortify]. Como antes, hemos de trabajar con la matriz transpuesta y ponemos como una variable adicional de nuestro `data.frame` *la variable que indica el tejido.*

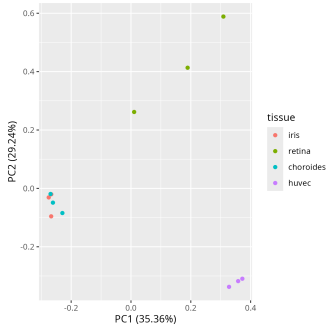


Figura 16.8: Componentes sobre los genes con [70].

```
df0 = t(exprs(gse20986))
tissue = pData(gse20986)[,"tissue"]
df = data.frame(tissue,df0)
pacman::p_load(ggfortify)
p = autoplot(prcomp(df[,-1]),data=df,colour="tissue")
ggsave(paste0(dirTamiFigures,"PCA46b.png"),p)
```

Vemos las componentes en la figura 16.8

Ejemplo 16.3 (GSE1397)**.** *Leemos los datos.*

```
data(gse1397,package="tamidata")
```

Tenemos los metadatos fenotípicos con

```
pData(gse1397)
```

Vemos que la variable que nos indica la alteración y el tejido son

```
pData(gse1397)[,'type']
```

y

```
pData(gse1397)[,'tissue']
```

En la figura 16.9 tenemos el resultado de un análisis de componentes principales considerando los genes como observaciones y diferenciando según el tipo de alteración.

```
png(paste0(dirTamiFigures,"PCA51.png"))
affycoretools::plotPCA(gse1397, groups = pData(gse1397)[,'type'],
          groupnames = levels(pData(gse1397)[,'type']))
dev.off()
```

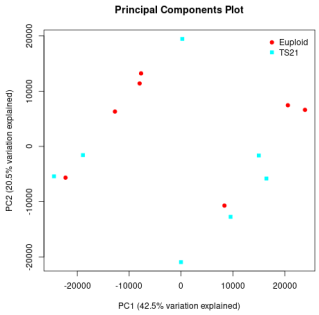


Figura 16.9: Dos primeras componentes principales de datos GSE1397 diferenciando el diagnóstico.

Las dos primeras componentes vemos como no nos diferencian el tipo de alteración. Sin embargo, cuando consideramos el tejido de donde obtuvimos la muestra se aprecia una mejor diferenciación. De hecho la primera componente diferencia bastante bien los tejidos. Lo podemos ver en la figura 16.10.

```
library(affycoretools)
png(paste0(dirTamiFigures,"PCA53.png"))
affycoretools::plotPCA(gse1397, groups = pData(gse1397)[,'tissue'],
        groupnames = levels(pData(gse1397)[,'tissue']))
dev.off()
```

.

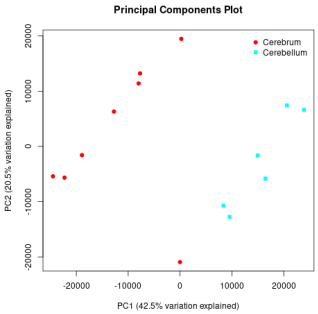


Figura 16.10: Dos primeras componentes principales de datos GSE1397 diferenciando el tejido.

Capítulo 17

Análisis cluster

17.1 Introducción

Este tema está dedicado al análisis cluster y su aplicación a datos de expresión de gen. Trataremos lo que en la literatura estadística recibe el nombre de *análisis cluster*[1] o, en mejor castellano, *análisis de conglomerados*. En la literatura de Inteligencia Artificial se utiliza la expresión *clasificación no supervisada*.[2]

¿Qué vamos a clasificar cuando trabajamos con datos de expresión de gen? Nuestra información es la matriz de expresión. Podemos considerar como observaciones a clasificar los genes. En este caso, la observación son los niveles de expresión observados en todas las muestras, el perfil de expresión del gen sobre las muestras analizadas.

También podemos considerar como observaciones a clasificar las muestras. En este caso cada muestra viene descrita por sus expresiones sobre todos los genes. Podemos trabajar con todos los genes o con un subconjunto de genes.

En uno u otro caso tenemos una muestra de observaciones multivariantes de dimensión d. ¿Qué pretendemos hacer con ellos? Encontrar grupos. Muy breve la respuesta pero: ¿qué son grupos? Imaginemos una imagen aérea fija de un patio de un colegio. En esta imagen los datos son las posiciones de los niños. ¿Se agrupan los niños formando grupos o todos están jugando con todos y los grupos son una consecuencia pasajera del juego (delanteros y defensas que aparecen agrupados en un ataque)?

Parece claro y simple el problema. Sí, lo parece. ¿Qué es un grupo? ¿Cómo defino un grupo? ¿Cuántos grupos distingo en los datos? Estamos viendo el efecto del ruido o realmente hay una estructura debajo que la vemos en un entorno con ruido.

¿Qué quiere decir encontrar grupos? Se trata de clasificar las observaciones en grupos de modo que las observaciones de un mismo grupo sean lo más similares que podamos y que los grupos entre sí sean muy distintos. El número de procedimientos que se han propuesto en la literatura es muy grande. La mayor parte de ellos no tienen un modelo probabilístico debajo, no son procedimientos *basados en modelo*. Son métodos que esencialmente utilizan el concepto de *proximidad*. Valoran de distintas formas lo próximos, lo cercanos que están los puntos, dentro de un mismo grupo y entre distintos grupos. Es pues, el primer punto a tra-

[1] Por cierto que la palabra cluster no existe en castellano

[2] Por tradición, utilizaremos la expresión de análisis cluster. Es la que, personalmente, prefiero.

tar: ¿cómo cuantificamos lo cerca o lejos que están los distintos puntos? En § 17.3 nos ocupamos de este punto. También será necesario, como veremos en el tema, valorar cuando dos conjuntos de puntos son más o menos parecidos, proximos, similares. En la misma sección nos ocupamos de ello. Supongamos que ya hemos clasificado en distintos grupos. ¿Hemos tenido éxito al hacerla? Cuando tenemos un análisis discriminante tenemos una muestra donde sabemos a qué grupo pertenece el individuo y dónde lo hemos clasificado. Esto nos permitía valorar si nuestro procedimiento clasifica bien o no. Aquí no vamos a tener esta referencia que nos da la muestra de entrenamiento. ¿Cómo valorarlo? Un concepto conocido por silueta y debido a Rousseeuw [78] nos va a servir para ello. No es ni tan simple ni tan satisfactorio como en análisis discriminante (como es de esperar si tenemos menos información para trabajar). Lo estudiamos en § 17.8.

Entre los muchos procedimientos de obtener los grupos a partir de los datos, los más utilizados son dos tipos: procedimientos jerárquicos y métodos de particionamiento. De los jerárquicos nos ocupamos en § 17.5. El método de las k-medias y el método de las k-mediodes (el castellano como siempre es muy sufrido pues no existe la palabra) son métodos de particionamiento y los tratamos en § 17.7.

Un texto antiguo pero de un interés enorme en este tema es [130].

17.2 Datos

En esta sección presentamos tres ejemplos sencillos de observaciones a agrupar y que utizamos como ilustración simple de los métodos que proponemos.

17.2.1 Un ejemplo artificial

Son unos datos muy conocidos, los datos Ruspini. Están en el paquete *cluster* [92].[3]

Cargamos el paquete y los datos.

```
pacman::p_load(cluster)
data(ruspini)
```

En la figura 17.1 mostramos estos datos.

```
pacman::p_load(ggplot2)
p = ggplot(ruspini,aes(x=x,y=y))+geom_point()
```

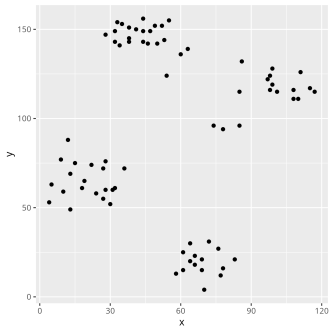


Figura 17.1: Datos ruspini

Son datos bivariantes. Visualmente vemos cómo se agrupan los puntos. Parece (muy) claro que podemos distinguir cuatro grupos.

17.2.2 Un ejemplo con muestras

Un ejemplo con los datos golub. Empezamos cargando los datos.

```
data(golub,package="multtest")
```

Previamente hemos visto que los valores de expresión de los genes "CCND3 Cyclin D3" y "Zyxin" permiten diferenciar entre ALL y AML. Localicemos las expresiones correspondientes a estos genes.

[3] En este tema el paquete cluster es, sin duda, el fundamental.

```
grep("CCND3 Cyclin D3",golub.gnames[,2])
```

```
[1] 1042
```

```
grep("Zyxin",golub.gnames[,2])
```

```
[1] 2124
```

Los datos aparecen en estas filas. Por lo tanto podemos construir la matriz de datos correspondiente.

```
cz.data = data.frame(golub[1042,],golub[2124,])
names(cz.data) = c("CCND3_Cyclin_D3","Zyxin")
```

Este será un segundo ejemplo para analizar. Los datos aparecen en la figura 17.2.

```
p = ggplot(cz.data,aes(x=CCND3_Cyclin_D3,y=Zyxin))+geom_point()
```

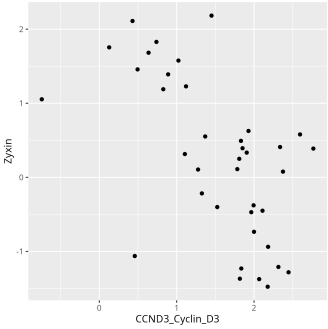


Figura 17.2: Datos cz.data

En este caso las observaciones corresponden a las muestras y las variables son los niveles de expresión de dos genes. ¿Hay grupos? Esto no son datos artificiales como los de Ruspini y ya no es tan claro.

17.2.3 Un ejemplo con genes

Consideremos (otra vez) los datos `golub`.

```
data(golub,package="multtest")
```

Para cada gen vamos a considerar la expresión media sobre los casos ALL.

```
x = apply(golub[,golub.cl == 0],1,mean)
```

y la media sobre los casos AML.

```
y = apply(golub[,golub.cl == 1],1,mean)
```

Y vamos a limitarnos a un conjunto de genes. En concreto vamos a considerar los genes que se relacionan con el término biológico "Cyclin". Determinamos qué filas ocupan.

```
sel = grep("Cyclin",golub.gnames[,2])
```

La función **`grep`** busca la palabra "Cyclin" en cada uno de los nombres que le pasamos. Y los nombres de estos genes son

```
golub.gnames[sel,2]
```

```
 [1] "CCND2 Cyclin D2"
 [2] "CDK2 Cyclin-dependent kinase 2"
 [3] "CCND3 Cyclin D3"
 [4] "CDKN1A Cyclin-dependent kinase inhibitor 1A (p21, Cip1)"
 [5] "CCNH Cyclin H"
 [6] "Cyclin-dependent kinase 4 (CDK4) gene"
 [7] "Cyclin G2 mRNA"
 [8] "Cyclin A1 mRNA"
 [9] "Cyclin-selective ubiquitin carrier protein mRNA"
[10] "CDK6 Cyclin-dependent kinase 6"
[11] "Cyclin G1 mRNA"
[12] "CCNF Cyclin F"
```

Ahora nos limitamos a considerar las expresiones medias para los casos ALL y los casos AML para estos genes.

```
cyclin.data = data.frame(x[sel],y[sel])
names(cyclin.data) = c("meanALL","meanAML")
```

En la figura 17.3 mostramos los datos.

```
p = ggplot(cyclin.data,aes(x=meanALL,y=meanAML))+geom_point()
```

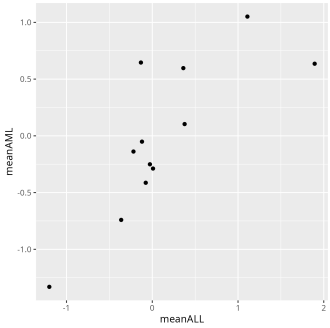


Figura 17.3: Datos cyclin.data.

No parece, al menos visualmente, que tengamos muchos grupos. De eso se trata. De saber si hay grupos y encontrarlos.

17.3 Disimilaridades

En lo que sigue denotaremos los datos a agrupar con x_i con $i = 1, \ldots n$ siendo $x_i \in \mathbb{R}^d$. Todos los ejemplos que hemos visto en la sección anterior eran (con objeto de poder representarlos) datos bidimensionales, $d = 2$.

17.3.1 Distancias euclídea y de Manhattan

Empezamos tratando el problema de cuantificar el grado de proximidad, de similaridad entre dos puntos en el espacio de dimensión d. Tradicionalmente este tema en Matemáticas se ha formalizado a través del concepto de *distancia* o *métrica*. Una métrica es una función que a cada par de puntos $x, y \in \mathbb{R}^d$ le asocia un valor positivo de modo que cuando mayor es más *distantes* son, más alejados están. Como siempre la formalización matemática de un concepto intuitivo ha de ser prudente y pedir que se verifiquen ciertos axiomas que resulten razonables y generalmente admisibles. Las distancias más utilizadas en análisis cluster son la distancia euclídea y la distancia de Manhattan. Para dos vectores x e y (en $\mathbb{R}^d$) entonces la distancia euclídea se define como

$$d_E(x, y) = \sqrt{\sum_{k=1}^{d} (x_k - y_k)^2}, \tag{17.1}$$

con $x, y \in \mathbb{R}^d$. La distancia de Manhattan viene dada por

$$d_M(x, y) = \sum_{k=1}^{d} |x_k - y_k|. \tag{17.2}$$

Las distancias euclídea y de Manhattan son adecuadas cuando trabajamos con variables continuas y que además estén en una misma escala. Son dos ejemplos de distancias o métricas.[4] En lo que sigue en lugar de distancia utilizaremos el término *medida de disimilaridad* indicando que no necesariamente ha de ser distancia. Es una visión menos restrictiva a la hora de definir que dos genes están cerca.

[4] En concreto la función d definida en el espacio producto $\mathbb{R}^d \times \mathbb{R}^d$ se dice que es una métrica si verifica:

No negativa $d(x, y) \geq 0$.

Un punto dista 0 de sí mismo $d(x, x) = 0$.

Simetría $d(x, y) = d(y, x)$.

Desigualdad triangular $d(x, z) \leq d(x, y) + d(y, z)$, para todo $x, y, z \in \mathbb{R}^d$.

En lo que sigue agrupamos bien genes bien muestras. Si esta clasificación no supervisada la basamos en una distancia lo que buscamos es encontrar grupos de genes que tienen perfiles de expresión similares. En definitiva que se comportan de un modo parecido en el experimento para todas las muestras. Si agrupamos muestras lo que buscamos son muestras que, para los genes considerados, tienen unas expresiones similares. Por ejemplo, si las muestras corresponden a un cierto tipo de cáncer entonces los grupos que podemos encontrar serían subtipos de cáncer. En resumen genes o muestras los consideramos similares cuando sus expresiones lo son. Además interpretamos la similaridad como una métrica. El concepto de distancia o métrica (del cual hemos visto dos ejemplos) es limitado en muchas aplicaciones. En ocasiones podemos interpretar que dos genes están *próximos* cuando sus expresiones están *relacionadas* de alguna forma, en resumen, cuando están *corregulados*. En este caso podemos usar para definir la disimilaridad a partir de una cuantificación de la dependencia entre los genes.

17.3.2 Disimilaridades utilizando coeficientes de correlación

Consideremos dos genes con perfiles $x = (x_1, \ldots, x_n)$ e $y = (y_1, \ldots, y_n)$. En primer lugar pretendemos cuantificar el grado de dependencia entre los pares (x_i, y_i) para $i = 1, \ldots, n$. Si nos limitamos a la posible dependencia lineal entre estos pares entonces el coeficiente de correlación de Pearson es la medida genéricamente aceptada. Se define como

$$r_{x,y} = \frac{\sum_{i=1}^{n}(x_i - \bar{x}_n)(y_i - \bar{y}_n)}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x}_n)^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y}_n)^2}}.$$

Lo fundamental es que varía entre -1 y 1. Cuando toma valores próximos a cero indica independencia entre los pares y cuando toma valor próximo a 1 o a -1 indica una dependencia lineal extrema. De hecho si vale 1 entonces existen constantes a y b tales que $y_1 = ax_i + b$ con $a > 0$. Si vale -1 entonces existen constantes a y b tales que $y_1 = ax_i + b$ con $a < 0$.

Supongamos que cuando evaluamos si dos genes están *próximos* o están *alejados* lo que estamos intentando expresar es que están corregulados. Esto indicaría que una mayor expresión de uno en una muestra se corresponde con una mayor expresión en la misma muestra. Si, además, la asociación es de tipo lineal esto significaría que el perfil de expresión de gen se podría obtener como una función lineal de la expresión del otro gen. En resumen su coeficiente de correlación de Pearson debiera de ser próximo a uno. También sería razonable entender como próximos aquellos genes donde cuando uno tiene un valor alto de expresión el otro lo tiene bajo. Es decir, finalmente tienen una fuerte relación lineal pero de tipo inverso.

A partir del coeficiente de correlación podemos definir distintas medidas de disimilaridad. Algunas opciones que se han propuesto son las siguientes:

1. $d(x, y) = 1 - r_{x,y}$,
2. $d(x, y) = (1 - r_{x,y})/2$,
3. $d(x, y) = 1 - |r_{x,y}|$,
4. $d(x, y) = \sqrt{1 - r_{x,y}^2}$.

[5] De un modo similar podemos definir disimilaridades sustituyendo el coeficiente de correlación de Pearson con el coeficiente de correlación de Spearman.

17.3.3 Matriz de disimilaridades

Tenemos una serie de conjunto de puntos a agrupar: $\{x_1, \ldots, x_n\}$. Una vez elegida una disimilaridad construiremos una matriz que en la fila i, columna j tiene la disimilaridad entre el punto x_i y el punto x_j. Denotaremos indistintamente $d(x_i, x_j)$ o d_{ij}. Por tanto, la matriz de disimilaridades es

$$D = [d(x_i, x_j)]_{i,j=1,\ldots,n} = [d_{ij}]_{i,j=1,\ldots,n}.$$

¿Cómo calculamos las matrices de disimilaridades que acabamos de proponer? La función *dist* de [117] es una buena opción cuando utilizamos las distancias euclídea o de Manhattan.

Un detalle es importante. Si la disimilaridad que estamos considerando es simétrica, esto es, si la disimilaridad d verifica

$$d(x_i, x_j) = d(x_j, x_i),$$

entonces la matriz de disimilaridades

$$D = [d(x_i, x_j)]_{i,j=1,\ldots,n}$$

es simétrica y estamos almacenando información redundante. Además hemos de considerar que

$$d(x_i, x_i) = 0,$$

es decir, la disimilaridad de un perfil de expresión consigo mismo es nula y por lo tanto, en la matriz de disimilaridades, la diagonal de la matriz es nula. Sabemos que trabajamos habitualmente con un número de genes muy grande. La matriz de disimilaridades total sería de tamaño $N \times N$ y por ello, es más que conveniente eliminar redundancias. Veamos cómo esto lo hace de un modo automático la función `dist`.

Vamos a calcular la matriz de distancias con la función *dist* para los datos *cz.data* propuestos en § 17.2.2.

```
cz.dist.euclidea = dist(cz.data,method="euclidian")
```

¿De qué clase es `cz.dist.euclidea`.

```
class(cz.dist.euclidea)
```

```
[1] "dist"
```

La distancia de Manhattan la podemos obtener con

```
cz.dist.manhattan = dist(cz.data,method="manhattan")
```

```
data(golub,package="multtest")
golub.cor1 = 1-cor(t(golub))
golub.cor2 = (1-cor(t(golub)))/2
golub.cor3 = 1-abs(cor(t(golub)))
golub.cor4 = sqrt(1-cor(t(golub))^2)
```

[5] En http://research.stowers-institute.org/efg/R/Visualization/cor-cluster/ podemos encontrar un breve estudio comparativo del funcionamiento de estas disimilaridades en clustering jerárquico.

En este caso `cz.dist` sí es una matriz completa (con todas sus filas y columnas). La vamos a transformar a un objeto de clase `dist`.

```
golub.cor1 = as.dist(golub.cor1)
golub.cor2 = as.dist(golub.cor2)
golub.cor3 = as.dist(golub.cor3)
golub.cor4 = as.dist(golub.cor4)
```

Con cada una de estas matrices podemos cuantificar la disimilaridad que hay entre los elementos originales de la muestra. Algunos de los procedimientos de agrupamiento que vamos a considerar en lo que sigue no necesitan conocer los datos originales. Pueden aplicarse con solo conocer esta matriz de disimilaridades. Otros no. Otros utilizan los datos a lo largo de las distintas etapas de aplicación del procedimiento.

17.4 Disimilaridades entre grupos de observaciones

En algunos procedimientos de agrupamiento (en particular, los jerárquicos) vamos a necesitar calcular disimilaridades entre conjuntos disjuntos de las observaciones originales. Estas disimilaridades las podemos calcular a partir de las disimilaridades originales entre puntos. Supongamos que tenemos un banco de datos con n individuos cuyos índices son $\{1, \ldots, n\}$. Sean A y B dos subconjuntos disjuntos del conjunto de índices de la muestra $\{1, \ldots, n\}$, esto es, dos subconjuntos de observaciones disjuntos. ¿Cómo podemos definir una disimilaridad entre A y B partiendo de las disimilaridades entre los datos individuales? Se han propuesto muchos procedimientos. Si denotamos la disimilaridad entre A y B como $d(A, B)$ entonces las disimilaridades más habitualmente utilizadas son las siguientes:

Enlace simple La disimilaridad entre los dos grupos es el mínimo de las disimilaridades entre las observaciones de uno y de otro. Tomamos la disimilaridad de los objetos que más se parecen en uno y otro grupo.

$$d(A, B) = \min_{a \in A, b \in B} d(a, b)$$

Enlace completo Ahora tomamos como disimilaridad entre los grupos como el máximo de las disimilaridades, en definitiva, la disimilaridad entre los objetos más alejados o más distintos.

$$d(A, B) = \max_{a \in A, b \in B} d(a, b)$$

Promedio La disimilaridad es el promedio de las disimilaridades entre todos los posibles pares.

$$d(A, B) = \frac{1}{|A| \times |B|} \sum_{a \in A, b \in B} d(a, b)$$

donde $|A|$ es el cardinal del conjunto A.

Es importante notar que solamente necesitamos conocer las disimilaridades entre los individuos para poder calcular las disimilaridades entre grupos de individuos.

En la siguiente sección nos vamos a ocupar de los métodos jerárquicos en los cuales es fundamental el procedimiento que elijamos para calcular distintas entre grupos.

17.5 Cluster jerárquico

La idea de estos procedimientos es construir una jerarquía de particiones del conjunto de índices.

Sea $\{1, \ldots, n\}$ el conjunto de índices que indexan las distintas observaciones. Supongamos que $\{C_1, \ldots, C_r\}$ es una partición de este conjunto de índices:

- $C_i \subset \{1, \ldots, n\}$; son disjuntos dos a dos, $C_i \cap C_j = \emptyset$ si $i \neq j$ con $i, j = 1, \ldots, n$ y
- $\cup_{i=}^{r} C_i = \{1, \ldots, n\}$.

Dada una partición del conjunto de índices podemos calcular la matriz $r \times r$ que en la posición (i, j) tiene la disimilaridad entre el conjunto C_i y C_j, $d(C_i, C_j)$ según alguno de los procedimientos antes indicados.

Veamos los procedimientos jerárquicos aglomerativos. En estos procedimientos vamos a iniciar el agrupamiento con la partición: $C_i = \{i\}$ con $i = 1, \ldots, n$, es decir, cada grupo es un individuo. En cada iteración vamos agrupando el par de conjuntos (elementos de la partición que tengamos en esa iteración) que estén *más próximos* según la disimilaridad entre grupos que estemos utilizando. El proceso continúa hasta que tengamos un único grupo.

Un esquema algorítmico del procedimiento indicado puede ser el siguiente:

Paso 0 Tenemos grupos unitarios formados por cada una de las observaciones. Tenemos pues una partición inicial $C_i = \{i\}$ con $i = 1, \ldots, n$. En un principio, cada dato es un grupo.

Paso 1 Calculamos las disimilaridades entre los elementos de la partición. Para ello utilizamos cualquiera de los procedimientos antes indicados.

Paso 2 Agrupamos los dos conjuntos de la partición más próximos y dejamos los demás conjuntos igual. Tenemos ahora C_i con $i = 1, \ldots, k$.

Paso 3 Si tenemos un solo conjunto en la partición paramos el procedimiento.

Paso 4 Volvemos al paso 1.

Hay una representación gráfica muy utilizada para describir los resultados de un cluster jerárquico aglomerativo como el que acabamos de describir. Esta representación tiene el nombre de **dendograma**. En el dendograma se va mostrando a qué valor de la medida de disimilaridad se produce la unión de los grupos y simultáneamente qué grupos se están uniendo para esa disimilaridad. También nos permite una valoración rápida de cuántos grupos puede haber en el banco de datos. Simplemente trazando una linea horizontal a la altura en que tengamos el número de grupos que *pensamos* que puede haber.

Ejemplo 17.1 (ruspini)**.** *Consideremos los datos ruspini. Apliquemos un cluster jerárquico aglomerativo utilizando como disimilaridad entre grupos el promedio de las disimilaridades y como medida de disimilaridad la distancia euclídea.*

```
pacman::p_load(cluster)
ruspini.ag = agnes(ruspini,metric = "euclidean",method="average")
```

Una opción alternativa con base::hclust al código anterior es

```
dd = dist(ruspini,method="euclidian")
ruspini.ag = hclust(dd,method="average")
```

Representamos el dendograma en la figura 17.4.

```
p = ggdendro::ggdendrogram(ruspini.ag)
ggsave(paste0(dirTamiFigures,"Cluster23.png"),p)
```

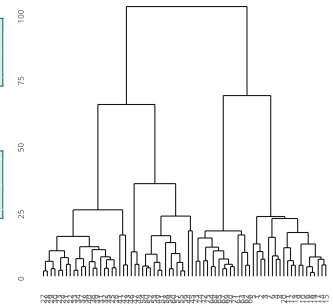


Figura 17.4: Dendograma para un cluster jerárquico aplicado a los datos ruspini con métrica euclídea y el promedio para la disimilaridad entre grupos.

Supongamos que decidimos quedarnos con cuatro grupos. Las clasificaciones de los datos son las siguientes donde la etiqueta que se asigna a cada grupo es arbitraria.

```
cutree(ruspini.ag,4)
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
 1  1  1  1  2  2  2  2  2  2  2  2  2  2  2  2
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 2  2  2  2  2  2  2  2  2  2  2  3  3  3  3  3
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 3  3  3  3  3  3  3  3  3  3  3  3  4  4  4  4
65 66 67 68 69 70 71 72 73 74 75
 4  4  4  4  4  4  4  4  4  4  4
```

Y ahora podemos representar los datos de modo que ponemos en un mismo color los datos que hemos clasificado en un grupo. Lo mostramos en la figura 17.5.

```
p = ggplot(ruspini,aes(x=x,y=y,color=cutree(ruspini.ag,4)))
p = p + geom_point(shape=cutree(ruspini.ag,4))
```

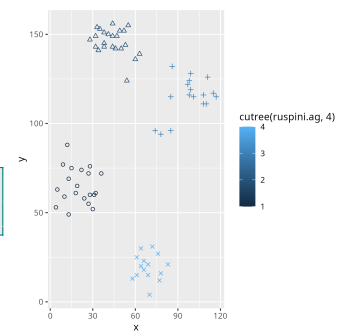


Figura 17.5: Clasificación obtenida para los datos ruspini utilizando un cluster jerárquico con distancia euclídea, promedio entre grupos y cuatro grupos.

Ejemplo 17.2 (cz.data)**.** *Apliquemos un cluster jerárquico aglomerativo utilizando como disimilaridad entre grupos el enlace simple y como distancia la de Manhattan.*

```
##cz.ag = agnes(cz.data,metric = "euclidean",method="single")
dd = dist(cz.data,method="euclidian")
cz.ag = hclust(dd,method="single")
```

La figura 17.6 muestra el dendograma que obtenemos.

```
pacman::p_load(ggdendro)
p = ggdendro::ggdendrogram(cz.ag , rotate = FALSE, size = 2)
```

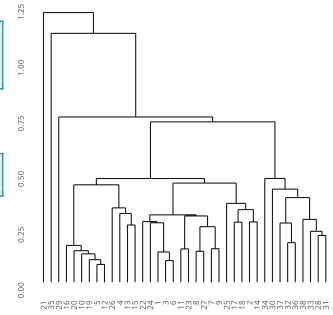


Figura 17.6: Dendograma para datos cz.data utilizando distancia euclídea y enlace simple.

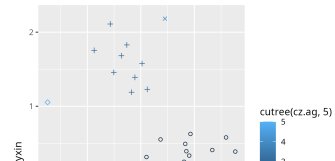
Supongamos que nos quedamos cinco grupos. Las clasificaciones son:

```
cutree(cz.ag,5)
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 1 1
[24] 1 1 1 1 3 4 3 3 3 3 3 5 3 3 3
```

En la figura 17.7 tenemos los resultados de la clasificación.

```
p = ggplot(cz.data,aes(x=CCND3_Cyclin_D3,y=Zyxin,color=cutree(cz.ag,5)))
p = p + geom_point(shape=cutree(cz.ag,5))
```



17.6 Distancia cofenética

Una vez hemos construido un clustering jerárquico podemos asociarle una distancia conocida como *distancia cofenética*. Dadas dos observaciones (en nuestro caso dos perfiles de expresión) consideramos el momento en que ambas observaciones han sido agrupadas en un mismo grupo. La distancia cofenética entre estas observaciones es la distancia entre los dos grupos a los que previamente pertenecían estas observaciones (una en cada uno de ellos) en el momento en que se unen. Se calcula con la función `cophenetic` del paquete [117, stats].

Ejemplo 17.3 (Distancia cofenética con cyclin.ag)**.** *Continuamos con el jerárquico aglomerativo aplicado a los datos cyclin.data. Repetimos el análisis cluster calculando previamente la matriz de disimilaridades. Empezamos calculando la matriz de distancias.*

```
cyclin.manhattan = daisy(cyclin.data,metric = "manhattan")
```

Y ahora el cluster jerárquico (observemos que no le damos los datos sino que le pasamos directamente la matriz de disimilaridades).

```
##cyclin.ag = agnes(cyclin.manhattan,diss = TRUE)
cyclin.ag = hclust(cyclin.manhattan,method="average")
```

Calculamos la distancia cofenética asociada al cluster jerárquico que acabamos de realizar.

```
cyclin.cofe = cophenetic(cyclin.ag)
```

Y determinamos el coeficiente de correlación de Pearson.

```
cor(cyclin.manhattan,cyclin.cofe)
```

```
[1] 0.8501727
```

El coeficiente de correlación observado 0.8501727 es un valor grande. Tenemos pues una buena descripción del modo en que los datos se agrupan.

Las distancias cofenéticas tienen sus problemas. Es una manera *natural* de medir disimilaridad entre los datos a partir del modo en que estos van agrupándose. Sin embargo, notemos que muchos pares de observaciones tienen la misma disimilaridad. Esto es lógico, cuando unimos dos grupos grandes la distancia cofenética entre cada observación de un grupo y del otro grupo tienen la misma distancia cofenética. Se suele calcular el coeficiente de correlación entre la matriz de disimilaridad original y la distancia cofenética. Si el valor es grande indica que el dendograma es una buena descripción de los datos.

Un estudio mucho más amplio de lo visto en esta sección puede verse en [130, capítulo 5].

17.7 Métodos de particionamiento

Suponemos ahora que tenemos una idea de cuántos grupos hay. Posiblemente hemos realizado un análisis jerárquico previo con todos los datos o, si eran muchos, con una selección aleatoria de los datos. Tenemos pues una idea de cuántos grupos tendremos que considerar. Obviamente podemos luego evaluar los resultados modificando el número de grupos. En principio, vamos a suponer que fijamos el número de grupos a considerar. Suponemos pues que *sabemos* el número de grupos y lo denotamos por k.

17.7.1 Método de las k-medias

El primer procedimiento que vamos a ver es el método de las *k-medias* (que por alguna extraña razón en la literatura de Inteligencia Artificial se le llama de las *c-medias* lo que demuestra que cada persona copia a sus amigos o, simplemente, conocidos). Supongamos que tenemos $C_1, \ldots, C_k$ una partición de $\{1, \ldots, n\}$. Un modo bastante natural de valorar la calidad del agrupamiento que la partición nos indica sería simplemente considerar la siguiente función.

$$\sum_{i=1}^{k} \sum_{j \in C_i} d_E(x_j, \bar{x}_{C_i})^2, \tag{17.3}$$

donde d_E denota aquí la distancia euclídea y

$$\bar{x}_{C_i} = \frac{1}{|C_i|} \sum_{j \in C_i} x_j, \tag{17.4}$$

es el vector de medias del grupo cuyos índices están en C_i. Una partición será tanto mejor cuanto menor sea el valor de la función dada en 17.3. El procedimiento de agrupamiento de las k-medias simplemente se basa en elegir como partición de los datos aquella que nos da el *mínimo* de la función objetivo considerada en ecuación 17.3. Notemos que en muchos textos se hablan del algoritmo de las k-medias y se identifica con un procedimiento concreto para encontrar el mínimo de la función. Aquí entendemos el procedimiento como la minimización de la función objetivo. De hecho, *R* ofrece hasta cuatro posibles procedimientos de los muchos que cabe proponer. Hay que diferenciar claramente el procedimiento del método de aplicación del mismo, del método de obtención de dicho mínimo.

Es importante darnos cuenta de que el procedimiento que acabamos de ver está basado en la utilización de la distancia euclídea y en que, dado un grupo, podemos calcular el vector de medias y esto solo lo podemos hacer si todas las variables son cuantitativas.

Vamos a aplicar el método de las k-medias en los tres ejemplos utilizando el número de grupos que hemos utilizado previamente.

Ejemplo 17.4 (ruspini). *Aplicamos el k-medias.*

```
ruspini.km = kmeans(ruspini,4)
```

La clasificación viene dada por

```
ruspini.km$cluster
```

```
 1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16
 1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32
 1  1  1  1  3  3  3  3  3  3  3  3  3  3  3  3
33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48
 3  3  3  3  3  3  3  3  3  3  4  4  4  4  4  4
49 50 51 52 53 54 55 56 57 58 59 60 61 62 63 64
 4  4  4  4  4  4  4  4  4  4  4  4  2  2  2  2
65 66 67 68 69 70 71 72 73 74 75
 2  2  2  2  2  2  2  2  2  2  2
```

La clasificación obtenida se corresponde con la que vemos en la figura 17.5.

Ejemplo 17.5 (Método de las k-medias y datos cz.data). *Empezamos con el k-medias.*

```
cz.km = kmeans(cz.data,2)
```

Las clasificaciones son

```
cz.km$cluster
```

```
 [1] 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
[24] 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
```

Y representamos los resultados.

```
pacman::p_load(factoextra)
png(paste0(dirTamiFigures,"Cluster50.png"))
fviz_cluster(cz.km,cz.data, ellipse.type = "norm")
dev.off()
```

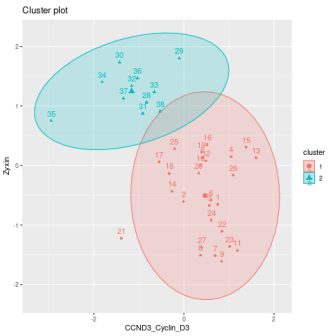


Figura 17.8: Clasificación obtenida utilizando el método de las k-medias aplicada a los datos cz.data.

Supongamos que probamos con tres grupos.

```
cz.km = kmeans(cz.data,3)
png(paste0(dirTamiFigures,"Cluster52.png"))
fviz_cluster(cz.km,cz.data, ellipse.type = "norm")
dev.off()
```

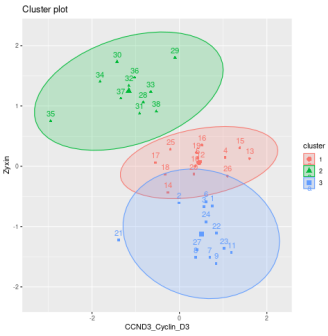


Figura 17.9: Clasificación obtenida utilizando el método de las k-medias aplicada a los datos cz.data con tres grupos.

17.7.2 Particionamiento alrededor de los mediodes

¿Y si no podemos calcular el vector de medias? ¿Y si no tiene sentido calcular el vector de medias? ¿Cómo promediar dos configuraciones de puntos distintas? ¿Cómo promediamos dos formas distintas descritas numéricamente? Cuando el concepto de promedio aritmético no tiene sentido podemos generalizar el procedimiento anterior y hablar de *k-mediodes*.

La idea ahora es sustituir esos centros calculados como vectores de medias de los individuos de un mismo grupo por individuos bien centrados, por individuos típicos, que sustituyan a las medias.

Supongamos que tomamos k individuos de la muestra que denotamos por m_i con $i = 1, \ldots, k$. Particionamos la muestra en k grupos de modo que el grupo C_i está formado por los individuos más próximos a m_i que a cualquier otro m_j con $j \neq i$,

$$C_i = \{l : d(x_l, x_i) = \min_{j \neq i} d(x_l, x_j)\}.$$

Consideremos la siguiente cantidad:

$$\sum_{i=1}^{k} \sum_{j \in C_i} d(x_j, x_{m_i}). \tag{17.5}$$

En el método de particionamiento alrededor de los mediodes nos planteamos encontrar las observaciones $m_1, \ldots, m_k$ que minimizan el valor dado en 17.5.

Ejemplo 17.6 (ruspini). *Aplicamos PAM.*

```
ruspini.pam = pam(ruspini,4)
```

La clasificación obtenida corresponde con la que mostramos en la figura 17.5. Con los datos ruspini cualquiera de los métodos que hemos utilizado cuando decidimos quedarnos con 4 grupos nos da los mismos resultados de clasificacion.

Ejemplo 17.7 (cz.data). *Empezamos con dos grupos.*

```
cz.pam = pam(cz.data,2)
```

Y representamos los resultados de la clasificación en la figura 17.10

```
png(paste0(dirTamiFigures,"Cluster58.png"))
fviz_cluster(cz.pam,cz.data, ellipse.type = "norm")
dev.off()
```

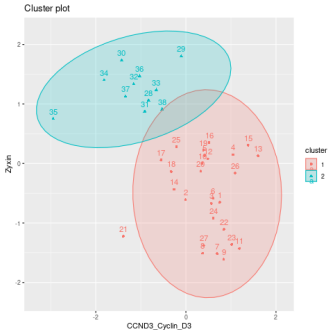


Figura 17.10: Clasificación de los datos cz.data con método PAM y dos grupos.

17.8 Silueta

Veamos cómo se construye la silueta. Para la observación i y el grupo C consideramos

$$\bar{d}(i, C) = \frac{1}{|C|} \sum_{j \in C} d(x_i, x_j),$$

la disimilaridad media i con los elementos del grupo C. Para cada observación i, sea A el cluster al cual lo ha asignado el procedimiento cluster que empleamos y calculamos $a(i)$ la disimilaridad media de i con todos los demás individuos del grupo A, $a(i) = \bar{d}(i, A)$. Obviamente estamos asumiendo que A contiene al menos otro objeto. Consideremos $\bar{d}(i, C)$ para todos los grupos $C \neq A$ y seleccionemos el que tiene el mínimo valor:

$$b(i) = \min_{C \neq A} \bar{d}(i, C).$$

El grupo B donde se alcanza este mínimo, es decir, $\bar{d}(i, B) = b(i)$ se le llama vecino del objeto i. [6] Definimos $s(i)$ como

$$\begin{aligned} s(i) &= 1 - \frac{a(i)}{b(i)} \text{ si } a(i) < b(i), && (17.6)\\ &= 0 \text{ si } a(i) = b(i), && (17.7)\\ &= \frac{b(i)}{a(i)} - 1 \text{ si } a(i) > b(i). && (17.8) \end{aligned}$$

Esto se puede expresar en una única ecuación como

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}.$$

En el caso en que el grupo A contenga un único objeto no está muy claro cómo definir $a(i)$. Tomaremos $s(i) = 0$ que es una elección arbitraria. Se comprueba con facilidad que $-1 \leq s(i) \leq 1$ para cualquier objeto i.

Para interpretar el significado de $s(i)$ es bueno ver los valores extremos. Si $s(i)$ es próximo a uno significa que $a(i)$ es mucho menor que $b(i)$ o lo que es lo mismo, que el objeto i está bien clasificado pues la disimilaridad con los de su propio grupo es mucho menor que la disimilaridad con los del grupo más próximo que no es el suyo. Un valor próximo a cero significa que $a(i)$ y $b(i)$ son similares y no tenemos muy claro si clasificarlo en A o en B. Finalmente un valor de $s(i)$ próximo a -1 significa que $a(i)$ es claramente mayor que $b(i)$. Su disimilaridad media con B es menor que la que tiene con A. Estaría mejor clasificado en B que en A. No está bien clasificado.

Los valores de $s(i)$ aparecerán representados para cada cluster en orden decreciente. Para cada objeto se representa una barra horizontal con

[6] No parece un nombre inadecuado.

Tabla 17.1: Silueta media y estructura en un conjunto de datos

SC	Interpretación
$0.71 - 1.00$	Fuerte estructura
$0.51 - 0.70$	Estructura razonable
$0.26 - 0.50$	Estructura débil. Probar otros métodos
≤ 0.25	No se encuentra estructura

longitud proporcional al valor $s(i)$. Una buena separación entre grupos o cluster viene indicada por unos valores positivos grandes de $s(i)$. Además de la representación gráfica se proporciona un análisis descriptivo. En concreto la media de los valores de la silueta dentro de cada cluster y la media de la silueta para todo el conjunto de datos. La clasificación será tanto mejor cuanto mayor sean estos valores medios. De hecho, se puede decidir el número de grupos en función del valor medio de las silueta sobre toda la muestra. Vamos probando distintos números de grupos y nos quedamos con el número que nos da la silueta media máxima.

¿Cuándo podemos decir que hay estructura de grupos en los datos que estamos analizando? Experiencias con datos sugieren la tabla 17.1.

Ejemplo 17.8 (ruspini). *Veamos el resumen de la silueta.*

```
ruspini.pam = pam(ruspini,4)
summary(silhouette(ruspini.pam))
```

```
Silhouette of 75 units in 4 clusters from pam(x = ruspini, k = 4) :
 Cluster sizes and average silhouette widths:
      20 23 17 15
0.7262347 0.7548344 0.6691154 0.8042285
Individual silhouette widths:
   Min. 1st Qu. Median Mean 3rd Qu. Max.
 0.4196 0.7145 0.7642 0.7377 0.7984 0.8549
```

También podemos representarla gráficamente en la figura 17.11.

```
png(paste0(dirTamiFigures,"Cluster63.png"))
fviz_silhouette(ruspini.pam)
dev.off()
```

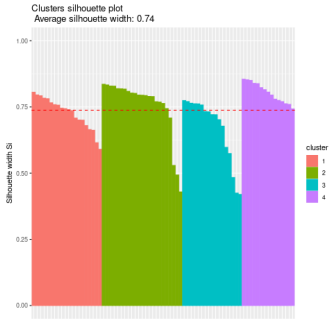


Figura 17.11: Silueta de la clasificación de los datos ruspini con método PAM y cuatro grupos.

17.9 Análisis cluster y datos de expresión de gen

137

En las secciones previas hemos visto algunos métodos de clasificación así como de valoración de hay estructura cluster en un conjunto de datos dados. Estos métodos se aplican (sin modificación) en contextos muy diversos. Sin embargo, ¿qué interés particular tiene su aplicación para datos de expresión de gen? Lo primero a considerar es que nos podemos plantear o bien la clasificación de las filas de la matriz de expresión o bien la clasificación de las columnas. Es decir, tiene sentido biológico clasificar genes o bien las muestras.

Si clasificamos las filas lo que tenemos en cada fila es el perfil de expresión del gen en todas las muestras del estudio. En este caso, puede ser

[137] Muchos comentarios e ideas de esta sección y del resto del capítulo las he tomado de la excelente presentación [43].

que los grupos que obtengamos nos puedan servir posteriormente (quizás cruzando información con alguna de las bases de datos de anotación de genes) para realizar un enriquecimiento del conjunto.

Si clasificamos las columnas, las muestras, entonces podemos estar, por ejemplo, buscando subclases de células tumorales. Esto es, tenemos muestras que, a priori, son indistinguibles y posiblemente tengamos alguna subclase que no podíamos preveer.

Los resultados de la clasificación de genes y muestras puede utilizarse como control de calidad. Sobre todo en las muestras tenemos un diseño experimental previo (habitualmente, el diseño grupo control frente a grupo de tratamiento).

17.10 Ejemplos

En esta sección incluimos análisis cluster completos de algunos bancos de datos. Utilizamos herramientas no solamente de este tema sino también de los anteriores.

17.10.1 Un análisis de los datos ALL

Son los datos del paquete [83, ALL] (§ 2.7.2). Cargamos los datos.

```
data("ALL",package="ALL")
```

Podemos comprobar que es un `ExpressionSet`.

```
class(ALL)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Se realiza el preprocesado de los datos que indicamos en § 2.7.2.

```
bcell = grep("^B",as.character(ALL$BT))
types = c("NEG","BCR/ABL")
moltyp = which(as.character(ALL$mol.biol) %in% types)
all = ALL[,intersect(bcell,moltyp)]
tipos = ALL$mol.biol[intersect(bcell,moltyp)]
tipos = factor(tipos) ## Eliminamos categorías vacías
```

Vemos que `all` es un `ExpressionSet`.

```
class(all)
```

```
[1] "ExpressionSet"
attr(,"package")
[1] "Biobase"
```

Vamos a realizar un filtrado independiente de los genes. En concreto los criterios estadísticos serán que el nivel medio de expresión sea mayor que un cierto valor mínimo tanto en un tipo de biología molecular como en el otro.

```
alltemp0 = genefilter::nsFilter(all,var.func = mean, var.cutoff = 0.7)
alltemp1 = genefilter::nsFilter(alltemp0$eset,var.func = IQR, var.cutoff = 0.7)
all1 = alltemp1$eset
```

```
pacman::p_load(cluster)
all1.d = dist(exprs(all1),method = "manhattan")
all1.km = kmeans(exprs(all1),2)
table(all1.km$cluster)
```

```
  1  2
510 280
```

```
summary(silhouette(all1.km$cluster,all1.d))
```

```
Silhouette of 790 units in 2 clusters from silhouette.default(x = all1.
    ↪ km$cluster, dist = all1.d) :
 Cluster sizes and average silhouette widths:
     510 280
0.4038224 0.2893169
Individual silhouette widths:
   Min. 1st Qu. Median Mean 3rd Qu. Max.
-0.1020 0.2432 0.4152 0.3632 0.4926 0.5804
```

Y mostramos la silueta en figura 17.12.

Figura 17.12: Silueta

```
pacman::p_load(factoextra)
sil0 = silhouette(all1.km$cluster,all1.d)
p = fviz_silhouette(sil0)
```

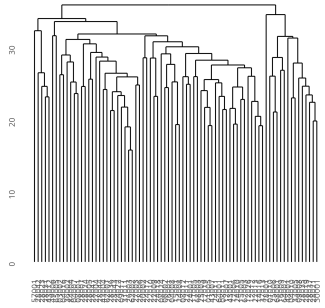
```
  cluster size ave.sil.width
1  1  510 0.40
2  2  280 0.29
```

Realizamos un cluster jerárquico aglomerativo.

```
all1.ag = hclust(all1.d,method="average")
```

O un análisis jerárquico de las muestras.

```
dd = dist(t(exprs(all1)),method="euclidian")## Transpuesta de la matriz
                                             ## de expresión
muestras.ag = hclust(dd,method="average")
p = ggdendro::ggdendrogram(muestras.ag)
```



Recordemos que las muestras correspondían a dos grupos que vienen definidos por la covariable `mol.biol` que nos daba la biología molecular. Veamos si hay coincidencia entre estos tipos y los grupos que hemos obtenido.

```
tipos.ag = cutree(muestras.ag,2)
```

Y construimos la tabla de contingencia.

```
table(tipos,tipos.ag)
```

```
       tipos.ag
tipos  1  2
  BCR/ABL 33 4
  NEG 31 11
```

Mediante un `heatmap` podemos ver conjuntamente los dos dendogramas y una representación gráfica de los niveles de expresión.

```
png(paste0(dirTamiFigures,"Cluster79.png"))
heatmap(exprs(all1),hclustfun=agnes)
dev.off()
```

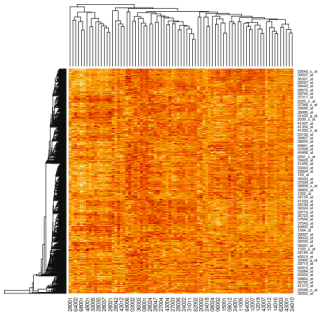


Figura 17.13: Heatmap

¿Para qué nos puede servir un dibujo como este? Básicamente podemos intentar ver si hay *grupos formados por filas (genes) y columnas (muestras)*.

Supongamos que nos planteamos un PAM y un k-medias con dos grupos para las muestras. Vamos a clasificar con cada uno de los métodos e intentar ver si hay relación con la variabel `tipos` que contiene la biología molecular.

```
all1.pam = pam(t(exprs(all1)),2)
table(tipos,all1.pam$cluster)
```

```
tipos 1 2
  BCR/ABL 26 11
  NEG 24 18
```

```
all1.km = kmeans(t(exprs(all1)),2)
table(tipos,all1.km$cluster)
```

```
tipos 1 2
  BCR/ABL 27 10
  NEG 21 21
```

No parece que tengamos ninguna relación clara. Como siempre hay que seguir trabajando los datos a ver si se encuentra algo.

17.10.2 Análisis cluster de GSE20986

Utilizamos los datos preprocesados. Los cargamos.

```
data(gse20986,package="tamidata")
```

```
library(genefilter)
gse0 = nsFilter(gse20986,var.func = mean, var.cutoff = 0.7)
gse1 = nsFilter(gse0$eset,var.func = IQR, var.cutoff = 0.7)
gse = gse1$eset
```

¿Con cuántas filas nos hemos quedado?

```
dim(exprs(gse))
```

```
[1] 1877 12
```

Vamos a clasificar las muestras y compararemos con la clasificación original de las mismas. Utilizamos distancia Manhattan y promedio con el método PAM. Primero calculamos la matriz de distancias.

```
d0 = dist(t(exprs(gse)),method = "manhattan")
```

Y aplicamos PAM.

```
library(cluster)
gse.pam = pam(d0,diss = TRUE,k=4)
```

La clasificación original es

```
tipo0 = rep(1:4,rep(3,4))
```

```
table(tipo0,gse.pam$clustering)
```

```
tipo0 1 2 3 4
   1 1 1 1 0
   2 2 1 0 0
   3 3 0 0 0
   4 0 0 0 3
```

Lo que sale es muy interesante y tiene una interpretación.

17.11 Otras aproximaciones y problemas

Sin duda, un problema de gran interés es lo que se conoce como biclustering o co-clustering. En este caso intentamos grupos de filas (genes) y columnas (muestras) que muestren unos valores de expresión similares. Es un problema importante pero no lo tratamos (de momento) en estas notas. Una referencia fundamental de biclustering es [91].

La clasificación es tanto mejor cuanta más información se utiliza. En particular un trabajo de interés para clasificación de genes incorporando información de Gene Ontology es [141].

Cuando se clasifican muestras el mayor problema es la enorme dimensión de los vectores que clasificamos (N en nuestro caso). Una posibilidad es seleccionar genes y no utilizarla la expresión para todos ellos. En [2] se propone un método utilizando información de Gene Ontology.

17.12 Ejercicios

Ex. 35 — Utilizando los datos `ALL` se pide:

1. Filtrar las muestras (columnas) con biología molecular "BCR/ABL" y "NEG".
2. Quedarse con los genes (filas) que verifiquen las siguientes condiciones:
   (a) La media de los niveles de expresión sobre las biologías moleculares indicadas sea superior a 5.62.
   (b) La desviación estándar de las expresiones del gen ha de ser superior a 1.85.
3. Aplicar un análisis cluster jerárquico a los genes. En concreto:
   (a) Hay que representar el dendograma.
   (b) Decidir por inspección visual qué número de grupos parece razonable. Supongamos que denotamos por k el número de grupos.
   (c) Representar la silueta y obtener su anchura media.
4. Aplicar un análisis cluster jerárquico a las muestras. En concreto:
   (a) Hay que representar el dendograma.
   (b) Decidir por inspección visual qué número de grupos parece razonable. Supongamos que denotamos por k el número de grupos.
   (c) Representar la silueta y obtener su anchura media.
5. En este apartado vamos a aplicar un k-medias. En concreto vamos a ir clasificando según este método y nos quedaremos con el número de grupos que tengan una anchura media de la silueta mayor.

6. Repetir el apartado anterior utilizando el método PAM. Discutir el número de grupos.
7. Comparar las clasificaciones obtenidas en los dos apartados anteriores.

Capítulo 18

Miscelánea

En el texto hemos utilizado una serie de técnicas y conceptos estadísticos. Es preferible, antes que insertar su explicación en el texto incluirlos en un capítulo aparte. Cada sección responde a un concepto distinto. No hay ningún tipo de continuidad entre las secciones. Es un capítulo auxiliar para el resto de los capítulos. Cuando explicamos los conceptos también indicamos cómo hacerlo con R.

18.1 Algoritmo bipesado de Tukey de un solo paso

Su denominación en la literatura estadística sería *One-Step Tukey's Biweight Algorithm*. Veamos cómo funciona. Tenemos unos datos $x_1, \ldots, x_n$ de los cuales pretendemos tener algún tipo de valor medio o de valor central que no esté muy afectado por valores anómalos como lo está, por ejemplo, la media muestral.

1. Calculamos la mediana m_x de los datos $x_1, \ldots, x_n$.
2. Determinamos las distancias $d_i = |x_i - m_x|$.
3. Calculamos la mediana m_d de d_i con $i = 1, \ldots, n$. Esta mediana m_d o mediana de las desviaciones absolutas se suele denotar com MAD (median absolute deviation). Es una medida de dispersión.
4. Consideramos[1]
$$u_i = \frac{x_i - m_x}{cm_d + \epsilon},$$
con $i = 1, \ldots, n$. Los valores por defecto son $c = 5$ y $e = 0.0001$ (con el valor de e evitamos la división por cero).
5. Consideremos una función bicuadrada (bisquare) definida como
$$w(u) = \begin{cases} (1-u^2)^2 & \text{si } |u| \leq 1 \\ 0 & \text{si } |u| > 1 \end{cases}$$
6. Se define el estimador en un solo paso como
$$T_{bi}(x_1, \ldots, x_n) = \frac{\sum_{i=1}^n w(u_i) x_i}{\sum_{i=1}^n w(u_i)}.$$

[1] En la implementación original del procedimiento $c = 1$ y $\epsilon = 0$.

Notemos que si sustituimos el valor m_x por T_b podríamos aplicar iterativamente el procedimiento hasta que se estabilice. No lo hacemos. Simplemente se hace una sola iteración.

Ejemplo 18.1. *Supongamos que tenemos los datos siguientes*

```
x = c(12,3,45,21,34,35,21,456)
```

Y ahora podemos usar la función `tukey.biweight` *del paquete [50, affy].*

```
library(affy)
tukey.biweight(x, c = 5, epsilon = 1e-04)
```

```
[1] 25.1902
```

18.2 Median polish

Tenemos una matriz de datos tal que en la fila i y columna j tenemos y_{ij} con $i = 1, \ldots, I$ y $j = 1, \ldots, J$. El procedimiento del median polish consiste en expresar estos valores de la siguiente forma

$$y_{ij} = m + a_i + b_j + e_{ij}$$

de forma que se verifique que:

- La mediana de $\{a_1, \ldots, a_I\}$ sea cero. Abreviadamente denotamos $Mediana\{a_1, \ldots, a_I\} = 0$.
- $Mediana\{b_1, \ldots, b_J\} = 0$.
- $Mediana\{e_{i1}, \ldots, e_{iJ}\} = 0$ para cada i.
- $Mediana\{e_{1j}, \ldots, e_{Ij}\} = 0$ para cada j.

Pueden haber muchas soluciones para las ecuaciones y restricciones que acabamos de considerar. Consideremos la siguiente matriz.

$$\begin{array}{ccc} y_{11} & \cdots & y_{1J} \\ \vdots & & \vdots \\ y_{I1} & \cdots & y_{IJ} \end{array}$$

El algoritmo median polish (propuesto por Tukey y Mosteller) consiste en aumentar la matriz con una columna y una fila adicionales. Consideremos la matriz aumentada siguiente

$$\begin{array}{ccc|c} e_{11} & \ldots & e_{1J} & a_1 \\ \vdots & & \vdots & \vdots \\ e_{I1} & \ldots & e_{IJ} & a_I \\ \hline b_1 & \ldots & b_J & \text{m} \end{array}$$

El procedimiento actúa iterativamente sobre esta matriz aumentada del siguiente modo:

1. Inicializamos el procedimiento con todos los $a_i = 0$ y $b_j = 0$, $m = 0$ y $e_{ij} = y_{ij}$.
2. Calculamos la mediana de las columnas de la 1 a la I.

3. Restamos dicha mediana a cada elemento de las filas de la 1 a la I y se la sumamos a la columna I+1, esto es, se la sumamos a los valores a_i y m.
4. Calculamos la mediana de cada columna para
5. Restamos dicha mediana a cada elemento de la columna.
6. Sumamos, para cada columna j, la mediana de la columna al valor de la última columna.
7.

Ejemplo 18.2. *Consideremos los datos.*

```
y = matrix(c(18,11,8,21,4,13,7,5,16,7,15,6,7,16,6,19,15,12,18,5),
  ncol=5)
rownames(y) = paste("chip",1:4)
colnames(y) = paste("sonda",1:5)
```

Veamos los datos.

```
y
```

```
      sonda 1 sonda 2 sonda 3 sonda 4 sonda 5
chip 1 18 4 16 7 15
chip 2 11 13 7 16 12
chip 3 8 7 15 6 18
chip 4 21 5 6 19 5
```

Y aplicamos una median polish.

```
medpolish(y)
```

```
1: 82
Final: 82
```

```

Median Polish Results (Dataset: "y")

Overall: 10.5

Row Effects:
chip 1 chip 2 chip 3 chip 4
    5 2 -2 -4

Column Effects:
sonda 1 sonda 2 sonda 3 sonda 4 sonda 5
   1.0 -1.5 0.0 0.5 -0.5

Residuals:
      sonda 1 sonda 2 sonda 3 sonda 4 sonda 5
chip 1 1.5 -10 0.5 -9 0
chip 2 -2.5 2 -5.5 3 0
chip 3 -1.5 0 6.5 -3 10
chip 4 13.5 0 -0.5 12 -1
```

18.3 Datos faltantes

Tenemos nuestra matriz de datos de expresión X y tenemos datos faltantes en algunas características para algunas muestras. Una posibilidad es no perder datos utilizando alguna técnica que impute un valor a los datos que no tenemos. En § 18.3.1 explicamos el procedimiento utilizado por el método SAM (§ 9.1) y por el método GSA (§ 13.9).

18.3.1 Algoritmo de normalización del k-vecino más próximo

Fijamos el número de vecinos k a utilizar en la imputación. Por defecto se toman $k = 10$. El procedimiento es el siguiente:

1. Para cada gen i con al menos un valor faltante:
   (a) Denotamos por S_i las muestras para los cuales *conocemos* la expresión del gen.
   (b) Determinamos los k-vecinos más próximos del gen i utilizando solamente las muestras S_i para calcular la distancia euclídea. Si los otros genes tienen datos faltantes en las muestras S_i se utilizan las muestras de las cuales se conoce su expresión.
   (c) Para las muestras que no están en S_i asignamos el promedio de los k-vecinos más próximos. Si alguno de estos k-vecinos no tiene alguna expresión entonces hacemos la media de los que disponemos.
2. Si un gen tiene todavía un dato faltante entonces asignamos la expresión media de la muestra considerada.

El algoritmo anterior puede ser lento cuando tenemos muchos genes. Con objeto de acelerar el proceso de asignación de datos faltantes se combina con el siguiente procedimiento de agrupamiento recursivo basado en el k-medias (§ 17.7.1).

1. Si el número de genes p es mayor que p_{max} (por defecto, vale 1500):
   (a) Ejecutamos una clasificación del procedimiento k-medias con k = 2 al conjunto de los genes (clasificamos genes y no muestras). Las distancias entre los genes están basadas en las muestras para las cuales se conocen ambas expresiones.
   (b) Formamos dos matrices de expresión una con cada conjunto de genes. Aplicamos recursivamente el paso anterior.
2. Si $p < p_{max}$ entonces aplicamos el procedimiento de imputación utilizando el promedio de los k-vecinos más próximos en cada uno de los grupos encontrados previamente.

El procedimiento fue propuesto en [139] y está implementado en la función impute::impute.knn.

18.4 Dibujo de la media-diferencia de Tukey o dibujo de Bland-Altman

El dibujo media-diferencia de Tukey es más habitualmente conocido como el dibujo Bland-Altman estos autores lo popularizaron utilizándolo en una revista médida.[138]

[138] El trabajo completo (con correcciones respecto de la publicación original) es Bland-Altman y corresponde con la referencia [18].

Supongamos que tenemos pares de medidas (x_i, y_i) con $i = 1, \ldots, n$ y pretendemos ver el grado de coincidencia de las mismas. O, dicho de otro modo, si valores mayores corresponden con diferencias mayores entre las cantidades observadas. Es de suponer que dos procedimientos que coincidan tendrán un coeficiente de correlación alto. Sin embargo, este

valor viene condicionado por la variabilidad de las medidas. El dibujo media-diferencia de Tukey o dibujo de Bland-Altman es una representación gráfica que sirve para valorar el grado de coincidencia de los pares de medidas. Supongamos que tenemos los pares (x_i, y_i) con $i = 1, \ldots, n$ entonces consideramos los puntos que tienen coordenadas

$$\left(\frac{x_i + y_i}{2}, (x_i - y_i)\right).$$

Es decir, en abscisas tenemos la media de las dos medidas $\frac{x_i+y_i}{2}$ y en ordenadas tenemos su diferencia, $x_i - y_i$.

En ocasiones, este dibujo se completa con dos líneas horizontales. Si $d_i = x_i - y_i$ entonces podemos completar la representación gráfica añadiendo dos líneas horizontales correspondientes con los límites del intervalo de confianza para la diferencia media entre los dos procedimientos esto es las líneas tendrían como ordenadas $\bar{d} \pm t_{n-1,1-\frac{\alpha}{2}} \frac{S}{\sqrt{n}}$ donde S es la desviación estándar de las diferencias d_i y $t_{n-1,1-\frac{\alpha}{2}}$ es el percentil $1 - \frac{\alpha}{2}$ de una t de Student con $n - 1$ grados de libertad.

Parte V

Investigación reproducible

Capítulo 19

Investigacion reproducible y repetible

Son dos conceptos fundamentales en investigación y, con frecuencia, confundidos.

Por investigación reproducible entenderemos que un investigador ha de poder reproducir la investigación realizada por otro. No repite la investigación en su propio laboratorio. Simplemente ha de disponer de todo lo necesario para poder reproducir lo que el otro ha hecho. Para ello obviamente debe disponer, al menos, de los datos originales, del código utilizado y una información detallada del análisis realizado. Y, de un modo más vago, de todo lo necesario para poder reproducir los resultados que el investigador original obtenía. Esto es factible y debiera de ser obligado en todas las disciplinas. El análisis de datos ómicos quizás sea de las áreas más reproducibles (sin echar las campanas al vuelo).

Por investigación repetible entramos en la propia experimentación. Este es un objetivo a alcanzar. Una investigación no repetible obviamente no es válida. Puede (de hecho, se asume) que no es intencionado pero no limita el hecho de que no es válida. Es claro que en los detalles está lo importante. Debe haber un buen diseño, unas potencias de los tests adecuadas, un trabajo de laboratorio riguroso. Incluso así puedes obtener conclusiones no repetibles, en definitiva, no válidas.

En este capítulo tratamos sobre **investigación reproducible**. Estamos acostumbrados a que los investigadores[139] en sus publicaciones comenten los resultados obtenidos en sus investigaciones. Han obtenido unos datos, los han analizado y, finalmente, los han interpretado y discutido. *H*emos de creer que han producido los datos correctamente, que han aplicado un tratamiento adecuado (que han hecho lo que dicen que han hecho es un mínimo) y que, finalmente, la interpretación de los resultados de las técnicas utilizadas es correcta. Los demás hemos de creer en ellos. Que los distintos paso se han hecho bien. Al menos, lo mejor que han podido. Y que además el nivel de corrección en todas las etapas es suficiente. Si no tenemos los datos, el código y la interpretación: ¿cómo sabemos que el trabajo es correcto? Es una cuestión de fe. ¿Dónde están los datos? Exactamente, no aproximadamente: ¿qué análisis de los datos han realizado? En otras palabras, deme el código para que yo (y cualquier otra persona) pueda reproducir *exactamente* el tratamiento estadístico que se ha realizado. La interpretación es lo único que tenemos: es la publicación científica que nos ha dado la *noticia* de la existencia de la investigación.

[139] En ciencias experimentales.

Por investigación reproducible entendemos procedimientos que permitan *reproducir* la investigación completa. En su totalidad. Y si esto es posible no se debiera de publicar. Algún ejemplo de lo que puede ocurrir si la transparencia no es total es https://www.theguardian.com/world/2020/jun/03/covid-19-surgisphere-who-world-health-organization-hydroxychloroquine?. Sobre los problemas de la publicación científica es conveniente leer [19].

Personalmente añadiría que los distintos elementos han de ser totalmente libres. Los datos han de estar a disposición de la comunidad[140] y el software que se utiliza para realizar los análisis han de ser libres también.[141]

[140] Una vez se han publicado los trabajos.

[141] Todo lo que he usado en este manual es libre. No se ha utilizado sofware propietario ni datos que no estén a disposición de todos en algún repositorio público.

Algunas direcciones de interés con herramientas para investigación reproducible son las siguientes.

1. El task view en el repositorio de R http://cran.r-project.org/web/-views/ReproducibleResearch.html.
2. La página de Harrell http://biostat.mc.vanderbilt.edu/twiki/bin/-view/Main/StatReport
3. Si se elige la opción de investigación reproducible utilizando LaTeXentonces una (muy) buena referencia [109].

Dos buenas referencias son [58, 57].

19.1 Documento dinámico

Este manual es un ejemplo de lo que Yihui Xie [158] llama un **documento dinámico**. Es un documento en donde se combinan las explicaciones metodológicas junto con el código que las implementa. Sobre la obtención de los datos hay que consultar las referencias bibliográficas.

En este tipo de documentos se distinguen dos partes. La primera en el documento dónde se indica el título, autores e información sobre el formato del documento. Suele hacerse en formato YAML o bien en en LaTeX.

La segunda parte está compuesta por bloques (chunks) [142] de dos tipos: chunks de texto y chunks de código. Los chunks de texto se pueden escribir en Markdown § 19.3 o en LaTeX. Los chunks de código pueden usar distintos lenguajes de programación.

[142] Obviamente no es castellano pero es cuestión de tipo que la RAE lo admita.

En este documento nos centramos en R y por ello la opción para investigación reproducible es combinar R bien con Markdown bien con LaTeX. Sin embargo, en otros lenguajes hay otras opciones similares y tan buenas como esta. Es recomendable leer https://blog.ouseful.info/2017/11/15/programming-meh-lets-teach-how-to-write-computational-essays-instead/.

19.2 LaTeX

Es la opción más potente pero que requiere un mayor aprendizaje inicial. Quizás el la opción adecuada para una persona que se acerque a este mundo con un conocimiento previo del lenguaje de marcas conocido con TeXy cuyas macros conocidas como LaTeXson las más usadas. Creo muy recomendable para un principiante [79]. Dos referencias recientes para un nivel más avanzado son [**Kottwitz2024**, **Kottwitz2021**].

Este manual combina el uso de LaTeXcon R utilizando el paquete [159]. Todo ello trabajando con el editor emacs [**sec:301222f**]. El compilador utilizado es LuaLatex.

19.3 Markdown

Es un lenguaje de marcas ligero (minimal). Pretende ser una forma rápida de escribir HTML. Según su autor, John Gruber, "HTML is a publishing format; Markdown is a writing format."

La dirección indicada contiene una exposición de este lenguaje de marcas. Al ser tan limitado en su sintaxis han aparecido diferentes variaciones que lo extienden (entre otras cosas para tablas) de las cuales la más interesante y usada es Pandoc Markdown. Pandoc Markdown.

19.4 Pandoc

Es un programa que convierte textos escritos en distintos lenguajes de marcas. Es absolutamente impresionante la cantidad de posibilidades que ofrece. En https://pandoc.org/ tenemos una detallada exposición de sus posibilidades. Está en la base [159, knitr], [8, rmarkdown] y quarto Quarto.

19.5 knitr

El paquete [159, knitr] ha sido desarrollado por Yihui Xie y suposo un punto de inflexión en investigación reproducible. Nos permite trabajar con Markdown y LaTeXen chunk de texto y con R como lenguaje de programación. Es una herramienta básica en RMarkdown y Quarto. Consultar https://yihui.org/knitr/. En [158] tenemos una detallada y amena exposición.

19.6 RMarkdown

El paquete [8, rmarkdown] incorpora una implementación del lenguaje de marcas Markdown y utilizado conjuntamente con [159, knitr] permite generar unos muy buenos informes. La página http://rmarkdown.rstudio.com/ es el mejor lugar para aprender a manejarlo.

19.7 Quarto

Es una evolución de § 19.6. Permite trabajar con distintos lenguajes de programación en los chunks de código. En particular, aquí nos interesa R y Bash aunque también incluye Python entre otros.

La sintaxis es muy similar a RMarkdown y en lo esencial las etiquetas que especifican el comportamiento de los chunks de código se incluyen dentro del chunk precedidas de `#\|`.

19.8 Entornos de desarrollo

Una vez elijamos una opción de trabajo para investigación reproducible necesitamos un entorno de desarrollo (IDE).

19.8.1 RStudio

RStudio es la mejor opción y la que tiene un manejo más sencillo cuando trabajamos con R y Markdown. También permite trabajar con R y LaTeXaunque no es tan buena la implementación.

19.8.2 emacs

En mi opinión la mejor opción de trabajo es esta y mi opción personal. Sin embargo, no es la más simple de usar. Hay un aprendizaje previo requerido. Es un editor de texto que puede ser configurado para trabajar con muchos lenguajes y para muchas utilidades. Su dirección principal de consulta es https://www.gnu.org/software/emacs/.

ESS Para el trabajo con R tenemos el modo **Emacs Speaks Statistics** (https://ess.r-project.org/).

RMarkdown Para trabajar con RMarkdown tenemos el modo polymode que podemos instalar en el subdirectorio `~/.emacs.d/` ejecutando

```
git clone https://github.com/vitoshka/polymode.git
```

Y luego incluimos en `.emacs` los caminos correspondientes así como los modos necesarios. Hay que añadir las siguientes líneas (sustituyendo el directorio personal por el correspondiente).

```
(add-to-list 'load-path "/home/gag/.emacs.d/polymode
    ↪ /")
(add-to-list 'load-path "/home/gag/.emacs.d/polymode
    ↪ /modes/")
(require 'poly-R)
(require 'poly-markdown)
```

Quarto Para trabajar con Quarto (§ 19.7) hay un `quarto-mode` de emacs en https://github.com/quarto-dev/quarto-emacs. En Debian/Ubuntu tenemos el paquete quarto que nos permite utilizarlo en la shell (por ejemplo, con la bash).

Capítulo 20

Generando un informe

En datos ómicos trabajamos con muchas variables y pocas muestras. Cuando analizamos su posible expresión diferencial marginal o por grupos, o bien cuando hacemos un análisis de asociación. En fin, cualquier análisis con centenares o miles de variables suele acabar con informes largos que contienen tablas interminables. No es un buen material para el experto que ha de interpretar para obtener unas conclusiones. Estas tablas están dando información sobre entidades biológicas (genes, exones, posiciones genómicas, grupos de genes, etc.) que no podemos conocer por su enorme cantidad. Y para cada una de estas entidades tenemos información (resuménes estadísticos, p-valores, etc.). ¿Cómo generar un informe útil? El informe que hemos de proporcionar al experto ha de ser **útil** (y no solo bonito con muchos colores). Posiblemente la mejor opción es generar archivos en HTML con enlaces a bases de datos donde nos proporcionan información sobre las entidades a que referimos nuestro análisis (genes, grupos de genes, SNPs, proteínas o lo que sea que estemos analizando). Por ejemplo, podemos querer enlazar los genes con Gene Ontology. Esto permitirá al experto interpretar los resultados con su navegador consultando una y otra vez rápidamente con las bases de datos. En la práctica estadística clásica se genera un informe (en formatos orientados a texto impreso como puede ser pdf) y el investigador interpreta los resultados y redacta un informe en un formato similar. En este contexto el problema es mayor. Básicamente se trabaja con grandes cantidades de variables y por ello los resultados han de ser validados y nuevas hipótesis generadas para posterior investigación. Un análisis estadístico de datos ómicos nunca es conclusivo. Ha de validarse con técnicas no ómicas pero con mayor precisión.

En este tema nos planteamos el siguiente problema: cómo generar un fichero centrado en las entidades biológicas de interés y que contenga enlaces a sus bases de datos en línea correspondientes y resúmenes estadísticos (p-valores originales, ajustados, q-valores, etc.) y gráficos (con moderación lo de los dibujos).

En § 20.1 vemos un análisis sencillo y cómo asociar distintos identificadores utilizando el paquete [60, annotate]. Construimos un `data.frame` con la información relevante del análisis. En § 20.2.1, § 20.2.2 y § 20.2.3 generamos ficheros HTML a partir del `data.frame` que hemos construido previamente utilizando los paquetes [71, ReportingTools], [160, DT] y [165, kableExtra] respectivamente.

20.1 Generando la información

Utilizamos como ejemplo los datos `tamidata::GSE20986` § 2.7.7.

```
pacman::p_load(Biobase)
data(gse20986,package="tamidata")
```

Comparamos las muestras extraídas de células endoteliales vasculares de la retina y de células endoteliales de la vena umbilical (abreviadamente células de la retina y huvec). Con el siguiente código (§ 5 y § 8) aplicamos un t-test, a los p-valores obtenidos le aplicamos el método de Benjamini-Hochberg utilizando un tasa de falso rechazo (FDR) de 0.05. Los genes significativos son los que utilizamos en el resto de tema para ilustrar la generación de un informe a partir de ellos.

```
eset = gse20986[, c(2, 3, 5, 10:12)]
tissue = factor(rep(1:2, each = 3), levels = 1:2,labels = c("retina", "huvec"))
tt = genefilter::rowttests(eset, tissue)
padj = p.adjust(tt[,"p.value"],method="BH")
sig = which(padj < 0.05)
```

En el vector `sig` tenemos los índices de las filas que ocupan, en la matriz de expresión, los genes declarados significativos por el procedimiento indicado. Podemos ver las primeras filas.

```
head(sig)
```

```
[1] 163 262 316 317 336 359
```

Tenemos 553. Guardamos los p-valores y los p-valores ajustados.

```
pvalor = tt[sig,"p.value"]
pajustado = padj[sig]
```

El primer problema es asociar a los genes (en general, la entidad biológica con la que trabajamos) seleccionados uno o varios identificadores. ¿Qué anotación tienen nuestros datos? Necesitamos el paquete [60, annotate].

```
pacman::p_load(annotate)
```

La anotación la obtenemos con `annotate::annotation()`.

```
annotation(eset)
```

```
[1] "hgu133plus2"
```

Cargamos el paquete correspondiente.

```
pacman::p_load(hgu133plus2.db)
```

Obtenemos los identificadores de nuestros genes.

```
ID = featureNames(eset)[sig]
```

Los primeros serían

```
head(ID)
```

```
[1] "1552487_a_at" "1552626_a_at" "1552701_a_at"
[4] "1552703_s_at" "1552730_at" "1552760_at"
```

Podemos determinar los nombres abreviados de nuestros genes (mostramos los dos primeros en todo lo que sigue).

```
lookUp(ID, "hgu133plus2.db", "SYMBOL")[1:2]
```

```
$`1552487_a_at`
[1] "BNC1"

$`1552626_a_at`
[1] "TMEM163"
```

```
getSYMBOL(ID,"hgu133plus2.db")[1:2]
```

```
1552487_a_at 1552626_a_at
     "BNC1" "TMEM163"
```

Sus nombres con

```
lookUp(ID, "hgu133plus2.db", "GENENAME")[1:2]
```

```
$`1552487_a_at`
[1] "basonuclin zinc finger protein 1"

$`1552626_a_at`
[1] "transmembrane protein 163"
```

Supongamos que queremos obtener información sobre estos genes en la base de datos Ensembl ¿Cuáles son sus identificadores en esta base de datos?

```
lookUp(ID, "hgu133plus2.db", "ENSEMBL")[1:2]
```

```
$`1552487_a_at`
[1] "ENSG00000169594"

$`1552626_a_at`
[1] "ENSG00000152128"
```

¿O en Gene Ontology?

```
lookUp(ID, "hgu133plus2.db", "GO")
```

O bien sus identificadores Entrez.

```
lookUp(ID, "hgu133plus2.db", "ENTREZID")[1:2]
```

Toda la información que podemos obtener se puede consultar en la ayuda del paquete [31, hgu133plus2.db] o bien con

```
ls("package:hgu133plus2.db")
```

Vamos a generar un `data.frame` que vamos a guardar en formato html utilizando distintos paquetes. Obtenemos distintos identificadores con alguna modificación de tipo y valores. En particular es interesante ver cómo generamos la dirección web para los identificadores `ENTREZID`.

```
ID = featureNames(eset)[sig]
Name = as.character(lookUp(ID,"hgu133plus2.db", "GENENAME"))
entrezid = as.character(lookUp(ID, "hgu133plus2.db", "ENTREZID"))
ID[ID == "NA"] = NA
Name[Name == "NA"] = NA
entrezid = ifelse(entrezid == "NA",NA,
   paste0("<a href='http://www.ncbi.nlm.nih.gov/gene/?term=",
          entrezid,"'>",entrezid,"</a>"))
```

La generación de URL a partir de del identificador está implementado en `tami::entrezid2url()`. De un modo análogo lo podemos hacer para otros identificadores. En las funciones `tami::ensembl2url()`, `tami::go2url()` y `tami::kegg2url()` vemos cómo hacerlo para los identificadores Ensembl, Gene Ontology y KEGG.

Posiblemente queramos añadir a estos descriptores los p-valores originales así como los ajustados. Generamos un **`data.frame`** con los distintos identificadores.

```
df = data.frame(ID = ID,Name = Name,entrezid = entrezid,
    pvalor = pvalor,pajustado = pajustado,stringsAsFactors=F)
```

En el **`data.frame`** que acabamos de generar podemos ver que aparecen sondas que no corresponden a ningún gen. Podemos eliminarlas con `stats::na.omit()` y generar el informe sin estas sondas.

```
df = na.omit(df)
```

20.2 Generando un informe en html

20.2.1 ReportingTools

En esta sección vemos cómo generar un informe con [71, ReportingTools].

```
pacman::p_load(ReportingTools)
```

Utilizando `ReportingTools::HTMLReport()` fijamos el nombre del fichero en que guardamos la información así como el directorio en donde queremos guardarlo.

```
foutput = "gse20986_DE"
htmlRep1 = HTMLReport(shortName = foutput,title = foutput,
                      reportDirectory = "./reports/")
```

Guardamos la información y cerramos el fichero con las funciones `ReportingTools::publish()` y `ReportingTools::finish()`.

```
publish(df,htmlRep1)
finish(htmlRep1)
```

```
[1] "./reports//gse20986_DE.html"
```

20.2.2 DT

El paquete [160] permite la generación de tablas en formato html de un modo muy simple y con muchas posibilidades. Si consideramos el **`data.frame`** que hemos generado en el punto anterior

```
ff = DT::datatable(df,escape=FALSE)
```

Podemos guardar el informe generado en el fichero `reports/gse20986_DE_DT.html`.

```
DT::saveWidget(ff, "reports/gse20986_DE_DT.html")
```

20.2.3 Utilizando kableExtra

Vamos a generar el mismo informe utilizando [**kableExtra**].

```
pacman::p_load(kableExtra)
df %>% kable(escape=FALSE) %>%
    kable_styling() %>%
    save_kable("reports/gse20986_DE_kE.html")
```

```
Error: LaTeX failed to compile /home/gag/ownCloud/alltami/tami13/manual
    ↪ /p7_InvestigacionReproducible/reports/gse20986_DE_kE.tex. See
    ↪ https://yihui.org/tinytex/r/#debugging for debugging tips. See
    ↪ gse20986_DE_kE.log for more info.
```

20.3 Generación de enlaces

En § 20.1 hemos necesitado generar en enlace a bases de datos que nos den información sobre la entidad que analizamos (genes en el ejemplo). Esto es habitual y con frecuencia tendremos que construirnos una función que genere el enlace a otra base de datos. En el paquete [**tami**] se han añadido algunos ejemplos que mostramos aquí.

```
tami::entrezid2url
```

```
function (id)
ifelse(id == "NA", NA, paste("<a href='http://www.ncbi.nlm.nih.gov/gene
    ↪ /?term=",
   id, "'>", id, "</a>", sep = ""))
<bytecode: 0x564fa3322648>
<environment: namespace:tami>
```

```
tami::ensembl2url
```

```
function (id, site = "http://www.ensembl.org")
paste("<a href='", site, "/id/", id, "'>", id, "</a>", sep = "")
<bytecode: 0x564f9f2579a0>
<environment: namespace:tami>
```

```
tami::go2url
```

```
function (id)
ifelse(id == "NA", NA, paste("<a href='http://amigo.geneontology.org/
    ↪ amigo/term/",
   id, "'>", id, "</a>", sep = ""))
<bytecode: 0x564fb08b1520>
<environment: namespace:tami>
```

```
tami::kegg2url
```

```
function (id)
ifelse(id == "NA", NA, paste("<a href='http://www.genome.jp/dbget-bin/
    ↪ www_bget?",
   id, "'>", id, "</a>", sep = ""))
<bytecode: 0x564f9f1fab48>
<environment: namespace:tami>
```

```
tami::WormBase2url
```

```
function (id)
ifelse(id == "NA", NA, paste("<a href='http://www.wormbase.org/species/
    ↪ c_elegans/gene/",
   id, "'>", id, "</a>", sep = ""))
<bytecode: 0x564f9f1caa90>
<environment: namespace:tami>
```

20.4 Ejercicios

Ex. 36 — Utilizando el `ExpressionSet` `tamidata::gse1397` se pide:

1. ¿Cuál es el modelo de chip utilizado?
2. Seleccionar el gen en la fila 678. ¿Cuál es su identificador Affymetrix o AffyID? Determinar los identificadores en las siguientes bases de datos: Ensembl, Gene Ontology y Entrez.
3. Elegir al azar 100 genes con la función `stats::sample()`. Determinar sus AffyIDs y los correspondientes en Ensembl, Gene Ontology y Entrez.
   (a) Generar un fichero HTML tal que en las cuatro primeras columnas tengamos los identificadores.
   (b) Repetir el punto anterior pero de modo que, asociado a los identificadores en Ensembl, Gene Ontology y Entrez nos aparezca el enlace para el acceso a la base de datos correspondiente.

Ex. 37 — En el ejemplo descrito en § 20.2.1 incorporar los enlaces a Ensembl.

Capítulo 21

Lo que se hace porque sí

En un texto de Estadística aplicada a datos ómicos parece que lo natural es explicar solamente cómo diseñar un experimento y cómo analizar de un modo (razonablemente) correcto los datos obtenidos. La propia experiencia me dice que raramente (los experimentadores) preguntan antes ni tienen interés en algo que no coincida con sus propias ideas. La modestia (no) es la madre de la ciencia. O la humildad. No lo sé. Buscan a un alma (caritativa aunque no necesitan caridad, ellos la poseen) que les demuestre lo inteligentes, intuitivos (el masculino no lo uso como genérico) que son. Sin haber hecho nada sabían el resultado.

En este tema hablamos de barbaridades. Pequeñas y grandes barbaridades. Las realmente peligrosas son las pequeñas por lo frecuentes. Ejemplos que me he ido encontrando de lo que se hace y (en mi opinión que no suele ser compartida por el excelso investigador que suele estar aprovechándose de una cohorte de becarios que suspiran porque el tiempo se detenga y pueda seguir siendo un becario/a felizmente explotado) no se debe hacer. Responde a una experiencia personal. Vamos a explicar en negativo (lo incorrecto) y no en positivo (lo correcto) como se pretende en el resto del texto. No es una mala opción. Si todo el texto se hiciera así sería muy largo ya que el número de incorrecciones que se pueden cometer es infinito.

21.1 ¿Variables u observaciones? ¿Qué estoy analizando?

Solo Dios lo sabe. En datos ómicos se tienen muchas variables y pocas observaciones. Existe una tradición (extraña para mi) de disponer los datos (de expresión y no solo de expresión) en una forma no habitual en Estadística. En concreto en la matriz de expresión tenemos en **filas** las variables (correspondientes a sondas, genes, isoformas, proteínas, etc.) y en columnas tenemos las observaciones. Pero **no hay que olvidar** lo que son observaciones y lo que son las variables. Esto parece obvio pero no lo es. Además tenemos la desgracia de que estamos observando una misma característica pero en distintos entes biológicos. Esto hace que el avispado (Dios le perdone) compare libremente dentro de una misma observación (columna de la matriz de expresión) diferentes conjuntos de valores correspondientes a distintos conjuntos de filas (por ejemplo, distintos grupos de genes). Utilizando lo que Dios le da a entender o que les sale significativo. Más bien lo segundo porque lo único que suele te-

ner interés es que sea **significativo** (que hay que publicar un artículo al mes). [143]

Hay revistas que se ocupan de estudiar y seguir los problemas en el mundo de la investigación científica. En particular es recomendable la revista Retraction Watch. También es aconsejable consultar Undark.

[143] Personalmente se lo he intentado explicar a algún catedrático de Bioquímica pero como el que oye llover. Además el argumento de la respuesta es brillante -Pero si me lo ha publicado en una buena revista la primera versión y las siguientes que solamente añaden algún dato más- No hay respuesta frente a esto. No la tengo. Me gustaría tenerla. Está publicado y ni Dios puede decir que no está bien. Y siguen haciéndolo. Porque tienen la verdad que les da sus 12 publicaciones mínimas al año. Menos mal que muchas revistas ya son electrónicas y, al menos, los árboles sufren menos.

21.2 Mejor no usarlo

Un breve listado de cosas que hay que evitar.

Diagramas de sectores Los periódicos los aman. Los experimentadores también. Y no lo entiendo. Lo más habitual entre los estadísticos es odiarlos. En mi caso con toda mi alma. No usarlos. Es mucho más claro un diagrama de barras como opción alternativa si queremos un dibujo de frecuencias. La comparación entre las frecuencias de las distintas categorías es mucho más simple. Evitarlos.

Histogramas Los histogramas como una estimación de la función de densidad de la variable (aleatoria) que estamos estudiando tienen una gran tradición en Estadística. Su principal problema es que la elección del número de clases es crítica. El dibujo cambia mucho cuando modificamos el número de clases. Depende demasiado de una elección previa del usuario. Los programas llevan procedimientos que fijan este número. Es mejor utilizar un estimador no paramétrico de la densidad o estimadores basados en funciones kernel. En el texto lo usamos. Con grandes cantidades de datos dan una muy buena (y robusta) estimación de la función de densidad de la variable aleatoria.

21.3 Combinando información de distintos experimentos

Las revistas científicas han introducido el buen hábito de obligar a que se depositen los datos en repositorios público. Este texto utiliza datos de estos repositorios. Esto es muy bueno. Además debiera de obligarse a poner el código que se ha desarrollado en el lenguaje o lenguajes que sean para poder reproducir el estudio. Este último punto todavía no es habitual pero creo que lo será en no mucho tiempo. Al menos en buenas revistas. Tener muchos bancos de datos disponibles es bueno. Pero puede ser malo, de hecho, puede ser malísimo si no se usan con sensatez. Ha aparecido otra bibliografía que utiliza estos datos *prestados* para realizar análisis conjuntos o **metaanálisis**. El meta-análisis es una parte de la Estadística que se explica poco. No suele estar en los programas de las asignaturas básicas. Y podría estar. Y debiera de estar porque se están cometiendo auténticos crimenes (contra la humanidad).

En datos ómicos hay una cierta confusión entre qué son las variables y qué son las observaciones. Y a esta confusión no ayuda el costumbre de representar las matrices de expresión traspuestas. La matriz de datos tiene habitualmente las filas correspondiendo con las observaciones y las columnas correspondiendo a las variables que observamos en dichas observaciones. Sin embargo, con datos ómicos es muy habitual (y en este texto seguimos este hábito) representar la matriz de expresión de modo

que las columnas son observaciones o muestras y las filas son las variables (genes, sondas, grupos de sondas, etc). Las columnas o muestras u observaciones son independientes mientras que las filas no lo son. Es conveniente seguir esta costumbre porque los paquetes de Bioconductor lo hacen y son nuestra fuente principal de software estadístico.

Cuando combinamos información de distintos experimentos es claro que las variables pueden parecer ser las mismas pero no serlo. Una variable puede estar cuantificando información sobre un mismo gen pero tiene porqué tener una misma distribución de probabilidad. No tenemos muestras aleatorias en donde se asume una distribución común. Por ejemplo, nos colocamos en la mejor de las situaciones. Cada experimento utiliza el mismo chip. ¿Podemos simplemente combinar las columnas y metadatos? Los distintos experimentos se han realizado en distintos laboratorios. ¿Sus protocolos experimentales son los mismos? ¿El material biológico se ha seleccionado de la misma forma? ¿Hay influencia del laboratorio? Son preguntas que si tienen respuesta negativa significa que no observamos lo mismo aunque sea el mismo chip y el mismo diseño.

Por supuesto si el chip no es el mismo no tiene sentido combinar la información considerando que tenemos una muestra mayor y quedándonos con aquellos genes que están en todos los chips. Aunque hablamos de unos mismos genes no tenemos la misma variable aleatoria. Si la covariable que utilizamos para evaluar los perfiles de expresión es la **misma** entonces podemos pensar en realizar un meta-análisis utilizando los p-valores pero no combinando efectos. No tenemos la misma variable.

¿Y con RNA-seq? La cosa no es tan simple de evaluar. Es claro que el tema de los protocolos han de ser los mismos. Y luego hay que utilizar los mismos alineadores y con los mismos parámetros. Hay que considerar los posibles errores sistemáticos producidos por los procedimientos de alineamiento. En este sentido si se utilizan datos procesados disponibles en la red no parece razonable combinar esta información. Se debe de utilizar el mismo material base, con lecturas de la misma longitud, conseguidas con el mismo producto.

Una práctica (que espero no se transforme en habitual) consiste en construir bancos de datos reunión datos de distintos experimentos.

Esta confusión hace que a veces se toman distintos experimentos. De cada uno de estos experimentos se observa una variable distinta asociada a un mismo gen. Y se construye una matriz de datos donde en filas tenemos genes y en columnas tenemos variables medidas en distintos experimentos. Las variables que observamos en distintas experimentaciones aunque se refieran a un mismo gen son observaciones independientes que además no están observadas en las mismas condiciones experimentales. Por lo tanto no podemos estudiar la dependencia que pueda existir entre ellas. Insisto, son valores independientes de distribuciones distintas y la posible dependencia que observemos será puro ruido. Y se hace.

21.4 Yo con la Excel me apaño

Es frecuente oir esta frase. Un artículo de interés sobre el problema de manejar nombres de genes en esta hoja de cálculo la podemos encontrar en https://retractionwatch.com/2023/09/20/guest-post-genomics-has-a-spreadsheet-problem/#more-127892. Quizás no es muy aconsejable su uso en el contexto de los datos ómicos.

Cuando intentas enseñar (siendo recomendable estudiártelo antes)

cómo resolver un problema estadístico utilizando herramientas como R o Bioconductor es frecuente oir: "Yo con la Excel me apaño". ¿Qué hay que hacer en estos casos? Hablar de series, películas o literatura y acabar la conversación en menos cinco minutos. Y **no** volver a hablar de trabajo con esa persona.

¿Qué hacer con la versión completa?: "Yo con la Excel y el Word me apaño". En esta situación extrema la buena educación no se aplica. Tenemos derecho a no responder, darnos la vuelta y largarnos. Es la única alternativa. Hasta la educación tiene sus límites y hay muchas cosas interesantes de qué ocuparnos y preocuparnos.[1]

21.5 Datos de conteo y t-test

[144] Respuesta a consulta de 03/06/20.

[144] Tenemos dos condiciones y nuestra información son conteos (números enteros no negativos). ¿Por qué no comparar las medias en las dos condiciones utilizando un test de la t? Al final de la historia un conteo es un número. Podemos promediar conteos. Podemos calcular su desviación estándar. En resumen podemos calcular los estadísticos y sus correspondientes p-valores. Aplicamos un test de la t y con una gran probabilidad el correspondiente revisor del trabajo encontrará correcto el tratamiento. No lo es. Son números enteros. El modelo probabilístico utilizado en los test de la t es la distribución normal. Y claramente un conteo no sigue una distribución normal. Además tenemos opciones alternativas utilizando modelos lineales generalizados que permite analizar estas situaciones. Tan potentes y, casi, tan desarrolladas como la teoría basada en la distribución normal. Por ello, hay que utilizar aproximaciones basadas en las distribuciones de probabilidad habituales con este tipo de datos: binomiales, Poisson, binomial negativa. Esta es la razón por la cual los métodos que se ven el manual y que tratan con este tipo de datos no utilizan los procedimientos basados en la normal. Efectivamente con tamaños muestrales grandes podemos recurrir a resultados como el teorema central del límite que permitirían su aplicación pero en este contexto de datos ómicos no andamos sobrados de muestras y, por tanto, este resultado no sería aplicable.

21.6 A la búsqueda desesperada del p-valor

[145] 11/12/22

[145] Ajustas modelos lineales o modelos lineales generalizados o algún otro tipo de modelización estadística. Explicas que cuando un factor se codifica en variables dummy la interpretación del efecto ha de ser global y de un modo jerárquico descartando del modelo interacciones de orden superior y vamos bajando hasta llegar a los efectos principales. Estos factores si tienen muchos niveles producen bastantes variables y empiezas a realizar más y más contrastes. Por puro azar alguno sale significativo. Este va al artículo. Aunque la interpretación biológica no sea clara. Lo que manda es el p-valor. El sacrosanto p-valor. Es lo único que el experimentador busca. Un confortablemente pequeño p-valor. Un detalle, si el estadístico del contraste tiene distribución nula continua el p-valor tiene una distribución uniforme bajo la hipótesis nula en el intervalo unitario $[0, 1]$. Y la uniforme también produce valores altos y bajos. A esto lo podríamos llamar (siguiendo a Piedrahita) **la avaricia del p-valor**.

[1] Un enlace curioso.

El investigador no pretender encontrar descubrimientos sentatos. No. El investigador busca p-valores del orden de 10^{-6} y de ahí para abajo. Los p-valores próximos a uno también tienen derecho a existir. Y no se les ve.

Parte VI

R/Bioconductor

Capítulo 22

Todo lo que siempre quiso saber sobre R pero nunca se atrevió a preguntar

Nuestra opción de trabajo es la utilización de R y Bioconductor (R/-Bioconductor abreviadamente). No es la única opción pero para análisis estadístico sí. R es excesivo y personalmente estoy un poco harto pero es lo que hay. Es útil y casi todo lo que a mi se me ocurre que tiene interés lo tiene implementado en algún paquete. En este capítulo se dan los conceptos mínimos para trabajar con este lenguaje. Para una introducción recomendamos [25] y para programación más avanzada la referencia imprescindible es [145].

En § 22.1 mostramos: instalación de R; en § 22.3 lo básico del lenguaje; en § 22.4 el manejo con vectores y algunas funciones útiles; en § 22.5 cómo trabajar con matrices. Utilizando los datos `golub` (§ 2.7.1) mostramos en § 22.6 cómo leer datos de un paquete, la función `class`, trabajo con la clase `factor` y algunos dibujos básicos con [150, ggplot2]. El manejo de listas (`list`) lo tenemos en § 22.8 y cómo programar una función básica en § 22.9.

22.1 Sobre la instalación de R

Los pasos a seguir son los siguientes:

1. Bajamos el programa de la siguiente dirección http://cran.r-project.org/.

2. Elegimos versión. Se recomienda cualquier versión de Linux. Si se utiliza Windows (en esto el mundo es bastante libre) elige la correspondiente versión.

3. Una vez hemos bajado el paquete se instala ejecutándolo con las opciones por defecto.

4. En el escritorio tenemos el icono de R. Simplemente clicando el icono iniciamos la sesión de trabajo.

22.2 Paquetes

R tiene muchos paquetes que extienden el R base que acabamos de instalar. De hecho, es casi imposible realizar un análisis estadístico por sencillo que sea sin utilizar paquetes adicionales.

Instalación

Vamos a instalar el paquete [142, UsingR]. Es un paquete con herramientas para la enseñanza de Estadística básica. La opción más simple es escribir en línea de comandos.

```
install.packages("UsingR")
```

Dos paquetes muy convenientes son [**R-devtools**, 121]. Los instalamos.

```
install.packages(c("devtools","pacman"))
```

Podemos instalar de GitHub con

```
devtools::install_github("JasonHackney/ReportingTools")
```

Observemos que para instalar de GitHub tenemos que utilizar "DeveloperName/PackageName".

Otra opción para instalar es utilizar [121, pacman]. La instalación de [142, UsingR] sería[146]

[146] Elegimos como repositorio https://ftp.cixug.es/CRAN.

```
pacman::p_install("UsingR",repos="https://ftp.cixug.es/CRAN/")
```

Cargando uno o varios paquetes

Una vez instalado el paquete, para poder usar las funciones o datos que contenga, debemos *cargarlo* mediante

```
library(UsingR)
```

Sin embargo, si pretendemos cargar varios paquetes hemos de ejecutar la línea anterior para cada uno de ellos. No es muy cómodo. Una opción (https://datasciencetut.com/load-multiple-packages-in-r/) puede ser

```
pkgs = c("UsingR","ggplot2")
lapply(pkgs, library, character.only=TRUE)
```

En este manual veremos que cargamos un paquete con el siguiente código.

```
pacman::p_load(UsingR)
```

Previamente hemos de tener instalado el paquete [121]. En este modo de cargar el paquete si no está instalado se instala de un modo automático. Tiene la ventaja de que podemos instalar varios paquetes simultáneamente con

```
pacman::p_load(UsingR,ggplot2)
```

22.3 Lo primero con R

Una referencia rápida a la sintaxis es http://cran.r-project.org/doc/contrib/Short-refcard.pdf. Una buena idea es imprimirla y tenerla a mano. .

De cómo trabajar con R

Hay dos formas de trabajar con R. La primera opción es utilizar un editor en el que escribimos código de R. Lo copiamos y luego lo pegamos en la línea de comandos de R.[1] Dentro de esta manera de trabajar podemos utilizar distintas editores (que llevan herramientas para facilitarnos el trabajo).

Con el editor de R El propio programa lleva un editor incorporado. Es básico pero suficiente.

RStudio Quizás ahora mismo sea la mejor opción. Es un programa que incorpora el editor, nos muestra las salidas, los gráficos y la historia previa. De manejo muy simple. http://rstudio.org/.

Ayuda con R

Supongamos que buscamos ayuda sobre las opciones de la función que nos dibuja un histograma, `hist`. Lo más simple es utilizar la ayuda en html. Utilizamos la siguiente función.

```
help.start()
```

Vemos que nos abre el navegador y nos ofrece distintas opciones. Quizás la opción más simple sea utilizar la herramienta de búsqueda. Otra opción es

```
?hist
```

O simplemente,

```
help(hist)
```

22.4 Una sola muestra

Supongamos que consideramos una técnica ómica (microarray, RNA-seq). Estamos estudiando la expresión de 10 genes y los valores observados son los siguientes.

```
[1] 3 6 3 6 7 1 4 7 4 4
```

Podemos leer estos datos con

```
x = c(3,6,3,6,7,1,4,7,4,4)
```

Tenemos ahora un objeto llamado `x`

```
class(x)
```

```
[1] "numeric"
```

que es un vector formado por números. ¿Cómo puede ver su contenido?

```
x
```

```
[1] 3 6 3 6 7 1 4 7 4 4
```

[1] Es la opción más educativa en donde aprendemos realmente a trabajar con el programa.

¿Cuál es el primer valor de este vector de datos?

```
x[1]
```

```
[1] 3
```

¿Y el que ocupa la posición 7?

```
x[7]
```

```
[1] 4
```

Podemos ver los datos que están entre el 3 y el 7. Para ello fijémonos en el siguiente código.

```
3:7
```

```
[1] 3 4 5 6 7
```

Cuando ponemos dos enteros separados por : nos devuelve todos los enteros entre el primero y el segundo. Lo mismo lo podemos hacer con

```
seq(3,7,1)
```

```
[1] 3 4 5 6 7
```

Ahora podemos ver los datos que ocupan estas posiciones en el vector x.

```
x[3:7]
```

```
[1] 3 6 7 1 4
```

Podemos tener interés en saber los valores de los datos que ocupan las posiciones 1, 3 y de la 5 a la 8. Estas posiciones las podemos obtener con

```
c(1,3,5:8)
```

```
[1] 1 3 5 6 7 8
```

y los valores de x serían

```
x[c(1,3,5:8)]
```

```
[1] 3 3 7 1 4 7
```

Puede que nuestro interés en ver los datos no venga dado por la posición que ocupan sino por su valor. Por ejemplo, queremos saber cuántos de estos datos superan o son iguales a 4. ¿Cómo lo hacemos? Lo lógico es comparar los valores de x con 4. Lo hacemos con

```
x >=4
```

```
 [1] FALSE TRUE FALSE TRUE TRUE FALSE TRUE
 [8] TRUE TRUE TRUE
```

Vemos que nos devuelve un vector diciéndonos si es cierta o no la condición que hemos preguntado, si es mayor o igual a 4. Pero: ¿qué valores son? Si hacemos

```
x[x >=4]
```

```
[1] 6 6 7 4 7 4 4
```

Nos devuelve los datos que ocupan las posiciones donde se daba la condición, donde la condición era cierta. Podemos saber qué valores toman los datos que son mayores que 37 con

```
x[x > 6]
```

```
[1] 7 7
```

o bien los datos que son mayores que 4 y menores o iguales que 6.

```
x[x > 4 & x ≤6]
```

```
[1] 6 6
```

Podemos querer que los casos que estamos seleccionando estén en un nuevo vector.

```
y = x[x > 4 & x ≤6]
```

Los valores de y son

```
y
```

```
[1] 6 6
```

Podemos querer los que son menores o iguales a 4 **o** mayores que 6.

```
x[x ≤4 | x > 6]
```

```
[1] 3 3 7 1 4 7 4 4
```

¿Y los que son iguales a 4? Observar el uso de ==.

```
x[x == 4]
```

```
[1] 4 4 4
```

¿Y en qué posiciones del vector tenemos los valores iguales a 4.

```
which(x == 4)
```

```
[1] 7 9 10
```

147

De cómo ordenar un vector Supongamos que queremos ordenar el vector x.

```
sort(x)
```

```
 [1] 1 3 3 4 4 4 6 6 7 7
```

Nos devuelve los valores ordenados. Sin embargo, con frecuencia necesitamos saber la posición que ocupaban en el vector original estos valores.

```
sort(x,index.return = TRUE)
```

```
$x
 [1] 1 3 3 4 4 4 6 6 7 7

$ix
 [1] 6 1 3 7 9 10 2 4 5 8
```

[147] El resto de operadores lógicos los encontramos en la ayuda de *Logical Operators*: & (&&) corresponde con la intersección, | (||) corresponde con la unión y la negación de una condición la obtenemos con !.

Hemos ordenado el vector x de menor a mayor. Vamos a ordenar de mayor a menor.

```
sort(x,decreasing = TRUE,index.return = TRUE)
```

```
$x
 [1] 7 7 6 6 4 4 4 3 3 1

$ix
 [1] 5 8 2 4 7 9 10 1 3 6
```

* **Ex. 38 —** Introducir los siguientes datos en R.

```
  [1] 9 5 6 5 2 8 3 2 4 9 8 6 6 11 6
 [16] 6 5 5 4 3 9 8 6 7 1 5 6 3 4 3
 [31] 3 4 4 4 3 3 3 5 4 7 2 5 7 2 5
 [46] 3 3 6 8 4 6 2 4 4 7 5 7 7 7 4
 [61] 6 6 6 0 5 3 4 5 4 2 3 6 4 7 2
 [76] 4 10 8 8 3 3 6 4 6 4 3 7 2 5 5
 [91] 5 4 5 9 5 8 8 5 4 3
```

Se pide:

1. ¿Cuál es el valor medio, la varianza y la desviación estándar de estos valores?
2. ¿Cuántos de los valores son 3?
3. ¿Cuál es el rango intercuartílico?
4. Ordena el vector y calcula las sumas acumuladas de los valores ordenados con la función base::cumsum.

* **Ex. 39 —** Ejecutad el siguiente código.[148]

```
x = rnorm(23489,mean=45,sd=2.3)
```

Se pide:

1. ¿Cuántos valores de x son mayores o iguales de 39.4?
2. ¿Cuántos valores de x son estrictamente menores que 46?
3. ¿Cuántos valores de x son estrictamente menores que 46 y mayores o iguales de 39.4?
4. ¿Cuántos valores son tales que su parte entera (por defecto) es igual a 40? Se puede utilizar también la función base::floor.

[148] Estamos generando valores con distribución normal con media 45 y desviación estándar 2.3. Simplemente son unos datos para trabajar con ellos.

22.5 Varias muestras

Obviamente no tendremos habitualmente una sola muestra sino que tendremos más de una muestra. En cada una de ellas tendremos observada la expresión de los mismos genes. Supongamos que tenemos la expresión 10 genes en 4 muestras. Simulamos estos datos.

```
set.seed(123)
(x = matrix(rpois(10*4,lambda=5),nrow=10))
```

```
     [,1] [,2] [,3] [,4]
 [1,] 4 9 8 9
 [2,] 7 5 6 8
 [3,] 4 6 6 6
 [4,] 8 5 11 7
 [5,] 9 2 6 1
 [6,] 2 8 6 5
```

```
 [7,] 5 3 5 6
 [8,] 8 2 5 3
 [9,] 5 4 4 4
[10,] 5 9 3 3
```

¿Qué tipo de dato es x?

```
class(x)
```

```
[1] "matrix" "array"
```

Es una matriz. Podemos darle nombre a sus filas y columnas.

```
colnames(x) = c("sample1","sample2","sample3","sample4")
```

[149] El siguiente código es equivalente al anterior y más simple.

```
colnames(x) = paste0("sample",1:4)
```

[149] La función base::c simplemente construye un vector y en cada posición tiene el nombre de la muestra. Podemos concatenar también números.

Además sabemos que en cada fila tenemos la expresión correspondiente a un gen distinto. De momento, supondremos que son "gene1", "gene2", etc. Vamos a cambiarle el nombre las filas utilizando el código que acabamos de ver.

```
rownames(x) = paste0("gene",1:10)
```

¿Qué tenemos ahora?

```
x
```

```
      sample1 sample2 sample3 sample4
gene1 4 9 8 9
gene2 7 5 6 8
gene3 4 6 6 6
gene4 8 5 11 7
gene5 9 2 6 1
gene6 2 8 6 5
gene7 5 3 5 6
gene8 8 2 5 3
gene9 5 4 4 4
gene10 5 9 3 3
```

Podemos ver la expresión del gen en la fila 6 y en la columna 2 con

```
x[6,2]
```

```
[1] 8
```

o bien con

```
x["gene6","sample2"]
```

```
[1] 8
```

Si queremos ver la expresión de los genes en la muestra 2 podemos hacerlo con

```
x[,2]
```

```
 gene1 gene2 gene3 gene4 gene5 gene6 gene7
     9 5 6 5 2 8 3
 gene8 gene9 gene10
     2 4 9
```

o

```
x[,"sample2"]
```

```
gene1 gene2 gene3 gene4 gene5 gene6 gene7
    9 5 6 5 2 8 3
gene8 gene9 gene10
    2 4 9
```

Ex. 40 — Ejecutad el siguiente código. Nos genera una matriz donde los valores siguen una distribución normal con media 34 y desviación estándar 3.21.

```
x = matrix(rnorm(2345*122,mean=34,sd=3.21),nrow=2345)
```

Se pide:

1. ¿Cuántas columnas tiene la matriz?
2. ¿Qué columna tiene la media más alta? Utilizad las funciones base::apply y base::which.max.
3. ¿Qué fila tiene la media más pequeña? Utilizad las funciones base::apply y base::which.min.
4. ¿Qué fila tiene el menor rango intercuartílico? Utilizad las funciones base::apply y base::which.min.
5. ¿En cuántas ocasiones la primera columna es mayor o igual que la segunda columna de la matriz?
6. Determinamos la media de cada una de las filas. ¿Cuántas filas son tales que la columna 45 es mayor o igual que el vector de medias que acabamos de calcular?

22.6 Análisis de los datos golub

Los datos golub son banco de datos de expresión de genes que vamos a utilizar. Aparecen en multtest::golub y fueron utilizados en [63]. Los datos son los niveles de expresión de 3051 genes (que aparecen en filas) para 38 pacientes de leucemia. De estos pacientes, 27 de ellos tienen leucemia linfoblástica aguda (ALL) y los restantes 11 tienen leucemia mieloide aguda (AML). El tipo de tumor viene indicado por el vector `golub.cl` donde ALL corresponde con 0 y AML corresponde con 1. Los nombres de los genes los tenemos en `golub.gnames`. Cargamos los datos.

```
data(golub,package = "multtest")
```

Al cargar estos datos hemos leído varias cosas:

golub.cl Es un vector numérico que toma valores 0 y 1 indicando si la muestra ha sido obtenida de un enfermo con leucemia linfoblástica aguda (0) o leucemia mioloide aguda (1).

golub Es una matriz de expresión donde cada fila corresponde con un gen y cada columna con una muestra distinta.

golub.gnames Es una matriz con el mismo número de filas de la matriz golub de modo que la fila i-ésima de golub.gnames nos da información sobre el gen de cuyos niveles de expresión aparecen en la fila i-ésima de la matriz golub.

Empezamos jugando con el vector `golub.cl`. Podemos ver los valores.

```
golub.cl
```

```
 [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
[24] 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
```

¿Qué tipo de dato tenemos?

```
class(golub.cl)
```

```
[1] "numeric"
```

Vemos que son números. Son números que, realmente, indican la pertenencia a un tipo de cáncer u otro. Por ello es más natural definirlo como un vector tipo `factor`.[150] Lo hacemos con base::factor.

[150] Como en R codificamos las variables categóricas.

```
(golub.fac = factor(golub.cl,levels=0:1,labels=c("ALL","AML")))
```

```
 [1] ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL
[12] ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL ALL
[23] ALL ALL ALL ALL ALL AML AML AML AML AML AML
[34] AML AML AML AML AML
Levels: ALL AML
```

¿Cuántas muestras tenemos de cada tipo? Un par de opciones para hacerlo son utilizando la función base::table.

```
table(golub.fac)
```

```
golub.fac
ALL AML
 27 11
```

o utilizando la función genérica `summary()`.

```
summary(golub.fac)
```

```
ALL AML
 27 11
```

Si queremos mostrar la distribución de los dos tipos en forma de diagrama de barras lo podemos hacer con (ver figura 22.1)

```
pacman::p_load(ggplot2)
df0 = data.frame(golub.fac)
png(paste0(dirTamiFigures,"RBio39.png"))
ggplot(df0,aes(x=golub.fac))+geom_bar()
dev.off()
```

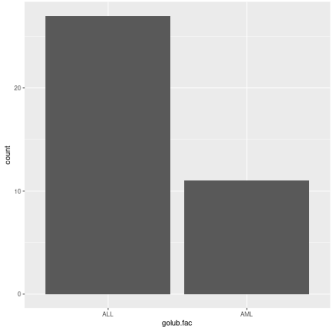


Figura 22.1: Diagrama de barras con las frecuencias absolutas de cada tipo de leucemia.

Sustituye en el código anterior `golub.fac` por `golub.cl` y verás que el resultado no es el deseable. ¿Por qué? Quizás los números no es una buena forma de representar categorías o clases.

Nos fijamos en `golub`. ¿Qué tipo de dato es?

```
class(golub)
```

```
[1] "matrix" "array"
```

Es una matriz. En una matriz lo primero es saber cuántas filas (características aquí) tenemos

```
nrow(golub)
```

```
[1] 3051
```

El número de columnas corresponde con el número de muestras.

```
ncol(golub)
```

```
[1] 38
```

También podemos obtener las dimensiones de la matriz conjuntamente con

```
dim(golub)
```

```
[1] 3051 38
```

En la fila i y columna j tiene una cuantificación de la expresión del gen i-ésimo[151] en la j-ésima muestra de nuestra experimentación. Por ejemplo, el gen en fila 2000 y columna 12 tiene expresión

[151] Ya veremos más adelante de qué gen estamos hablando.

```
golub[2000,12]
```

```
[1] 0.19595
```

[152] Y todos los niveles de este gen lo tendremos con

[152] Estos datos han sido preprocesados partiendo de los datos originales. Pueden aparecer valores negativos. En § 2.7.1 se explica y se entiende el porqué del valor negativo. Lo importante valor menor indica menor expresión, valor mayor indica mayor expresión.

```
golub[2000,]
```

```
 [1] 0.73124 -0.19598 0.51981 -0.54371 0.55596
 [6] 1.40683 0.79772 0.59493 0.99503 0.39529
[11] 0.09834 0.19595 0.85017 -1.39979 1.09789
[16] -0.74362 0.44207 0.27698 -0.04128 -1.60767
[21] -1.06221 -1.12665 0.47863 -0.44014 0.22286
[26] 0.42795 0.65427 0.07257 -0.28093 -0.20985
[31] 0.05160 -1.44434 -0.17118 -1.34158 0.92325
[36] -0.21462 -1.34579 1.17048
```

Estos valores reciben el nombre de *perfil de expresión*.[153] Podemos representar el perfil de expresión (figura 22.2). Definimos qué van en el eje de abscisas y en el eje de ordenadas.

[153] Expression profile.

```
pacman::p_load(ggplot2)
muestra = 1:ncol(golub) ## En abscisas el número de la muestra
y2000 = golub[2000,] ## En ordenadas su expresión
df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,])
png(paste0(dirTamiFigures,"RBio47.png"))
ggplot(df,aes(x = muestra,y= y2000))+geom_point()
dev.off()
```

Lo podemos ver en figura 22.2(a).

Si queremos ver las expresiones correspondientes a los pacientes ALL lo podemos hacer con [154]

[154] Observermos que "==" se utiliza para comparar e indica TRUE O FALSE según sea cierta o falsa la condición.

```
golub[2000,golub.fac == "ALL"]
```

```
 [1] 0.73124 -0.19598 0.51981 -0.54371 0.55596
 [6] 1.40683 0.79772 0.59493 0.99503 0.39529
[11] 0.09834 0.19595 0.85017 -1.39979 1.09789
[16] -0.74362 0.44207 0.27698 -0.04128 -1.60767
[21] -1.06221 -1.12665 0.47863 -0.44014 0.22286
[26] 0.42795 0.65427
```

y los correspondientes a los pacientes AML.

```
golub[2000,golub.fac == "AML"]
```

```
 [1] 0.07257 -0.28093 -0.20985 0.05160 -1.44434
 [6] -0.17118 -1.34158 0.92325 -0.21462 -1.34579
[11] 1.17048
```

Supongamos que volvemos a representar el perfil de expresión del gen en la fila 2000, y2000, pero pretendemos diferenciar con un color distinto la condición que nos indica el tipo de leucemia, golub.fac. Lo podemos hacer con (figura 22.2(b)):

```
df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],
                tipo = golub.fac)
png(paste0(dirTamiFigures,"RBio53.png"))
ggplot(df,aes(x = muestra,y= y2000,color=tipo))+geom_point()
dev.off()
```

Otra opción sería representar separadamente[155] los datos correspondientes a cada tipo de leucemia (figura 22.2(c)). También podemos hacerlo con

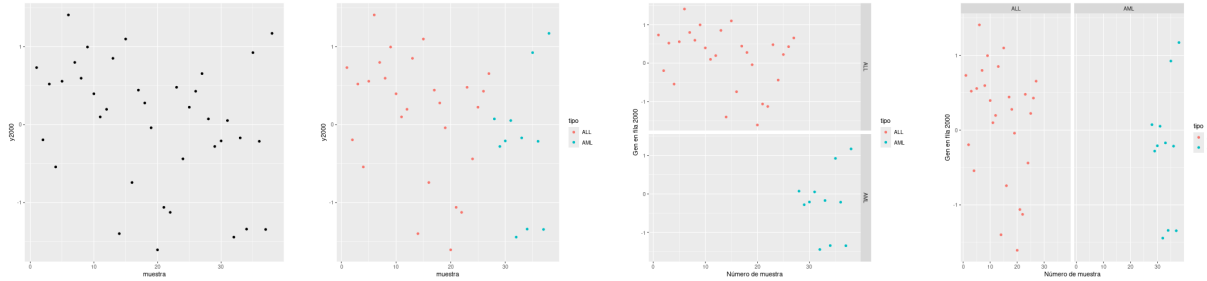
[155] En facets distintas.

```
df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],
                tipo = golub.fac)
png(paste0(dirTamiFigures,"RBio56.png"))
ggplot(df,aes(x = muestra,y= y2000,colour=tipo))+geom_point() +
    xlab("Número de muestra") + ylab("Gen en fila 2000") +
    facet_grid(tipo~.)
dev.off()
```

Podemos colocar los distintos subdibujos (o facets) horizontalmente (22.2(d)).

```
df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],
                tipo = golub.fac)
png(paste0(dirTamiFigures,"RBio57.png"))
ggplot(df,aes(x = muestra,y= y2000,colour=tipo))+geom_point() +
    xlab("Número de muestra") + ylab("Gen en fila 2000") +
    facet_grid(. ~tipo) #Modificamos esta línea
dev.off()
```

En gráficos (en todo) la simplicidad es un valor. De las tres figuras anteriores quizás la figura 22.2(b) sea la más adecuada.



(a) (b) (c) (d)

Figura 22.2: (a) Perfil de expresión del gen en la fila 2000. (b) Perfil de expresión del gen en fila 2000 diferenciando el tipo de leucemia. (c) Perfil de expresión del gen en fila 2000 diferenciando el tipo de leucemia mediante un color distinto. Utilizamos distintos dibujos (facets) alineados en vertical. (d) Perfil de expresión del gen en fila 2000 diferenciando el tipo de leucemia mediante un color distinto. Utilizamos distintos dibujos (facets) alineados en vertical.

¿De qué genes estamos hablando? ¿A qué gen corresponde esta fila? La información la podemos obtener con

```
golub.gnames[2000,]
```

¿Qué tipo de datos tenemos ahora?

```
class(golub.gnames)
```

```
[1] "matrix" "array"
```

Es una matriz con dimensiones

```
dim(golub.gnames)
```

```
[1] 3051 3
```

Cada fila nos da información con el gen del cual tenemos su perfil de expresión en la matriz golub. En concreto, ¿qué tenemos en la fila 2000?

```
golub.gnames[2000,]
```

```
[1] "4544" "CDC21 HOMOLOG"
[3] "X74794_at"
```

Los datos originales de la experimentación tenían más genes estudiados. Se hizo una selección (§ 2.7.1). Este gen ocupaba la fila 4544 en la matriz de expresión con todos los genes. Su nombre es

```
golub.gnames[2000,2]
```

```
[1] "CDC21 HOMOLOG"
```

El identificador de la sonda Affymetrix o AffyID (§ 2.2) ocupa la tercera columna y vale, para este gen,

```
golub.gnames[2000,3]
```

```
[1] "X74794_at"
```

Vamos a modificar la matriz `golub` de modo que recoja la información que tenemos de las filas (qué genes son) y de las columnas (qué tipo de leucemia). Para ello podemos aplicar las funciones **`rownames`** y **`colnames`** ↪ a dichas matrices.

1. Identificamos las filas con el identificador Affy.

   ```
   rownames(golub) = golub.gnames[,3]
   ```

2. Describimos las columnas con el tipo de leucemia.

   ```
   colnames(golub) = golub.fac
   ```

Tenemos la información de un modo más compacto.

22.7 La función apply

Vamos a explorar los datos golub utilizando la función base::apply. [156]

[156] Otras muy relacionadas que utilizaremos son base::lapply y base::sapply.

¿Qué pretendemos hacer? Vamos a realizar dos dibujos. En abscisas pretendemos dar una medida de localización (media o mediana) de la expresión del gen en el primer tipo de leucemia, ALL. En ordenadas lo mismo pero con el segundo tipo, AML.

Primero hemos de calcular estos valores. La función que calcula la media (muestral) de un vector de números es base::mean. Por ejemplo, la media del perfil de expresión del gen en la fila 2000 sería

```
mean(golub[2000,])
```

```
[1] 0.02080211
```

También parece una buena opción cuantificar el valor alrededor de donde se observan las expresiones mediante la mediana. La tenemos con

```
median(golub[2000,])
```

```
[1] 0.147145
```

Vemos que son muy distintos indicando asimetría.[157]

¿Y cómo lo hacemos para todos los genes de una vez? La función base::apply lo hace. Primero seleccionamos las partes de la matriz `golub` con las muestras `ALL` y con las muestras `AML`.

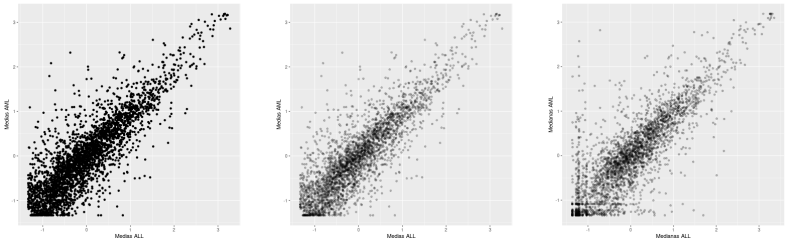
```
golub.ALL = golub[,golub.fac == "ALL"]
golub.AML = golub[,golub.fac == "AML"]
```

Ahora utilizamos base::apply. El primer argumento es la matriz con la que trabajamos, el segundo argumento indica si la función a aplicar es por filas (1) o por columnas (2). El tercer argumento es la función que queremos aplicar. Calculamos la media y la mediana para cada submatriz de golub.

```
ALL.mean = apply(golub[,golub.fac == "ALL"],1,mean)
AML.mean = apply(golub[,golub.fac == "AML"],1,mean)
ALL.median = apply(golub[,golub.fac == "ALL"],1,median)
AML.median = apply(golub[,golub.fac == "AML"],1,median)
```

Representamos las medias de ALL (abscisas) frente a las medias AML (ordenadas) (figura 22.3(a)).

```
df = data.frame(x0 = ALL.mean, y0 = AML.mean)
png(paste(dirTamiFigures,"RBio72.png",sep=""))
p = ggplot(df,aes(x=x0,y=y0))+geom_point()
p + xlab("Medias ALL") + ylab("Medias AML")
dev.off()
```



(a) (b) (c)

Figura 22.3: Medias en muestras AML frente a medias en muestras ALL para datos multtest::golub. En (a) tenemos los puntos tal cual y en (b) utilizamos un sombreado. (c) Medianas en muestras AML frente a medianas en muestras ALL para datos multtest::golub. sustituimos la media por la mediana.

En la figura 22.3(a) hay muchos puntos representados. Es un scatterplot o diagrama de puntos muy denso. Esto será habitual en lo que sigue pues trabajamos con grandes bancos de datos. Tantos puntos que se van superponiendo nos impide ver en qué zonas hay una mayor densidad de puntos. No es útil un dibujo así. Quizás ayude un sombreado que esté relacionado con la densidad local de puntos que representamos. Lo conseguimos con el argumento `alpha` (figura 22.3(b)).[158]

[157] Cuando para un conjunto de datos no hay coincidencia entre media y mediana lo que sucede es que no hay una disposición simétrica de los datos alrededor de estas medidas de localización.

[158] Este argumento alpha indicado como una fracción $1/4$ en el ejemplo indica que cuando se superpongan 4 puntos entonces se representa el punto completamente negro. Por debajo se utilizan

```
png(paste0(dirTamiFigures,"RBio73.png"))
ggplot(df,aes(x=x0,y=y0))+geom_point(alpha=.25)+xlab("Medias ALL") +
    ylab("Medias AML")
dev.off()
```

Reproducimos el dibujo sustituyendo la media por la mediana (figura 22.3(c)).

```
df = data.frame(x0 = ALL.median, y0 = AML.median)
png(paste(dirTamiFigures,"RBio75.png",sep=""))
p = ggplot(df,aes(x=x0,y=y0))+geom_point(alpha=.25)
p + xlab("Medianas ALL") + ylab("Medianas AML")
dev.off()
```

* **Ex. 41 —** Utilizando la matriz de expresión de los datos `golub` se pide:

1. Calcular para cada gen la expresión media sin considerar el tipo de cáncer.
2. Calcular para cada gen la desviación estándar.
3. Representar gráficamente un dibujo que, para cada gen, nos muestre en abscisas la expresión media y en ordenadas la desviación estándar.
4. Utilizando la función grDevices::png guardar el dibujo anterior en un fichero externo.

* **Ex. 42 —** Con los datos golub se pide:

1. Calcular la expresión media para cada una de las condiciones definidas por la variable golub.cl.
2. Representar una media frente a la otra.
3. Con la función *abline()* añadir la recta $y = x$ así como las rectas $y - x = \delta$ e $y - x = -\delta$ donde δ es un valor que iremos variando.
4. ¿Qué nos indicarían los genes tales que sus puntos asociados están fuera de la región $\{(x, y) : |y - x| > \delta\}$.

* **Ex. 43 —** Consideremos los datos `golub`. Se pide:

1. Determinar la desviación estándar de las expresiones de cada gen.
2. Representar un histograma de estas desviaciones estándar.
3. Calcular el percentil de orden 0.9 de las desviaciones calculadas en el punto anterior. Denotemos este valor por $q_{0.9}$.
4. Utilizando la función *abline* añadir al dibujo del apartado 2 una línea vertical cuya abscisa coincida con el valor $q_{0.9}$.
5. Seleccionar aquellos genes cuya desviación estándar sea mayor que el valor $q_{0.9}$ del punto anterior.

* **Ex. 44 —** El siguiente código genera una matriz y la guarda en `x`.

```
set.seed(123)
x = matrix(runif(500),nrow=100)
```

Ejecuta en código anterior. Las preguntas que siguen se refieren a la matriz `x`.

1. ¿Cuántas columnas tiene la matriz anterior?
2. Calcula los valores máximos en cada una de las columnas de `x`.

3. Calcula los valores mínimos de las filas de `x`.
4. ¿En qué fila tenemos el mínimo más pequeño de los mínimos por fila calculados en 3?
5. ¿En qué fila tenemos el mínimo más grande de los mínimos por fila calculados en 3?
6. Determina el percentil de orden 0.34 de la primera columna de la matriz `x`.
7. Calcular el percentil de orden 0.23 de las medias por fila de la matriz `x`.
8. ¿Cuál es el mínimo, máximo y valor medio de `x`?
9. Representar un histograma de los datos de la primera columna de `x` utilizando [150, ggplot2].
10. Representar un estimador kernel de la densidad de los datos de la primera columna de `x` utilizando [150, ggplot2].
11. ¿Cuántos elementos de la matriz `x` son mayores o iguales a 0.12?
12. Para cada fila de la matriz `x` calcula cuántos valores son menores o iguales a 0.12.
13. De los valores calculados en 12 determina el rango intercuartílico.
14. ¿Cuántas filas son tales que la primera columna es mayor o igual a la segunda?

13.** **Ex. 45 —** Construir un `data.frame` utilizando los datos multtest::golub que tenga las siguientes caracteríscas.

1. Cada variable (columna) corresponda a una sonda y cada fila corresponda con una muestra distinta.
2. Los nombres de las variables han de ser el nombre "probe1", "probe2", etc.
3. Añadir una variable (que llamaremos `type` que indique el grupo de leucemia al que pertenece cada muestra. Esta variable ha de ser de tipo `factor` y las etiquetas (o `labels`) han de ser `AML` (leucemia mieloide aguda) y `ALL` (leucemia linfoblástica aguda).

[159]

[159] En lo que sigue veremos una clase llamada ExpressionSet mucho más adecuada que el data.frame para este tipo de datos.

22.8 Listas

Las listas son muy importantes en R. En este manual las usaremos, entre otras cosas, cuando trabajamos con grupos de genes. Supongamos que estamos trabajando con un grupo de 10 genes y los representamos con un número. Por tanto nuestro universo de genes es el conjunto $\{1, \ldots, 10\}$. De estos diez genes suponemos que se ha descrito asociación para los siguientes grupos:

```
geneset1 = c(1,4)
geneset2 = c(2,5,7,8)
geneset3 = c(1,5,6)
```

¿Cómo almacenamos esta información? Lo mejor es una lista. La empezamos creando.

```
(genesets = vector("list",3))
```

```
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL
```

De momento la tenemos vacía. Hacemos que el primer elemento de la lista contenga al primer conjunto.

```
genesets[[1]] = geneset1
```

También para el segundo y tercero.

```
genesets[[2]] = geneset2
genesets[[3]] = geneset3
```

Podemos ver lo que tenemos ahora.

```
genesets
```

```
[[1]]
[1] 1 4

[[2]]
[1] 2 5 7 8

[[3]]
[1] 1 5 6
```

Parece natural darle un nombre a cada elemento. Supongamos que pretendemos llamarlos `set1`, `set2` y `3` respectivamente.

```
names(genesets) = paste0("set",1:3)
```

Los genes tienen distintas denominaciones como vemos más adelante. De momento, nos conformamos con llamarlos `gene1`, etc. Construimos un vector de nombres.

```
genenames = paste0("gene",1:10)
```

Convertimos los elementos de la lista utilizando los nombres en lugar de las posiciones. Por ejemplo, para el primero elemento sería

```
genesets[[1]] = genenames[geneset1]
```

Y lo mismo para los otros.

```
genesets[[2]] = genenames[geneset2]
genesets[[3]] = genenames[geneset3]
```

Ya tenemos una lista con los grupos de genes descritos.

22.9 Funciones con R

No podemos trabajar con R sin utilizar funciones.[160] La mayor potencia de R es que podemos definir funciones para extenderlo.

[160] Es un lenguaje funcional.

La estructura básica de una función en R es

```
nombre.funcion = function(argumentos){
 CODIGO
 resultado
}
```

Supongamos que queremos calcular la media y desviación estándar de un vector de datos. La siguiente función puede valer.

```
MediaDesviacion = function(x){
  resultado = c(mean(x),sd(x))
  resultado
}
```

La función devuelve lo que tiene en la última línea.[161] Una vez le declaramos la función podemos usarla.[162] La aplicamos a datos generados con distribución normal.

[161] También podemos usar *return*.

[162] Copiamos y pegamos en línea de comandos.

```
x = rnorm(23,mean=12,sd=1.3)
MediaDesviacion(x)
```

```
[1] 12.274904 1.278401
```

Queremos calcular la media y desviación estándar pero de los datos que están por encima del percentil de orden p. El valor de p será un *argumento* y queremos fijarlo por defecto en 0.5.

```
MediaDesviacionPercentil = function(x,ordenpercentil = 0.5){
  qp = quantile(x,probs=ordenpercentil)
  x0 = x[x >qp]
  resultado = c(mean(x0),sd(x0))
  resultado
}
```

Y la utilizamos.

```
MediaDesviacionPercentil(x)
```

```
[1] 13.3144744 0.8567743
```

Si queremos cambiar el valor por defecto del orden del percentil hacemos

```
MediaDesviacionPercentil(x,ordenpercentil=.7)
```

```
[1] 13.7880029 0.6425422
```

No hemos comprobado que efectivamente es un vector. Además pretendemos que no nos devuelva nada si no lo que recibe no es un vector. En este caso que nos avise.

```
MediaDesviacionPercentilError = function(x,ordenpercentil = 0.5){
  if(!is.vector(x)) return("No es un vector")
  qp = quantile(x,probs=ordenpercentil)
  x0 = x[x >qp]
  resultado = c(mean(x0),sd(x0))
  resultado
}
```

Vamos a ver qué tal funciona. Definimos una matriz.

```
y = matrix(1:16,ncol=4)
```

Comprobamos que no es un vector.

```
is.vector(y)
```

```
[1] FALSE
```

Y probamos nuestra función.

```
MediaDesviacionPercentilError(y)
```

```
[1] "No es un vector"
```

Funciona y todo. No obstante queda un poco feo que cuando es un vector simplemente nos devuelve otro vector y hemos de saber que es una media lo primero y una desviación lo segundo. Podemos devolver una lista de modo que le demos un nombre a cada cosa.

```
MediaDesviacionPercentilErrorLista = function(x,ordenpercentil = 0.5){
  if(!is.vector(x)) return("No es un vector")
  qp = quantile(x,probs=ordenpercentil)
  x0 = x[x >qp]
  resultado = list(media = mean(x0),de = sd(x0))
  resultado
}
```

Y la probamos.

```
MediaDesviacionPercentilErrorLista(x)
```

```
$media
[1] 13.31447

$de
[1] 0.8567743
```

Por ello podemos guardar lo que nos devuelve y luego acceder a cada elemento de la lista.

```
x.des = MediaDesviacionPercentilErrorLista(x)
```

Y tendremos que la media es

```
x.des$media
```

```
[1] 13.31447
```

y la desviación estándar sería

```
x.des$de
```

```
[1] 0.8567743
```

** **Ex. 46 —** Dado un vector `x` programar una función que nos devuelva en una lista la media y la varianza del vector. Los elementos de la lista se han de llamar `media` y `varianza`.

** **Ex. 47 —** Programa una función que tenga como argumentos un vector `x` y un valor numérico `alpha` entre 0 y 1. Eliminar el `alpha` por uno de los más pequeños y la misma cantidad de los más grandes. De lo que queda ha de devolver, en forma de lista, la media y varianza con los nombres `media_ajustada` y `varianza_ajustada`.

22.10 Pasar argumentos a un script de R por línea de comandos

[163] Luis Orduña me mostró cómo hacerlo.

[163] El script de R debe comenzar de la siguiente manera:

```
args = commandArgs(trailingOnly=TRUE)
if (length(args)==0) {
  stop("At least one argument must be supplied (path to the folder
that contains raw_data).", call.=FALSE)
}
```

Para ejecutar el script pasándole argumentos simplemente hay que hacer

```
Rscript --vanilla script.R argument
```

Se puede encontrar información adicional en https://www.r-bloggers.com/passing-arguments-to-an-r-script-from-command-lines/.

22.11 tamitasks

¿Cómo hacer un paquete en la medida en que vamos desarrollando un proyecto? ¿Cómo hacer un paquete sin complicarse mucho la vida? Trabajar con paquetes en R es muy importante. Permite organizar y comprobar que estamos realizando un trabajo correctamente. Desde el principio hay que aprender a construir nuestro propio paquete. De esto hablamos aquí. Lo que comentamos en esta sección está ejemplificado en el paquete `tamitasks` que podemos encontrar en http://www.uv.es/ayala/docencia/tami/tamitasks. Una muy buena referencia sobre cómo construir un paquete es [146]. Lo que sigue es una breve forma de construir un paquete muy sencillo en Linux.[164]

[164] He de reconocer que no tengo el más mínimo interés en saber cómo se hace en Windows. No al menos en esta vida.

1. Hemos de crear un directorio con el nombre del paquete.

```
mkdir tamitasks
```

2. Luego creamos directorios dentro de `tamitasks`.

```
cd tamitasks
mkdir R
mkdir data
mkdir man
mkdir doc
mkdir vignettes
```

3. Creamos en `tamitasks` el fichero `DESCRIPTION`. Un posible ejemplo puede ser

```
Package: tamitasks
Type: Package
Title: tamitasks
Version: 0.2
Date: "`r Sys.Date()`"
Authors@R: person("Guillermo","Ayala",email="Guillermo.Ayala@uv.es",
                  role = c("aut","cre"))
Description: A (very) basic package
Imports:
```

```
    AnnotationDbi,
    annotate,
    affy,
    Biobase,
    DBI,
    edgeR,
    genefilter,
    ggfortify,
    ggplot2,
    gridExtra,
    GSA,
    GSEABase,
    GEOquery,
    hgu133plus2.db,
    limma,
    MASS,
    matrixStats,
    methods,
    multtest,
    pacman,
    pbapply,
    qvalue,
    ReportingTools,
    reshape2,
    samr,
    stats,
    SummarizedExperiment,
    tamidata
Suggests: knitr,
          rmarkdown
URL: http://www.uv.es/ayala/docencia/tami
VignetteBuilder: knitr
License: GPL-2
LazyLoad: yes
RoxygenNote: 7.1.0
```

4. Copiamos en `data` los ficheros binarios de datos. En mi caso son los ficheros `geod71810.rda` y `PRJNA297664.rda`.

5. Copiamos en `vignettes` las viñetas. En mi caso son las viñetas en donde se muestra cómo se han construido los datos del punto anterior. Son las viñetas `geod71810.Rmd` y `PRJNA297664.Rmd`.

6. Copiamos en el subdirectorio `tamitasks/R` los ficheros con el código en R. En mi caso son los ficheros `tasks-data.R` y `tasks-functions.R`. En `tasks-data.R` se muestra cómo definir datos para poder cargarlos.

```
#' @title
PRJNA297664
@description
Saccharomyces cerevisiae
Expression profiling by high throughput
    ↪ sequencing
Identification of the difference in transcriptome
    ↪  between wild-type and
```

```
SEC66 deletion mutant cells
@source
\url{http://www.ncbi.nlm.nih.gov/bioproject/
    ↪ 297664}
\url{http://www.ncbi.nlm.nih.gov/geo/query/acc.
    ↪ cgi?acc=GSE73681}
The procedure to produce the counts per gene from
    ↪  the original sra files
can be found at
\url{http://www.uv.es/ayala/docencia/tami/
    ↪ CaseStudies/PRJNA297664.html}
@examples
data(PRJNA297664,package="tamitasks")
@docType data
@keywords datasets
@format SummarizedExperiment
@name PRJNA297664
NULL


geod71810
@description
ExpressionSet \code{geod71810}
@examples
data(geod71810,package="tamitasks")
@docType data
@keywords datasets
@format ExpressionSet
@name geod71810
NULL
```

En `tasks-functions.R` tenemos un par de ejemplos de funciones.

```
@title Trapezoid rule for numerical integration
@description
An implementation of the trapezoid rule for
    ↪ numerical integration
@param x Abscisas
@param y Function values at \code{x}
@export
TrapezoidRule = function(x,y){
  idx = 2:length(x)
  return (as.double( (x[idx] - x[idx-1]) %*% (y[
      ↪ idx] + y[idx-1])) / 2)
}

URL's from entrez identifiers
 @description
 It makes the url's from entrez identifiers
@param id ENTREZ identifiers
@return
The corresponding URL's in the database
 @export
 @family URL generation
```

```
entrezid2url = function(id)
ifelse(id == "NA",NA,
paste("<a href='http://www.ncbi.nlm.nih.gov/gene
↪ /?term=",
id,"'>",id,"</a>",sep=""))
```

Como podemos ver en los ejemplos anteriores documentamos el paquete utilizando el paquete [151].

7. Ya tenemos hecho el paquete y hemos de construirlo. Empezamos con algo de código de fácil comprensió.

```
pkg = "tamitasks"
version = "0.2"
dirbasepkg = "/home/gag/ownCloud/alltami/"
dirpkg = paste0(dirbasepkg,pkg)
dirvignettes = paste0(dirbasepkg,pkg,"-notes/
    ↪ vignettes/")
pkgtar = paste0(pkg,"_",version,".tar.gz")
```

8. Cargamos paquetes necesarios.

```
pacman::p_load(devtools,pkgdown)
```

9. Fijamos directorio para construir el paquete.

```
setwd(dirbasepkg)
```

10. Generamos la documentación con [151].

```
devtools::document(pkg)
```

11. Vemos si tenemos algún error.

```
devtools::check(pkg)
```

12. Y construimos el paquete. Tenemos dos opciones:

    (a) Utilizando `devtools::build()`.

    ```
    devtools::build(pkg)
    ```

    (b) Directamente utilizando la línea de comandos.

    ```
    R CMD build --resave-data tamitasks
    ```

13. Instalamos el paquete.

```
install.packages(paste0(pkg,"_",version,".tar.gz"
    ↪ ),repos=NULL,source=TRUE)
```

14. Generamos la página web asociada. Queda bonito.

```
library(tamitasks)
setwd(dirpkg)
pkgdown::build_site()
```

15. Si después de todo lo previo funciona has tenido suerte. Si no empieza desde el primer punto y mira en qué te has equivocado. Creo que esto es la (Bio)Informática.

22.12 Actualizar paquetes

Es frecuente que tengamos que actualizar todos los paquetes que tenemos instalados. Con el siguiente código lo podemos hacer.

```
install.packages(
    lib = lib <-.libPaths()[1],
    pkgs = as.data.frame(installed.packages(lib),
                         stringsAsFactors=FALSE)$Package,type = 'source'
)
```

Capítulo 23

Bioconductor

En este curso tan (o más) importante que el propio R es el proyecto Bioconductor. Básicamente es una colección de paquetes de R para Bioinformática.

Para instalar paquetes de Bioconductor necesitamos el paquete [102, BiocManager]. Se instala con (la versión hay que actualizarla)

```
if (!require("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install(version = "3.20") ##Revisar cuál es la versión actualizada
```

Una vez instalado la parte básica de Bioconductor podemos instalar paquetes adicionales como [115, multtest] con

```
BiocManager::install("multtest")
```

Podemos elegir el repositorio desde el cual instalamos los paquetes de Bioconductor con

```
chooseBioCmirror()
```

También podemos elegir interactivamente el repositorio con

```
setRepositories()
```

Bioconductor y CRAN tienen muchísimos paquetes y con frecuencia encontrar los que nos pueden interesar no es nada fácil. La función `BiocManager` ↪ `::available()` es útil. Por ejemplo si buscamos paquetes que tengan que ver con Agilent podemos hacer

```
BiocManager::available(pattern = "Agilent",include_installed = TRUE)
```

Capítulo 24

Anotación

¿De qué variables hablamos? En este manual trabajamos con medidas de abundancia entendida como expresión de un gen o presencia de una mayor cantidad de proteína. En este contexto cuando hablamos de variables utilizaremos con frecuencia el nombre de características. Estas son nuestras variables. Pero, repetimos, ¿de qué variables hablamos? En ocasiones nos estaremos refiriendo a expresión de un gen cuantificado via microarrays o RNA-Seq o alguna otra técnica. En otras ocasiones hablaremos de exones o bien, cuando sea pertinente, de isoformas del gen obtenidas con empalmes alternativos.[165] Por tanto es de gran importancia[166] poder manejar para el organismo que nos ocupe bases de datos que nos permitan conocer cómo denominar a las características: distintos identificadores de genes, exones, isoformas, proteínas. Además cómo pasar de unos identificadores a otros. Cuáles son las correspondencias entre ellos que con frecuencia no son correspondencias 1-1. Para cada organismo podemos encontrar bases de datos disponibles online en la red. Algunas de ellas se ocupan de más de un organismo. Hay bases de datos específicas de cierto tipo de genes (como de microRNAs). Es habitual en la práctica del investigador acudir y usar este tipo de bases de datos con el gran hallazgo informático de copiar y pegar. Es un trabajo tedioso y, posiblemente, innnesario. ¿Todo lo que hacemos en este tema se puede hacer de este modo? Diría que no. En este tema nos ocupamos de cómo acceder a esta información pero desde R/Bioconductor.[167]

Es muy conveniente consultar http://www.bioconductor.org/help/workflows/annotation/annotation/. Los paquetes de anotación de Bioconductor los tenemos en https://www.bioconductor.org/packages/release/data/annotation/.

[165] Alternative splicing.

[166] Estamos al principio del texto.

[167] El navegador para leer periódicos.

Tipos de paquetes de anotación. Tenemos diferentes tipos de paquetes de anotación:

ChipDb Se refieren a una plataforma concreta. Por ejemplo, un chip de microarray. § 24.2

OrgDb Centrados en el organismo.

TxDb Paquetes relativos a transcriptomas de un organismo.

BSgenome Paquetes con genomas.

[168]

[168] En lo que sigue seguimos el flujo de trabajo annotation de Bioconductor.

24.1 AnnotationDbi

Las bases de datos de tipo `ChipDb`, `OrgDb` y `TxDb` heredan todos los métodos de la clase `AnnotationDb` que está definida en [113, AnnotationDbi]. Por ello los métodos aplicables a `AnnotationDb` son aplicables a las demás: `columns`, `keytypes`, `keys` y `select`. Con estos métodos podremos extraer información de las bases de las datos (objetos de clase `ChipDb`, `OrgDb` y `TxDb`) correspondientes. Previo al uso de las distintas bases de datos de anotación es pues conveniente conocer los objetos `AnnotationDb`.

```
pacman::p_load("AnnotationDbi")
```

Para ilustrar los métodos vamos a considerar un paquete de tipo `ChipDb`, en concreto [30, hgu133a.db].

```
pacman::p_load("hgu133a.db")
```

La información contenida la podemos ver con

```
ls("package:hgu133a.db")
```

```
 [1] "hgu133a" "hgu133a_dbconn"
 [3] "hgu133a_dbfile" "hgu133a_dbInfo"
 [5] "hgu133a_dbschema" "hgu133a.db"
 [7] "hgu133aACCNUM" "hgu133aALIAS2PROBE"
 [9] "hgu133aCHR" "hgu133aCHRLENGTHS"
[11] "hgu133aCHRLOC" "hgu133aCHRLOCEND"
[13] "hgu133aENSEMBL" "hgu133aENSEMBL2PROBE"
[15] "hgu133aENTREZID" "hgu133aENZYME"
[17] "hgu133aENZYME2PROBE" "hgu133aGENENAME"
[19] "hgu133aGO" "hgu133aGO2ALLPROBES"
[21] "hgu133aGO2PROBE" "hgu133aMAP"
[23] "hgu133aMAPCOUNTS" "hgu133aOMIM"
[25] "hgu133aORGANISM" "hgu133aORGPKG"
[27] "hgu133aPATH" "hgu133aPATH2PROBE"
[29] "hgu133aPFAM" "hgu133aPMID"
[31] "hgu133aPMID2PROBE" "hgu133aPROSITE"
[33] "hgu133aREFSEQ" "hgu133aSYMBOL"
[35] "hgu133aUNIPROT"
```

Con el nombre del paquete también tenemos información.

```
hgu133a.db
```

```
ChipDb object:
| DBSCHEMAVERSION: 2.1
| Db type: ChipDb
| Supporting package: AnnotationDbi
| DBSCHEMA: HUMANCHIP_DB
| ORGANISM: Homo sapiens
| SPECIES: Human
| MANUFACTURER: Affymetrix
| CHIPNAME: Affymetrix HG-U133A Array
| MANUFACTURERURL: http://www.affymetrix.com
| EGSOURCEDATE: 2021-Apr14
| EGSOURCENAME: Entrez Gene
| EGSOURCEURL: ftp://ftp.ncbi.nlm.nih.gov/gene/DATA
| CENTRALID: ENTREZID
| TAXID: 9606
| GOSOURCENAME: Gene Ontology
| GOSOURCEURL: http://current.geneontology.org/ontology/go-basic.obo
| GOSOURCEDATE: 2021-02-01
| GOEGSOURCEDATE: 2021-Apr14
| GOEGSOURCENAME: Entrez Gene
```

```
| GOEGSOURCEURL: ftp://ftp.ncbi.nlm.nih.gov/gene/DATA
| KEGGSOURCENAME: KEGG GENOME
| KEGGSOURCEURL: ftp://ftp.genome.jp/pub/kegg/genomes
| KEGGSOURCEDATE: 2011-Mar15
| GPSOURCENAME: UCSC Genome Bioinformatics (Homo sapiens)
| GPSOURCEURL:
| GPSOURCEDATE: 2021-Feb16
| ENSOURCEDATE: 2021-Feb16
| ENSOURCENAME: Ensembl
| ENSOURCEURL: ftp://ftp.ensembl.org/pub/current_fasta
| UPSOURCENAME: Uniprot
| UPSOURCEURL: http://www.UniProt.org/
| UPSOURCEDATE: Mon Apr 26 21:53:12 2021
```

En los paquetes de anotación tenemos `columns`. Algunas de estas columnas pueden ser `keys`. Podemos realizar consultas en la base de datos utilizando una `key` y pedir que nos devuelva una o más de una `columns`. ¿Qué información podemos recuperar utilizando `select`?

```
columns(hgu133a.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

Pero: ¿qué información es? Lo obtenemos con[169]

```
help("ENTREZID")
```

[169] Podemos poner cualquiera de los nombres anteriores y nos saldrá la misma ayuda.

No todas las variables que hemos obtenido con `columns` son utilizables para realizar consultas. Aquellas utilizables para las consultas las podemos conocer con `keytypes`. A estas variables las llamamos llaves (`keys`).

```
keytypes(hgu133a.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

¿Cómo conseguir todas los valores de una llave determinada?

```
head(keys(hgu133a.db,keytype="ENTREZID"))
```

```
[1] "10" "100" "1000"
[4] "10000" "100008586" "10001"
```

```
head(keys(hgu133a.db,keytype="ENSEMBL"))
```

```
[1] "ENSG00000121410" "ENSG00000175899"
[3] "ENSG00000291190" "ENSG00000171428"
[5] "ENSG00000156006" "ENSG00000196136"
```

Supongamos que tenemos algunos identificadores Affy (AffyID) y pretendemos conocer sus identificadores ENTREZID. Empezamos eligiendo cinco identificadores Affy al azar.[170]

[170] De ahí el uso de la función base::sample.

```
(ids = sample(keys(hgu133a.db,keytype="PROBEID"),5))
```

```
[1] "205812_s_at" "220444_at" "222110_at"
[4] "214052_x_at" "205021_s_at"
```

Vamos a obtener sus identificadores Entrez, los de ENSEMBL y su SYMBOL.

```
df = AnnotationDbi::select(hgu133a.db,keys=ids,
                           columns=c("ENTREZID","ENSEMBL","SYMBOL"),
                           keytype="PROBEID")
```

```
data(gse20986,package="tamidata")
ids = featureNames(gse20986)
library(hgu133plus2.db)
df = AnnotationDbi::select(hgu133plus2.db,keys=ids,
                           columns=c("ENTREZID","ENSEMBL"),
                           keytype="PROBEID")
a = match(featureNames(gse20986),df[,"PROBEID"])
df1 = df[a,]
df1[1,]
```

```
   PROBEID ENTREZID ENSEMBL
1 1007_s_at 780 ENSG00000204580
```

```
dim(df1)
```

```
[1] 54675 3
```

```
dim(gse20986)
```

```
Features Samples
  54675 12
```

```
fData(gse20986) = df1
library("limma")
pData(gse20986)
```

```
                 tissue
GSM524662.CEL.gz iris
GSM524663.CEL.gz retina
GSM524664.CEL.gz retina
GSM524665.CEL.gz iris
GSM524666.CEL.gz retina
GSM524667.CEL.gz iris
GSM524668.CEL.gz choroides
GSM524669.CEL.gz choroides
GSM524670.CEL.gz choroides
GSM524671.CEL.gz huvec
GSM524672.CEL.gz huvec
GSM524673.CEL.gz huvec
```

```
design = model.matrix(~ 0 + pData(gse20986)[,"tissue"])
colnames(design) = levels(pData(gse20986)[,"tissue"])
design
```

```
  iris retina choroides huvec
1 1 0 0 0
2 0 1 0 0
3 0 1 0 0
4 1 0 0 0
5 0 1 0 0
6 1 0 0 0
7 0 0 1 0
8 0 0 1 0
9 0 0 1 0
10 0 0 0 1
11 0 0 0 1
12 0 0 0 1
attr(,"assign")
[1] 1 1 1 1
attr(,"contrasts")
attr(,"contrasts")$`pData(gse20986)[, "tissue"]`
[1] "contr.treatment"
```

```
(contrast.matrix = makeContrasts(dif12 = iris − retina,
                                 dif13 = iris − choroides,
                                 dif14 = iris − huvec
                                ,levels = design))
```

```
         Contrasts
Levels dif12 dif13 dif14
  iris 1 1 1
  retina -1 0 0
  choroides 0 -1 0
  huvec 0 0 -1
```

```
fit = lmFit(gse20986,design)
fit2 = contrasts.fit(fit,contrast.matrix)
fit3 = eBayes(fit2)
topTable(fit3,coef=1,number=2)
```

```
                                  PROBEID
AFFX-HUMRGE/M10098_3_at AFFX-HUMRGE/M10098_3_at
219273_at 219273_at
                     ENTREZID ENSEMBL
AFFX-HUMRGE/M10098_3_at <NA> <NA>
219273_at 8812 ENSG00000090061
                       logFC AveExpr
AFFX-HUMRGE/M10098_3_at 4.716983 10.696277
219273_at 2.696519 4.241303
                          t P.Value
AFFX-HUMRGE/M10098_3_at 19.69965 2.588982e-08
219273_at 17.96796 5.507795e-08
                      adj.P.Val B
AFFX-HUMRGE/M10098_3_at 0.001047691 9.974955
219273_at 0.001047691 9.202545
```

24.2 ChipDb

Si trabajamos con microarrays (§ 2) nuestras características serán las sondas.[171] Estas sondas tendrán un identificador que el fabricante del chip le ha asignado. Esto no es informativo para nosotros. Hemos de poder hacer corresponder este identificador con el gen al que corresponde. Estas bases de datos son de tipo `ChipDb`. Uno de los chips más populares de Affymetrix, Affymetrix Human Genome U133. El paquete Bioconductor con la anotación de este chip es [30, hgu133a.db]. Lo cargamos.

[171] Asociadas a genes.

```
pacman::p_load(hgu133a.db)
```

[172] Probes.

¿Qué sondas[172] tienen correspondencia en Entrez?

```
mappedProbes = mappedkeys(hgu133aENTREZID)
```

Lo guardamos en forma de lista.

```
mappedProbesList = as.list(hgu133aENTREZID[mappedProbes])
```

[173] AffyID.

[174] ENTREZID.

Por ejemplo, la primera posición de la lista nos da el identificador de Affymetrix[173] y su identificador Entrez[174].

```
mappedProbesList[1]
```

```
$`1007_s_at`
[1] "780"
```

O el correspondiente a la posición 4567.

```
mappedProbesList[4567]
```

```
$`205155_s_at`
[1] "6712"
```

Si hacemos

```
ls("package:hgu133a.db")
```

```
 [1] "hgu133a" "hgu133a_dbconn"
 [3] "hgu133a_dbfile" "hgu133a_dbInfo"
 [5] "hgu133a_dbschema" "hgu133a.db"
 [7] "hgu133aACCNUM" "hgu133aALIAS2PROBE"
 [9] "hgu133aCHR" "hgu133aCHRLENGTHS"
[11] "hgu133aCHRLOC" "hgu133aCHRLOCEND"
[13] "hgu133aENSEMBL" "hgu133aENSEMBL2PROBE"
[15] "hgu133aENTREZID" "hgu133aENZYME"
[17] "hgu133aENZYME2PROBE" "hgu133aGENENAME"
[19] "hgu133aGO" "hgu133aGO2ALLPROBES"
[21] "hgu133aGO2PROBE" "hgu133aMAP"
[23] "hgu133aMAPCOUNTS" "hgu133aOMIM"
[25] "hgu133aORGANISM" "hgu133aORGPKG"
[27] "hgu133aPATH" "hgu133aPATH2PROBE"
[29] "hgu133aPFAM" "hgu133aPMID"
[31] "hgu133aPMID2PROBE" "hgu133aPROSITE"
[33] "hgu133aREFSEQ" "hgu133aSYMBOL"
[35] "hgu133aUNIPROT"
```

podemos ver todas las correspondencias que nos ofrece el paquete. Consideremos unos identificadores Affymetrix.

```
ids = c("39730_at", "1635_at", "1674_at", "40504_at", "40202_at")
```

Se supone que el chip es `hgu95av2`. Cargamos el paquete de anotación correspondiente al chip utilizado [32, hgu95av2.db].

```
pacman::p_load("hgu95av2.db")
```

Y lo mismo que antes. Ahora tenemos además la correspondencia entre las sondas y los genes.

```
columns(hgu95av2.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

```
keytypes(hgu95av2.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

Los identificadores que teníamos eran los AffyID, esto es, los identificadores de las sondas utilizadas o PROBEID. Podemos plantearnos su correspondencia con el símbolo o nombre del gen y las proteínas asociadas en la base de datos PFAM.

```
columns = c("PFAM","SYMBOL")
AnnotationDbi::select(hgu95av2.db, keys=ids, columns, keytype="PROBEID")
```

24.3 OrgDb

Este tipo de paquete se refiere a un organismo dado y está centrado en el gen. Veamos como ejemplo el relativo a ser humano.

```
pacman::p_load(org.Hs.eg.db)
```

¿Qué tipo de cosas o qué claves podemos manejar?

```
columns(org.Hs.eg.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROSITE" "REFSEQ" "SYMBOL"
[25] "UCSCKG" "UNIPROT"
```

O bien con

```
keytypes(org.Hs.eg.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROSITE" "REFSEQ" "SYMBOL"
[25] "UCSCKG" "UNIPROT"
```

Por ejemplo, si queremos los primeros identificadores de genes utilizando los identificadores Entrez o ENTREZID tenemos

```
head(keys(org.Hs.eg.db, keytype="ENTREZID"))
```

```
[1] "1" "2" "3" "9" "10" "11"
```

Si consideramos sus identificadores en la base de datos Ensembl entonces podemos usar

```
head(keys(org.Hs.eg.db, keytype="ENSEMBL"))
```

```
[1] "ENSG00000121410" "ENSG00000175899"
[3] "ENSG00000291190" "ENSG00000171428"
[5] "ENSG00000156006" "ENSG00000196136"
```

Y en Gene Ontology.

```
head(keys(org.Hs.eg.db, keytype="GO"))
```

```
[1] "GO:0002764" "GO:0005576" "GO:0005615"
[4] "GO:0005886" "GO:0031093" "GO:0034774"
```

Supongamos que elegimos un sistema de identificación, por ejemplo, ENTREZID. En concreto, los cinco primeros genes.

```
(ids = keys(org.Hs.eg.db, keytype="ENTREZID")[1:5])
```

```
[1] "1" "2" "3" "9" "10"
```

A partir de estos identicadores podemos obtener el resto.

```
AnnotationDbi::select(org.Hs.eg.db,keys=ids,column="SYMBOL",
                      keytype='ENTREZID')
```

```
  ENTREZID SYMBOL
1 1 A1BG
2 2 A2M
3 3 A2MP1
4 9 NAT1
5 10 NAT2
```

Supongamos que nos fijamos en el gen con código Ensembl `ENSG00000000003` ↪ .

```
(id = "ENSG00000171428")
```

```
[1] "ENSG00000171428"
```

Buscamos su correspondencia en Gene Ontology.

```
(res = AnnotationDbi::select(org.Hs.eg.db, keys=id, column="GO",
                             keytype="ENSEMBL"))
```

```
        ENSEMBL GO EVIDENCE ONTOLOGY
1 ENSG00000171428 GO:0004060 IBA MF
2 ENSG00000171428 GO:0004060 TAS MF
3 ENSG00000171428 GO:0005515 IPI MF
4 ENSG00000171428 GO:0005829 TAS CC
5 ENSG00000171428 GO:0006805 TAS BP
```

Como vemos no tenemos una correspondencia 1-1. Al mismo gen le corresponden distintos términos Gene Ontology. Si solamente tenemos interés en ellos podemos hacer

```
res[,"GO"]
```

```
[1] "GO:0004060" "GO:0004060" "GO:0005515"
[4] "GO:0005829" "GO:0006805"
```

Puesto que tenemos identificadores Gene Ontology podemos utilizar el paquete [29, GO.db] para obtener los términos Gene Ontology correspondientes.

```
pacman::p_load(GO.db)
```

Y los términos Gene Ontology serían

```
AnnotationDbi::select(GO.db, keys=res[,"GO"], columns="TERM",
                      keytype="GOID")
```

24.4 TxDb

Los paquetes `TxDb` están centrados en el genoma. Vamos a trabajar con la drosophila melanogaster. https://en.wikipedia.org/wiki/Drosophila_melanogaster. Cargamos la base de datos.

```
library("GenomicFeatures")
library("TxDb.Dmelanogaster.UCSC.dm3.ensGene")
```

Al cargar este paquete lo que hemos hecho es leer un objeto que se llama como el propio paquete y de clase `TxDb`.[175]

[175] Todos los paquetes de Bioconductor que empiezan con TxDb son de este tipo.

```
class(TxDb.Dmelanogaster.UCSC.dm3.ensGene)
```

```
[1] "TxDb"
attr(,"package")
[1] "GenomicFeatures"
```

Hacemos una copia con un nombre más breve.

```
txdb = TxDb.Dmelanogaster.UCSC.dm3.ensGene
```

¿De qué clase es este objeto?

```
class(txdb)
```

```
[1] "TxDb"
attr(,"package")
[1] "GenomicFeatures"
```

Es pues un objeto de clase `TxDb`. Tenemos un resumen sobre los datos contenidos en `txdb` con

```
txdb
```

```
TxDb object:
# Db type: TxDb
# Supporting package: GenomicFeatures
# Data source: UCSC
# Genome: dm3
# Organism: Drosophila melanogaster
# Taxonomy ID: 7227
# UCSC Table: ensGene
# Resource URL: http://genome.ucsc.edu/
# Type of Gene ID: Ensembl gene ID
# Full dataset: yes
```

```
# miRBase build ID: NA
# transcript_nrow: 29173
# exon_nrow: 76920
# cds_nrow: 62135
# Db created by: GenomicFeatures package from Bioconductor
# Creation time: 2015-10-07 18:15:53 +0000 (Wed, 07 Oct 2015)
# GenomicFeatures version at creation time: 1.21.30
# RSQLite version at creation time: 1.0.0
# DBSCHEMAVERSION: 1.1
```

Podemos ver la información de la que disponemos con

```
columns(txdb)
```

```
 [1] "CDSCHROM" "CDSEND" "CDSID"
 [4] "CDSNAME" "CDSSTART" "CDSSTRAND"
 [7] "EXONCHROM" "EXONEND" "EXONID"
[10] "EXONNAME" "EXONRANK" "EXONSTART"
[13] "EXONSTRAND" "GENEID" "TXCHROM"
[16] "TXEND" "TXID" "TXNAME"
[19] "TXSTART" "TXSTRAND" "TXTYPE"
```

y con

```
keytypes(txdb)
```

```
[1] "CDSID" "CDSNAME" "EXONID" "EXONNAME"
[5] "GENEID" "TXID" "TXNAME"
```

Para el manejo de este tipo de bases de datos es útil [27, GenomicFeatures].

```
pacman::p_load(GenomicFeatures)
```

Por ejemplo: ¿Qué cromosomas tenemos?[176]

[176] En http://ucscbrowser.genenetwork.org/cgi-bin/hgGateway?hgsid=732&clade=insect&org=0&db=0 podemos encontrar una explicación detallada.

```
seqlevels(txdb)
```

```
 [1] "chr2L" "chr2R" "chr3L"
 [4] "chr3R" "chr4" "chrX"
 [7] "chrU" "chrM" "chr2LHet"
[10] "chr2RHet" "chr3LHet" "chr3RHet"
[13] "chrXHet" "chrYHet" "chrUextra"
```

Cuando se carga la base de datos todos los cromosomas están activos y lo que hagamos nos dará información sobre todos ellos. Supongamos que no queremos esto. Queremos, por ejemplo, trabajar solamente con `chr2L` ↪ . Lo conseguimos con

```
seqlevels(txdb) = "chr2L"
```

Supongamos que queremos conocer los `GENEID` de los primeros genes.

```
(keysGENEID = head(keys(txdb, keytype="GENEID"),n=3))
```

```
[1] "FBgn0000003" "FBgn0000008" "FBgn0000014"
```

```
columns = c("TXNAME", "TXSTART","TXEND","TXSTRAND")
AnnotationDbi::select(txdb, keysGENEID, columns, keytype="GENEID")
```

```
      GENEID TXNAME TXSTRAND TXSTART
1 FBgn0000003 FBtr0081624 + 2648220
2 FBgn0000008 FBtr0100521 + 18024494
3 FBgn0000008 FBtr0071763 + 18024496
4 FBgn0000008 FBtr0071764 + 18024938
```

```
5 FBgn0000014 FBtr0306337 - 12632936
6 FBgn0000014 FBtr0083388 - 12633349
7 FBgn0000014 FBtr0083387 - 12633349
8 FBgn0000014 FBtr0300485 - 12633349
    TXEND
1 2648518
2 18060339
3 18060346
4 18060346
5 12655767
6 12653845
7 12655300
8 12655474
```

Nos devuelve el identificador del gen (`GENEID`), el nombre del transcrito, la hebra o cadena y el punto de inicio y de finalización. Notemos que el primer gen solo tiene un transcrito mientras que el segundo gen tiene tres transcritos.

Los objetos `TxDb` nos permiten obtener las anotaciones como `GRanges` ↪ . Empezamos con los transcritos.

```
(txdb.tr = transcripts(txdb))
```

```
GRanges object with 5384 ranges and 2 metadata columns:
        seqnames ranges strand |
          <Rle> <IRanges> <Rle> |
    [1] chr2L 7529-9484 + |
    [2] chr2L 7529-9484 + |
    [3] chr2L 7529-9484 + |
    [4] chr2L 21952-24237 + |
    [5] chr2L 66584-71390 + |
    ... ... ... ... .
  [5380] chr2L 22892306-22918560 - |
  [5381] chr2L 22892306-22918647 - |
  [5382] chr2L 22959606-22960915 - |
  [5383] chr2L 22959606-22961179 - |
  [5384] chr2L 22959606-22961179 - |
          tx_id tx_name
        <integer> <character>
    [1] 1 FBtr0300689
    [2] 2 FBtr0300690
    [3] 3 FBtr0330654
    [4] 4 FBtr0309810
    [5] 5 FBtr0306539
    ... ... ...
  [5380] 5380 FBtr0331166
  [5381] 5381 FBtr0111127
  [5382] 5382 FBtr0111241
  [5383] 5383 FBtr0111239
  [5384] 5384 FBtr0111240
  -------
  seqinfo: 1 sequence from dm3 genome
```

Como estamos trabajando con el cromosoma `chr2L` nos devuelve la información en este nuevo (y mejor) formato. Los que ocupan las posiciones 1, 2, 3 y 1000 serían

```
txdb.tr[c(1:3,1000)]
```

```
GRanges object with 4 ranges and 2 metadata columns:
     seqnames ranges strand | tx_id
        <Rle> <IRanges> <Rle> | <integer>
  [1] chr2L 7529-9484 + | 1
  [2] chr2L 7529-9484 + | 2
  [3] chr2L 7529-9484 + | 3
  [4] chr2L 8011405-8026898 + | 1000
```

```
        tx_name
    <character>
  [1] FBtr0300689
  [2] FBtr0300690
  [3] FBtr0330654
  [4] FBtr0079533
  -------
  seqinfo: 1 sequence from dm3 genome
```

También podemos obtener información sobre los exones en modo de un objeto `GRanges`.

```
(txdb.ex = exons(txdb))
```

```
GRanges object with 13850 ranges and 1 metadata column:
        seqnames ranges strand |
           <Rle> <IRanges> <Rle> |
     [1] chr2L 7529-8116 + |
     [2] chr2L 8193-8589 + |
     [3] chr2L 8193-9484 + |
     [4] chr2L 8229-9484 + |
     [5] chr2L 8668-9484 + |
     ... ... ... ... .
  [13846] chr2L 22959606-22959815 - |
  [13847] chr2L 22959877-22960833 - |
  [13848] chr2L 22959877-22960876 - |
  [13849] chr2L 22959877-22960915 - |
  [13850] chr2L 22960932-22961179 - |
          exon_id
        <integer>
     [1] 1
     [2] 2
     [3] 3
     [4] 4
     [5] 5
     ... ...
  [13846] 13846
  [13847] 13847
  [13848] 13848
  [13849] 13849
  [13850] 13850
  -------
  seqinfo: 1 sequence from dm3 genome
```

Como antes el exon que ocupa la posición 123 sería

```
txdb.ex[123]
```

```
GRanges object with 1 range and 1 metadata column:
     seqnames ranges strand | exon_id
        <Rle> <IRanges> <Rle> | <integer>
  [1] chr2L 277930-278323 + | 123
  -------
  seqinfo: 1 sequence from dm3 genome
```

Podemos incluir metadatos adicionales como puede ser el identificador del gen.

```
transcripts(txdb, columns = c("tx_id","tx_name","gene_id"))
```

```
GRanges object with 5384 ranges and 3 metadata columns:
        seqnames ranges strand |
           <Rle> <IRanges> <Rle> |
     [1] chr2L 7529-9484 + |
     [2] chr2L 7529-9484 + |
     [3] chr2L 7529-9484 + |
     [4] chr2L 21952-24237 + |
```

```
   [5] chr2L 66584-71390 + |
   ... ... ... ... .
 [5380] chr2L 22892306-22918560 - |
 [5381] chr2L 22892306-22918647 - |
 [5382] chr2L 22959606-22960915 - |
 [5383] chr2L 22959606-22961179 - |
 [5384] chr2L 22959606-22961179 - |
          tx_id tx_name gene_id
      <integer> <character> <CharacterList>
   [1] 1 FBtr0300689 FBgn0031208
   [2] 2 FBtr0300690 FBgn0031208
   [3] 3 FBtr0330654 FBgn0031208
   [4] 4 FBtr0309810 FBgn0263584
   [5] 5 FBtr0306539 FBgn0067779
   ... ... ... ...
 [5380] 5380 FBtr0331166 FBgn0250907
 [5381] 5381 FBtr0111127 FBgn0250907
 [5382] 5382 FBtr0111241 FBgn0086683
 [5383] 5383 FBtr0111239 FBgn0086683
 [5384] 5384 FBtr0111240 FBgn0086683
 -------
 seqinfo: 1 sequence from dm3 genome
```

Obtenemos las regiones CDS con

```
(txdb.cds = cds(txdb))
```

```
GRanges object with 11003 ranges and 1 metadata column:
        seqnames ranges strand |
          <Rle> <IRanges> <Rle> |
     [1] chr2L 7680-8116 + |
     [2] chr2L 8193-8589 + |
     [3] chr2L 8193-8610 + |
     [4] chr2L 8229-8610 + |
     [5] chr2L 8668-9276 + |
     ... ... ... ... .
 [10999] chr2L 22959877-22960833 - |
 [11000] chr2L 22959877-22960873 - |
 [11001] chr2L 22959877-22960876 - |
 [11002] chr2L 22960932-22960995 - |
 [11003] chr2L 22960932-22961048 - |
          cds_id
        <integer>
     [1] 1
     [2] 2
     [3] 3
     [4] 4
     [5] 5
     ... ...
 [10999] 10999
 [11000] 11000
 [11001] 11001
 [11002] 11002
 [11003] 11003
 -------
 seqinfo: 1 sequence from dm3 genome
```

En los tres casos hemos obtenido un objeto `GRanges`. A este tipo de objetos podemos aplicar otros métodos que nos dan información adicional. Por ejemplo, el número de elementos que lo componen

```
length(txdb.cds)
```

```
[1] 11003
```

Podemos ver la hebra en la que están.

```
strand(txdb.cds)
```

A partir de un objeto de clase `TxDb` podemos obtener un `GRangesList` en la cual separamos por alguna característica. Por ejemplo, los objetos `GRanges` para los distintos genes.

```
transcriptsBy(txdb, by="gene")
```

Podemos agrupar los exones por gen.

```
exonsBy(txdb, by="gene")
```

O bien podemos tener los CDS agrupados por transcrito.

```
cdsBy(txdb, by="tx")
```

O los intrones agrupados por transcrito.

```
intronsByTranscript(txdb)
```

También podemos tener las regiones UTR 5' y 3' agrupadas por transcrito.

```
fiveUTRsByTranscript(txdb)
```

```
threeUTRsByTranscript(txdb)
```

24.5 BSgenome

Estos paquetes contienen datos de secuencias para organismos secuenciados.

BSgenome.Dmelanogaster.UCSC.dm3

```
pacman::p_load(BSgenome.Dmelanogaster.UCSC.dm3,GenomicRanges)
tx2seqs = extractTranscriptSeqs(BSgenome.Dmelanogaster.UCSC.dm3,
    ↪ TxDb.Dmelanogaster.UCSC.dm3.ensGene)
```

La secuencia correspondiente al primer gen sería

```
tx2seqs[[1]]
```

Si queremos conocer la secuencia entre las posiciones 1000 y 1020 entonces

```
tx2seqs[[1]][1000:1020]
```

También podemos traducir estas secuencias a proteínas con

```
suppressWarnings(translate(tx2seqs[[1]]))
```

Para todos los transcritos lo hacemos con

```
suppressWarnings(translate(tx2seqs))
```

No todo lo que se transcribe se traduce. Para obtener obtener las que realmente se traducen se puede hacer

```
cds2seqs = extractTranscriptSeqs(BSgenome.Dmelanogaster.UCSC.dm3,
                                 cdsBy(txdb, by="tx"))
translate(cds2seqs)
```

BSgenome.Hsapiens.UCSC.hg19

Estos paquetes contienen datos de secuencias para organismos secuenciados.

```
pacman::p_load(BSgenome.Hsapiens.UCSC.hg19)
```

```
Hsapiens
```

```
| BSgenome object for Human
| - organism: Homo sapiens
| - provider: UCSC
| - genome: hg19
| - release date: June 2013
| - 298 sequence(s):
| chr1 chr2
| chr3 chr4
| chr5 chr6
| chr7 chr8
| chr9 chr10
| ... ...
| chr19_gl949752_alt chr19_gl949753_alt
| chr20_gl383577_alt chr21_gl383578_alt
| chr21_gl383579_alt chr21_gl383580_alt
| chr21_gl383581_alt chr22_gl383582_alt
| chr22_gl383583_alt chr22_kb663609_alt
|
| Tips: call 'seqnames()' on the object to get all
| the sequence names, call 'seqinfo()' to get the
| full sequence info, use the '$' or '[[' operator
| to access a given sequence, see '?BSgenome' for
| more information.
```

Nos fijamos en las secuencias.

```
seqNms = seqnames(Hsapiens)
head(seqNms)
```

```
[1] "chr1" "chr2" "chr3" "chr4" "chr5" "chr6"
```

Nos fijamos en las dos primeras.

```
getSeq(Hsapiens, seqNms[1:2])
```

Podemos, utilizando un objeto GRanges, obtener la secuencia de bases en los cromosomas que queramos, en la hebra que queramos, entre las posiciones que queramos.

```
rngs <-GRanges(seqnames = c('chr1', 'chr4'), strand=c('+','-'),
                    ranges = IRanges(start=c(100000,300000),
                         end=c(100023,300037)))
rngs
res <-getSeq(Hsapiens, rngs)
res
```

24.6 OrganismDb

Un paquete de tipo OrganismDb nos permite combinar información (para el organismo con que estemos trabajando) de GO.db con el correspondiente TxDb y OrgDb.[177]

[177] Buscar OrganismDb en Bioconductor.

```
pacman::p_load(Homo.sapiens)
```

Tomamos las dos primeras.

```
keys = head(keys(Homo.sapiens, keytype="ENTREZID"), n=2)
columns = c("SYMBOL","TXNAME")
AnnotationDbi::select(Homo.sapiens, keys, columns, keytype="ENTREZID")
```

```
  ENTREZID SYMBOL TXNAME
1 1 A1BG uc002qsd.4
2 1 A1BG uc002qsf.2
3 2 A2M uc001qvk.1
4 2 A2M uc009zgk.1
```

También podemos obtener `GRanges`.

```
transcripts(Homo.sapiens, columns=c("TXNAME","SYMBOL"))
```

```
GRanges object with 82960 ranges and 2 metadata columns:
              seqnames ranges strand |
                 <Rle> <IRanges> <Rle> |
     [1] chr1 11874-14409 + |
     [2] chr1 11874-14409 + |
     [3] chr1 11874-14409 + |
     [4] chr1 69091-70008 + |
     [5] chr1 321084-321115 + |
     ... ... ... ... .
  [82956] chrUn_gl000237 1-2686 - |
  [82957] chrUn_gl000241 20433-36875 - |
  [82958] chrUn_gl000243 11501-11530 + |
  [82959] chrUn_gl000243 13608-13637 + |
  [82960] chrUn_gl000247 5787-5816 - |
                TXNAME SYMBOL
        <CharacterList> <CharacterList>
     [1] uc001aaa.3 DDX11L1
     [2] uc010nxq.1 DDX11L1
     [3] uc010nxr.1 DDX11L1
     [4] uc001aal.1 OR4F5
     [5] uc001aaq.2 <NA>
     ... ... ...
  [82956] uc011mgu.1 <NA>
  [82957] uc011mgv.2 <NA>
  [82958] uc011mgw.1 <NA>
  [82959] uc022brq.1 <NA>
  [82960] uc022brr.1 <NA>
  -------
  seqinfo: 93 sequences (1 circular) from hg19 genome
```

24.7 biomaRt

El paquete [44, biomaRt] es un interfaz para poder acceder a una serie de bases de datos que implementan BioMart. [178]

[178] http://www.biomart.org.

```
pacman::p_load(biomaRt)
```

Podemos ver la lista de `mart`'s que tenemos (mostramos los primeros).

```
listMarts(host="https://www.ensembl.org")
```

```
             biomart version
1 ENSEMBL_MART_ENSEMBL Ensembl Genes 114
2 ENSEMBL_MART_MOUSE Mouse strains 114
3 ENSEMBL_MART_SNP Ensembl Variation 114
4 ENSEMBL_MART_FUNCGEN Ensembl Regulation 114
```

Elegimos utilizar `ensembl`.

```
(ensembl = useMart("ENSEMBL_MART_ENSEMBL",host="https://www.ensembl.org")
    ↪ )
```

```
Object of class 'Mart':
  Using the ENSEMBL_MART_ENSEMBL BioMart database
  No dataset selected.
```

Ahora hemos de elegir el conjunto de datos a utilizar. Podemos ver los disponibles con

```
head(listDatasets(ensembl))
```

```
                    dataset
1 abrachyrhynchus_gene_ensembl
2 acalliptera_gene_ensembl
3 acarolinensis_gene_ensembl
4 acchrysaetos_gene_ensembl
5 acitrinellus_gene_ensembl
6 amelanoleuca_gene_ensembl
                        description
1 Pink-footed goose genes (ASM259213v1)
2 Eastern happy genes (fAstCal1.3)
3 Green anole genes (AnoCar2.0v2)
4 Golden eagle genes (bAquChr1.2)
5 Midas cichlid genes (Midas_v5)
6 Giant panda genes (ASM200744v2)
     version
1 ASM259213v1
2 fAstCal1.3
3 AnoCar2.0v2
4 bAquChr1.2
5 Midas_v5
6 ASM200744v2
```

Elegimos la correspondiente a ser humano.

```
(ensembl = useMart("ENSEMBL_MART_ENSEMBL",dataset="hsapiens_gene_ensembl
    ↪ ",
    host="https://www.ensembl.org"))
```

Podemos ver los atributos con

```
head(listAttributes(ensembl))
```

De hecho, son

```
nrow(listAttributes(ensembl))
```

Podemos comprobar que hay muchos que identifican el gen con las sondas de Affymetrix, en concreto, con los AffyID. Supongamos que nos fijamos en los siguientes AffyID.

```
affyids=c("202763_at","209310_s_at","207500_at")
```

¿A qué genes corresponden?

```
getBM(attributes=c('affy_hg_u133_plus_2', 'entrezgene'),
                    filters = 'affy_hg_u133_plus_2',
                    values = affyids, mart = ensembl)
```

Podemos obtener todos los identificadores.

```
head(getBM(attributes='affy_hg_u133_plus_2', mart = ensembl))
```

24.8 KEGGREST

Con [135, KEGGREST] podemos acceder a la base de datos KEGG. En concreto permite el acceso a KEGG REST API.

24.9 Tareas habituales con anotaciones

¿Cuáles son las tareas que tenemos que realizar habitualmente? En esta sección vemos posibles soluciones que las resuelvan.

24.9.1 Cambiar identificadores de un ExpressionSet

[179] § 2.7.6.

Consideremos el ExpressionSet tamidata::gse1397.[179] Las características están codificadas utilizando los identificadores Affymetrix. Pretendemos incorporar la información que nos permita conocer la correspondencia entre estos identificadores (PROBEID) y los identificadores Entrez y Ensembl. Esta información la incorporamos como `fData` del ExpressionSet.

Necesitamos los paquetes.

```
pacman::p_load("Biobase","AnnotationDbi","BiocGenerics")
```

Leemos los datos.

```
data(gse1397,package="tamidata")
```

¿Qué paquete de anotación necesitamos?

```
Biobase::annotation(gse1397)
```

```
[1] "hgu133a"
```

Lo cargamos. En https://www.bioconductor.org/packages/3.3/data/annotation/ tenemos el listado de los paquetes de anotación de los que dispone Bioconductor.

```
pacman::p_load("hgu133a.db")
```

Como hemos visto en § 24.1 con `AnnotationDbi::columns` podemos ver la información que tenemos y con AnnotationDbi::keytypes las llaves con las que podemos realizar consultas.

```
columns(hgu133a.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

```
keytypes(hgu133a.db)
```

```
 [1] "ACCNUM" "ALIAS" "ENSEMBL"
 [4] "ENSEMBLPROT" "ENSEMBLTRANS" "ENTREZID"
 [7] "ENZYME" "EVIDENCE" "EVIDENCEALL"
[10] "GENENAME" "GENETYPE" "GO"
[13] "GOALL" "IPI" "MAP"
[16] "OMIM" "ONTOLOGY" "ONTOLOGYALL"
[19] "PATH" "PFAM" "PMID"
[22] "PROBEID" "PROSITE" "REFSEQ"
[25] "SYMBOL" "UCSCKG" "UNIPROT"
```

Vemos que, en este caso coinciden `columns` y `keys`. Los valores los tenemos con

```
head(keys(hgu133a.db,keytype="PROBEID"))
```

```
[1] "1007_s_at" "1053_at" "117_at"
[4] "121_at" "1255_g_at" "1294_at"
```

que corresponde con los identificadores Affy o identificadores de las sondas. Podemos ver los primeros que tenemos en nuestros datos.

```
head(featureNames(gse1397))
```

```
[1] "1007_s_at" "1053_at" "117_at"
[4] "121_at" "1255_g_at" "1294_at"
```

¿Coinciden todos?

```
table(featureNames(gse1397) == keys(hgu133a.db,keytype="PROBEID"))
```

```

FALSE TRUE
   53 22230
```

Vemos que no todos coinciden. ¿Quienes son?

```
(control = which(featureNames(gse1397) != keys(hgu133a.db,keytype="PROBEID")))
```

```
 [1] 22231 22232 22233 22234 22235 22236 22237
 [8] 22238 22239 22240 22241 22242 22243 22244
[15] 22245 22246 22247 22248 22249 22250 22251
[22] 22252 22253 22254 22255 22256 22257 22258
[29] 22259 22260 22261 22262 22263 22264 22265
[36] 22266 22267 22268 22269 22270 22271 22272
[43] 22273 22274 22275 22276 22277 22278 22279
[50] 22280 22281 22282 22283
```

¿Que identificadores tienen estas sondas?

```
head(featureNames(gse1397)[control])
```

```
[1] "AFFX-hum_alu_at"
[2] "AFFX-HUMGAPDH/M33197_3_at"
[3] "AFFX-HUMGAPDH/M33197_5_at"
[4] "AFFX-HUMGAPDH/M33197_M_at"
[5] "AFFX-HUMISGF3A/M97935_3_at"
[6] "AFFX-HUMISGF3A/M97935_5_at"
```

Son sondas de control que no corresponden a ningún gen. Hemos dicho antes que queremos modificar los identificadores utilizados en el ExpressionSet. Y queremos hacerlo pasando a Entrez y Ensembl. Buscamos las correspondencias.

```
probeid2entrez =
    AnnotationDbi::select(hgu133a.db,keys=featureNames(gse1397),
                          columns="ENTREZID",keytype="PROBEID")
```

Nos ha devuelvo un `data.frame`.

```
class(probeid2entrez)
```

```
[1] "data.frame"
```

Es importante notar que no hay una correspondencia 1-1 entre los distintos identificadores.

```
head(probeid2entrez)
```

```
   PROBEID ENTREZID
1 1007_s_at 780
2 1053_at 5982
3 117_at 3310
4 121_at 7849
5 1255_g_at 2978
6 1294_at 7318
```

La primera sonda tiene dos identificadores Entrez distintos. Una opción para resolver esta multiplicidad es utilizar BiocGenerics::match.

```
indices = match(featureNames(gse1397),probeid2entrez$PROBEID)
```

¿Cómo se ha resuelto?

```
head(featureNames(gse1397))
```

```
[1] "1007_s_at" "1053_at" "117_at"
[4] "121_at" "1255_g_at" "1294_at"
```

```
head(probeid2entrez$PROBEID,n=7)
```

```
[1] "1007_s_at" "1053_at" "117_at"
[4] "121_at" "1255_g_at" "1294_at"
[7] "1316_at"
```

```
head(indices)
```

```
[1] 1 2 3 4 5 6
```

Elige la primera correspondencia e ignora las demas como hemos visto.

```
eset = gse1397
fData(eset) = probeid2entrez[indices,]
```

Podemos comprobar, con Base::all.equal, si son iguales los nombres del ExpressionSet con la columna de identificadores Affymetrix.

```
all.equal(fData(eset)$PROBEID,featureNames(eset))
```

```
[1] TRUE
```

24.10 Ejercicios

* **Ex. 48 —** Determinar los códigos Entrez, Gene Ontology y Ensembl para los genes BRCA1 y BRCA2 implicados en el cáncer de mama.

* **Ex. 49 —** Se pide encontrar el gen BRCA1 en el ratón (Mus musculus). Utilizad el paquete de anotación [**R-org.Mm.eg.db**].

Apéndice A

Código sin más

Incluimos en este capítulo código al que queremos referenciar desde el resto del manual.

A.1 GSE198668

Fueron inicialmente analizados por Aarón García Blázquez en el curso 2021-22.

Construcción del ExpressionSet

Se encuentran en https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE198668. Cargamos los paquetes necesarios.

```
pacman::p_load(Biobase,GEOquery,affy)
```

Descargamos datos.

```
dir = getwd()
gcel = getGEOSuppFiles("GSE198668")
setwd("GSE198668/")
system("tar xvf GSE198668_RAW.tar")
GSE198668raw = affy::ReadAffy()
setwd("../")
system("rm -fr GSE198668")
setwd(dir)
```

Normalizados los datos.

```
GSE198668 = affy::rma(GSE198668raw)
```

Construimos el `Biobase::ExpressionSet`.

```
#Añadimos los datos fenotípicos a partir de un documento de texto
pd0 = read.csv(paste0(dirTamiData,"GSE198668_metadata.csv"),
                header=TRUE,row.names ="ID")

#Añadimos los datos del experimento
exd0 = new("MIAME",name="Hellerud et al.",lab = "Medical Biochemistry",
           contact = "o.k.olstad@medisin.uio.no",
           title = "Massive Organ Inflammation in Experimental and in Clinical
           Meningococcal Septic Shock",
           abstract = "The aims of the present study were to investigate the
           inflammatory responses in various organs as compared with the
           systemic circulation in an experimental porcine model of
           meningococcal sepsis using wild-type N. meningitidis;
           to study the role of LPS versus non-LPS microbial molecules as
```

```
                triggers of organ inflammation in meningococcal sepsis",
                url = "https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE198668")
metadata = data.frame(labelDrescription =
                                    c("Órgano del que procede la muestra",
                                      "Cepa inoculada", "Dosis"),
                              row.names = colnames(pd0))
datosfenotipo = AnnotatedDataFrame(data = pd0, varMetadata = metadata)
GSE198668 = ExpressionSet(assayData=exprs(GSE198668), phenoData = datosfenotipo,
                                 experimentData = exd0, annotation = "porcine.db")
```

Modificamos las variables fenotípicas.

```
pData(GSE198668)$Organ = factor(pData(GSE198668)$Organ)
pData(GSE198668)$Strand = factor(pData(GSE198668)$Strand)
```

El chip empleado en este estudio no posee como clave la entrada de Ensembl pero se ha añadido el símbolo del gen a su sonda.

```
pacman::p_load(AnnotationDbi,porcine.db)
probeid2entrez =
     AnnotationDbi::select(porcine.db,
                                    keys = featureNames(GSE198668),
                                    columns=c("ENTREZID","SYMBOL"),
                                    keytype = "PROBEID")
indices = match(featureNames(GSE198668),probeid2entrez$PROBEID)
fData(GSE198668) = probeid2entrez[indices,]
all.equal(fData(GSE198668)$PROBEID,featureNames(GSE198668))
```

Eliminación de las correspondencias múltiples.

```
which(table(probeid2entrez$PROBEID)>1)
a = probeid2entrez
c1 = match(unique(a[,1]),a[,1])
a1 = a[c1,]
c2 = match(unique(a1[,2]),a1[,2])
a2 = a1[c2,]
a2 = a2[−which(is.na(a2$ENTREZID) & is.na(a2$SYMBOL)),]
GSE198668 = GSE198668[match(a2$PROBEID,fData(GSE198668)$PROBEID),]
fData(GSE198668) = a2
```

Y lo guardamos.

```
save(GSE198668,file=paste0(dirTamiData,"GSE198668.rda"))
```

A.2 gse80200

Datos procesados inicialmente por Daniel González Gamblor en el curso 2021-22. Se incluye su código con leves modificaciones.

Se eligió el experimento GSE80200, un estudio de expresión en Arabidopsis thaliana. Este estudio utiliza la plataforma ATH1-121501 GPL21063 de Affymetrix.

```
pacman::p_load(GEOquery,affy,Biobase,ath1121501.db)
download = getGEOSuppFiles("GSE80200")
untar("GSE80200/GSE80200_RAW.tar")
gse80200raw = ReadAffy(sampleNames =
                                    c("S1_control", "S2_control",
                                      "S3_control", "S4_treatment",
                                      "S5_treatment", "S6_treatment"))
gse80200 = rma(gse80200raw)

data = new("MIAME",
                name="Willems P, Denecker J, Van Breusegem F",
                lab="PSB",
                contact="pawil@psb.ugent.be",
```

```
                title="Transcriptional responses in Arabidopsis seedlings after
hydrogen peroxide treatment",
abstract="Excessive levels of reactive oxygen species (ROS) cause cellular
stress through damage to all classes of macromolecules and result in cell
death. However, ROS can also act as signaling molecules in various biological
 processes. In plants, ROS signaling has been documented in environmental
stress perception, plant development and cell death amongst others.
Knowledge on the regulatory events governing ROS signal transduction is
however still scratching the surface. To further elucidate the transcriptional
 response and regulation upon ROS accumulation we supplemented Arabidopsis
 seedlings with a 10mM hydrogen peroxide (H2O2) solution to trigger oxidative
 stress.",
url="https://www-ncbi-nlm-nih-gov.ezproxy.u-pec.fr/geo/query/acc.cgi?acc=
    ↪ GSE80200",
pubMedIds="27246095"
)
experimentData(gse80200) = data

## pData
type = factor(c(0, 0, 0, 1, 1, 1), levels = 0:1, labels=c("H2O", "H2O2"))
pd = data.frame(type)
rownames(pd) = colnames(gse80200)
pData(gse80200) = pd

## ath1121501.db tiene opciones limitadas, no contiene ENSEMBL
## Utiliza `Biomart`
pacman::p_load(biomaRt)

#Select mart
mart = useMart(biomart="plants_mart", host="https://plants.ensembl.org")
mart = useDataset("athaliana_eg_gene", mart)
# Annotation
info = getBM(mart=mart,
             attributes=c("affy_ath1_121501","entrezgene_id",
                          "ensembl_gene_id","external_gene_name"),
             filters="affy_ath1_121501",
              values = featureNames(gse80200), uniqueRows=TRUE)

##La columna de IDs de ensembl corresponde en realidad con los
    ↪ identificadores
##de la base de datos más popular para *Arabidopsis thaliana*: la
    ↪ Arabidopsis
##Infor#mation Resource ([TAIR](https://www.arabidopsis.org/index.jsp)).

m <-match(featureNames(gse80200), info[,"affy_ath1_121501"])
fData(gse80200) = info[m,]

## Utilizo como identificadores primarios los ENSEMBL
ids <-fData(gse80200)
ids <-ids[,c("affy_ath1_121501", "ensembl_gene_id")]
m1 <-match(unique(ids[,1]), ids[,1])
gse80200 = gse80200[m1,]
ids <-fData(gse80200)
m2 <-match(unique(ids[,2]), ids[,2])
gse80200 = gse80200[m2,]
names(fData(gse80200)) = c("PROBEID","ENTREZID","ENSEMBL","GENENAME")
pData(gse80200)$type = as.factor(pData(gse80200)$type)
save(gse80200, file =paste0(dirTamiData,"gse80200.rda"))
```

A.3 gse21443

Lo que sigue está modificado a partir de un código original de Gonzalo Antón Bernat (20.06.23). La finalidad del estudio es comprender cómo las plantas adquieren y procesan nutrientes como el hierro. Los grupos experimentales son los siguientes: arabidopsis WT con hierro (2 répli-

cas), arabidopsis WT sin hierro (2 réplicas), arabidopsis pye con hierro (2 réplicas) y arabidopsis pye sin hierro (2 réplicas).

```
## gse21443
pacman::p_load(GEOquery,affy)
getGEOSuppFiles("GSE21443")
setwd("GSE21443")
system("tar xvf GSE21443_RAW.tar")
gse21443raw = ReadAffy()
gse21443 = affy::rma(gse21443raw)
setwd("../")
datosexperimento <-new('MIAME',
                        name = 'Terri Anita Long',
                        lab = 'Philip Benfey',
                        contact = ' tlong@duke.edu',
                        title = 'Expression analysis of pye-1 mutants
and root pericycle cells to iron sufficient or iron deficient conditions',
pubMedIds = '20675571'
)

experimentData(gse21443) = datosexperimento

muestras = rownames(pData(gse21443))
condicion = factor(rep(0:1,each=4),
                   levels = 0:1, labels = c("Control", "Popeye"))
medio = factor(c(0, 0, 1, 1, 0, 0, 1, 1), levels = 0:1,
               labels = c("Con_hierro", "Sin_hierro"))
replica = factor(c(0, 1, 0, 1, 0, 1, 0, 1), levels = 0:1,
                 labels = c("Replica_1", "Replica_2"))
pd = data.frame(muestras, condicion, medio, replica)
rownames(pd) = colnames(exprs(gse21443))
pData(gse21443) = pd
save(gse21443,file="gse21443.rda")
```

```
pacman::p_load("EnrichmentBrowser")
arabi_go = getGenesets(org='ath', # El organismo debe estar en código KEGG
                       db='go',
                       onto='BP')
save(arabi_go, file = 'arabi_go.rda')

arabi_KEGG = getGenesets(org = 'ath',
                         db='kegg')
save(arabi_KEGG, file = 'arabi_KEGG.rda')
```

A.4 bcrneg

```
pacman::p_load("Biobase","ALL")
data(ALL)
bcell = grep("^B",as.character(ALL$BT))
types = c("NEG","BCR/ABL")
moltyp = which(as.character(ALL$mol.biol) %in% types)
bcrneg = ALL[,intersect(bcell,moltyp)]
save(bcrneg,file=paste0(dirTamiData,"bcrneg.rda"))
```

A.5 gse183019

Los datos GSE183019 es un estudio sobre cáncer de prostata.[1] Podemos construir el `SummarizedExperiment:RangedSummarizedExperiment` ↪ con el siguiente código.

[1] Angela Riffo-Campos propuso su estudio.

```
pacman::p_load(GEOquery,SummarizedExperiment,Biobase,edgeR)
x0 = getGEO("GSE183019")
y = pData(x0[[1]]) ## Variables fenotípicas
```

En particular necesitamos transformar la variable fenotípica que nos identifica la muestra.

```
y$source_name_ch1 =unlist(lapply(y$source_name_ch1,function(i){
    a = strsplit(i,split=" ")
    b = tail(a[[1]], n=1)
    gsub(b,pattern="-",replacement=".")
}))
```

Bajamos el fichero con los conteos y la anotación.

```
wget ttps://ftp.ncbi.nlm.nih.gov/geo/series/GSE183nnn/
    ↪ GSE183019/suppl/GSE183019%5Fprocessed%5Fcounts.
    ↪ txt.gz
gzip -d GSE183019_processed_counts.txt.gz
```

Leemos los datos.

```
x = read.csv("GSE183019_processed_counts.txt",header=TRUE,sep="\t")
system("rm GSE183019_processed_counts.txt")
```

Fijamos la matriz de conteos inicial.

```
counts = x[,-c(1:2)]
```

Vemos correspondencia entre columnas de la matriz de conteos y las variables fenotípicas.

```
w = match(y$source_name_ch1,names(counts))
counts1 = counts[,w]
colnames(counts1) = rownames(y)
se = SummarizedExperiment(assay=counts1,colData = y)
rowData(se) = x[,1:2] ## Anotación
```

Eliminamos genes con conteos bajos. En particular aquellos que no lleguen a 0.5 lecturas por millón en al menos 20 muestras. Tenemos 226 muestras en el estudio.

```
to_keep = rowSums(edgeR::cpm(assay(se)) > 0.5) > 20
se = se[to_keep,]
```

Normalizamos los tamaños de las librerías.

```
dge = edgeR::DGEList(counts=assay(se))
dge.c = edgeR::estimateCommonDisp(dge)
assay(se) = dge.c$pseudo.counts
gse183019 = se
```

Cambiamos los tipos y nombres de algunas variables fenotípicas.

```
names(colData(gse183019))[53] = "age"
names(colData(gse183019))[57] = "pathologic_stage"
names(colData(gse183019))[62] = "tissue"
colData(gse183019)["age"] = as.numeric(colData(gse183019)["age"])
colData(gse183019)["tissue"] = factor(colData(gse183019)["tissue"])
colData(gse183019)["pathologic_stage"] =
    factor(colData(gse183019)["pathologic_stage"])
```

Guardamos el `SummarizedExperiment`.

```
save(gse183019,file = paste0(dirTamiData,"gse183019.rda"))
```

Bibliografía

[1] H. Abdi y L. J. Williams. «Principal component analysis». En: *Wiley Interdisciplinary Reviews: Computational Statistics* 2.4 (2010), págs. 433-459.

[2] Sudipta Acharya, Sriparna Saha y N. Nikhil. «Unsupervised gene selection using biological knowledge : application in sample clustering». En: *BMC Bioinformatics* 18.1 (2017), pág. 513. ISSN: 1471-2105. DOI: 10.1186/s12859-017-1933-0.

[3] A. Agresti. *Categorical Data Analysis*. Second. Wiley, 2002.

[4] A. Agresti. *Categorical Data Analysis*. Third. Wiley-Interscience, 2013.

[5] Alan Agresti. *An Introduction to Categorical Data Analysis*. Third edition. Wiley John + Sons, 2019. 400 págs. ISBN: 1119405262.

[6] Alan Agresti. *Foundations of Linear and Generalized Linear Models*. Wiley, 2015. 480 págs. ISBN: 1118730038.

[7] Alan Agresti y Maria Kateri. *Foundations of Statistics for Data Scientists*. Chapman y Hall/CRC, 2021. DOI: 10.1201/9781003159834.

[8] JJ Allaire et al. *rmarkdown: Dynamic Documents for R*. R package version 2.29. 2024.

[9] S. Anders et al. «Count-based differential expression analysis of RNA sequencing data using R and Bioconductor». En: *Nat. Protocols* 8.9 (2013), págs. 1765-1786. DOI: 10.1038/nprot.2013.099.

[10] Simon Anders y Wolfgang Huber. «Differential expression analysis for sequence count data». En: *Genome Biology* 11.10 (2010), R106. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-10-r106.

[11] Guillermo Ayala. *tami: Statistical Bioinformatics*. R package version 1.0. 2024.

[12] Guillermo Ayala. *tamidata: Data sets for Statistical Bioinformatics*. R package version 0.6. 2025.

[13] Guillermo Ayala. *tamidata3: Omics data sets*. R package version 0.3. 2024.

[14] M. Baker y D. Penny. «Is there a reproducibility crisis?» En: *Nature* 533.7604 (2016). cited By 44, págs. 452-454. DOI: 10.1038/533452A.

[15] Andrew J. Bass et al. *biobroom: Turn Bioconductor objects into tidy data frames*. R package version 1.39.0. 2024. DOI: 10.18129/B9.bioc.biobroom.

[16] Yoav Benjamini y Yosef Hochberg. «Controlling the False Discovery Rate: A Practical and Powerful Approach to Multiple Testing». English. En: *Journal of the Royal Statistical Society. Series B (Methodological)* 57.1 (1995), págs. 289-300. ISSN: 00359246.

[17] Yoav Benjamini y Daniel Yekutieli. «The control of the false discovery rate in multiple testing under dependency». En: *The Annals of Statistics* 29.4 (2001), págs. 1165-1188.

[18] J. M. Bland y D. G. Altman. «Statistical methods for assessing agreement between two methods of clinical measurement». En: *Lancet* (1986), págs. 307-310.

[19] Mark J. Bolland, Alison Avenell y Andrew Grey. «Publication integrity: what is it, why does it matter, how it is safeguarded and how could we do better?» En: *Journal of the Royal Society of New Zealand* (2024), págs. 1-20. ISSN: 1175-8899. DOI: 10.1080/03036758.2024.2325004.

[20] B.M. Bolstad et al. «A comparison of normalization methods for high density oligonucleotide array data based on variance and bias». En: *Bioinformatics* 19.2 (2003), págs. 185-193. DOI: 10.1093/bioinformatics/19.2.185. eprint: http://bioinformatics.oxfordjournals.org/content/19/2/185.full.pdf+html.

[21] Ben Bolstad. *affyPLM: Methods for fitting probe-level models*. R package version 1.83.3. 2025. DOI: 10.18129/B9.bioc.affyPLM.

[22] Ben Bolstad. «Methods in Microarray Normalization». En: Drug Discovery Series 10. CRC Press, 2008. Cap. 3, págs. 41-60.

[23] Richard Bourgon, Robert Gentleman y Wolfgang Huber. «Independent filtering increases detection power for high-throughput experiments». En: *Proceedings of the National Academy of Sciences* 107.21 (2010), págs. 9546-9551. DOI: 10.1073/pnas.0914005107. eprint: http://www.pnas.org/content/107/21/9546.full.pdf+html.

[24] Rosemary Braun, Leslie Cope y Giovanni Parmigiani. «Identifying differential correlation in gene/pathway combinations». En: *BMC Bioinformatics* 9.1 (2008), pág. 488. ISSN: 1471-2105. DOI: 10.1186/1471-2105-9-488.

[25] W. John Braun y Duncan J. Murdoch. *A First Course in Statistical Programming with R*. Cambridge University Press, 202021. 280 págs. ISBN: 1108995144.

[26] Andrew C Browning et al. «Comparative gene expression profiling of human umbilical vein endothelial cells and ocular vascular endothelial cells». En: *British Journal of Ophthalmology* 96.1 (2012), págs. 128-132. DOI: 10.1136/bjophthalmol-2011-300572. eprint: http://bjo.bmj.com/content/96/1/128.full.pdf+html.

[27] M. Carlson et al. *GenomicFeatures: Query the gene models of a given organism/assembly*. R package version 1.59.1. 2024. DOI: 10.18129/B9.bioc.GenomicFeatures.

[28] Marc Carlson. *ath1121501.db: Affymetrix Affymetrix ATH1-121501 Array annotation data (chip ath1121501)*. R package version 3.13.0. 2021.

[29] Marc Carlson. *GO.db: A set of annotation maps describing the entire Gene Ontology*. R package version 3.21.0. 2025.

[30] Marc Carlson. *hgu133a.db: Affymetrix Affymetrix HG-U133A Array annotation data (chip hgu133a)*. R package version 3.13.0. 2021.

[31] Marc Carlson. *hgu133plus2.db: Affymetrix Affymetrix HG-U133*$_Plus_2 Arrayannotationdata(chiphgu1$
R package version 3.13.0. 2021.

[32] Marc Carlson. *hgu95av2.db: Affymetrix Affymetrix HG*$_U 95Av2Arrayannotationdata(chiphgu95av2)$.
R package version 3.13.0. 2021.

[33] Benilton Carvalho y Rafael Irizarry. *oligo: Preprocessing tools for oligonucleotide arrays*. R package version 1.71.7. 2025. DOI: 10.18129/B9.bioc.oligo.

[34] Hanbo Chen. *VennDiagram: Generate High-Resolution Venn and Euler Plots*. R package version 1.7.3. 2022. DOI: 10.32614/CRAN.package.VennDiagram.

[35] Min Chen et al. «A powerful Bayesian meta-analysis method to integrate multiple gene set enrichment studies». En: *Bioinformatics* 29.7 (2013), págs. 862-869. DOI: 10.1093/bioinformatics/btt068. eprint: http://bioinformatics.oxfordjournals.org/content/29/7/862.full.pdf+html.

[36] Yunshun Chen, Aaron T. L. Lun y Gordon K. Smyth. «From reads to genes to pathways: differential expression analysis of RNA-Seq experiments using Rsubread and the edgeR quasi-likelihood pipeline». En: *F1000Research* 5 (2016), pág. 1438. DOI: 10.12688/f1000research.8987.2.

[37] Yunshun Chen et al. «edgeR v4: powerful differential analysis of sequencing data with expanded functionality and improved support for small counts and larger datasets». En: *Nucleic Acids Research* 53.2 (2025), gkaf018. ISSN: 1362-4962. DOI: 10.1093/nar/gkaf018. eprint: https://academic.oup.com/nar/article-pdf/53/2/gkaf018/61573115/gkaf018.pdf.

[38] Yunshun Chen et al. *edgeR: Empirical Analysis of Digital Gene Expression Data in R*. R package version 4.5.10. 2025. DOI: 10.18129/B9.bioc.edgeR.

[39] Ana Conesa et al. «A survey of best practices for RNA-seq data analysis». En: *Genome Biology* 17.1 (2016), págs. 1-19. ISSN: 1474-760X. DOI: 10.1186/s13059-016-0881-8.

[40] Aedín C. Culhane et al. «GeneSigDB: a manually curated database and resource for analysis of gene expression signatures». En: *Nucleic Acids Research* 40.D1 (2012), págs. D1060-D1066. DOI: 10.1093/nar/gkr901. eprint: http://nar.oxfordjournals.org/content/40/D1/D1060.full.pdf+html.

[41] Sean Davis. *GEOquery: Get data from NCBI Gene Expression Omnibus (GEO)*. R package version 2.75.0. 2024. DOI: 10.18129/B9.bioc.GEOquery.

[42] S. Dudoit, J.P. Shaffer y J.C. Boldrick. «Multiple hypothesis testing in microarray experiments». En: *Statistical Science* 18 (2003). Microarrays, págs. 71-103.

[43] Sandrine Dudoit y Robert Gentleman. *Cluster Analysis in DNA Microarray Experiments*. Bioconductor Short Course Winter 2002. Bioconductor.org. 2002. URL: http://www.bioconductor.org/help/course-materials/2002/Seattle02/Cluster/cluster.pdf.

[44] Steffen Durinck y Wolfgang Huber. *biomaRt: Interface to BioMart databases (i.e. Ensembl)*. R package version 2.63.3. 2025. DOI: 10.18129/B9.bioc.biomaRt.

[45] Brad Efron y R. Tibshirani. *GSA: Gene Set Analysis*. R package version 1.03.3. 2024.

[46] Bradley Efron y Robert Tibshirani. «On testing the significance of sets». En: *Annals of Applied Statistics* 1.1 (2007). Gene set analysis, págs. 107-129. DOI: 10.1214/07-AOAS101.

[47] B. Ewing y P. Green. «Base-calling of automated sequencer traces using phred. II. Error probabilities.» En: *Genome research* 8 (3 1998), págs. 186-194. ISSN: 1088-9051. ppublish.

[48] Brent Ewing et al. «Base-Calling of Automated Sequencer Traces UsingiPhred./i I. Accuracy Assessment». En: *Genome Research* 8.3 (1998), págs. 175-185. DOI: 10.1101/gr.8.3.175.

[49] S. Falcon y R. Gentleman. «Using GOstats to test gene lists for GO term association». En: *Bioinformatics* 23.2 (2007), págs. 257-258. DOI: 10.1093/bioinformatics/btl567. eprint: http://bioinformatics.oxfordjournals.org/content/23/2/257.full.pdf+html.

[50] Laurent Gautier et al. «affy—analysis of Affymetrix GeneChip data at the probe level». En: *Bioinformatics* 20.3 (2004), págs. 307-315. ISSN: 1367-4803. DOI: http://dx.doi.org/10.1093/bioinformatics/btg405.

[51] Ludwig Geistlinger, Gergely Csaba y Ralf Zimmer. *EnrichmentBrowser: Seamless navigation through combined results of set-based and network-based enrichment analysis*. R package version 2.37.0. 2024. DOI: 10.18129/B9.bioc.EnrichmentBrowser.

[52] Ludwig Geistlinger et al. «From sets to graphs: towards a realistic enrichment analysis of transcriptomic systems». En: *Bioinformatics* 27.13 (2011), págs. i366-i373. DOI: 10.1093/bioinformatics/btr228. eprint: http://bioinformatics.oxfordjournals.org/content/27/13/i366.full.pdf+html.

[53] R. Gentleman et al. *Biobase: Base functions for Bioconductor*. R package version 2.67.0. 2024. DOI: 10.18129/B9.bioc.Biobase.

[54] R. Gentleman et al., eds. *Bioinformatics and Computational Biology Solutions Using R and Bioconductor*. Springer, 2005.

[55] Robert Gentleman. *Category: Category Analysis*. R package version 2.73.0. 2024. DOI: 10.18129/B9.bioc.Category.

[56] Robert Gentleman. *GOstats: Tools for manipulating GO and microarrays*. R package version 2.73.0. 2024. DOI: 10.18129/B9.bioc.GOstats.

[57] Robert Gentleman. «Reproducible Research: A Bioinformatics Case Study». En: *Statistical Applications in Genetics and Molecular Biology* 4.1 (2005), pág. 2. DOI: 10.2202/1544-6115.1034.

[58] Robert Gentleman y Duncan Temple Lang. *Statistical Analyses and Reproducible Research*. Inf. téc. Bioconductor Project. Bioconductor Project Working Papers, 2004.

[59] Robert Gentleman et al. *genefilter: methods for filtering genes from high-throughput experiments*. R package version 1.89.0. 2024. DOI: 10.18129/B9.bioc.genefilter.

[60] Jeff Gentry. *annotate: Annotation for microarrays*. R package version 1.85.0. 2024. DOI: 10.18129/B9.bioc.annotate.

[61] Jean Dickinson Gibbons y Subhabrata Chakraborti. *Nonparametric Statistical Inference, Fourth Edition: Revised and Expanded (Statistics: A Series of Textbooks and Monographs)*. CRC Press, 2003. ISBN: 0-8247-4052-1.

[62] J. J. Goeman y P. Buhlmann. «Analyzing gene expression data in terms of gene sets: methodological issues». En: *Bioinformatics* 23.8 (2007), págs. 980-987. DOI: 10.1093/bioinformatics/btm051. eprint: http://bioinformatics.oxfordjournals.org/content/23/8/980.full.pdf+html.

[63] T. R. Golub et al. «Molecular classification of cancer: class discovery and class prediction by gene expression monitoring.» eng. En: *Science* 286.5439 (1999), págs. 531-537.

[64] Cedric Gondro. *Primer to Analysis of Genomic Data Using R*. Springer International Publishing, 2015. DOI: 10.1007/978-3-319-14475-7.

[65] Malachi Griffith et al. «Informatics for RNA Sequencing: A Web Resource for Analysis on the Cloud». En: *PLOS Computational Biology* 11.8 (2015), págs. 1-20. DOI: 10.1371/journal.pcbi.1004393.

[66] Manuel Ugidos Guerrero. «Estudio del perfil de expresión de microRNA en pacientes de cáncer de mama». Tesis de mtría. Universidad de Valencia, 2017.

[67] Florian Hahne et al. *Bioconductor Case Studies*. Use R! Springer, 2008.

[68] Sonja Hanzelmann, Robert Castelo y Justin Guinney. «GSVA: gene set variation analysis for microarray and RNA-Seq data». En: *BMC Bioinformatics* 14.1 (2013), pág. 7. ISSN: 1471-2105. DOI: 10.1186/1471-2105-14-7.

[69] T. Hastie, R. Tibshirani y M. Wainwright. *Statistical Learning with Sparsity. The Lasso and Generalizations*. Monographs on Statistics and Applied Probability 143. CHAPMAN & HALL/CRC, 2015.

[70] Masaaki Horikoshi y Yuan Tang. *ggfortify: Data Visualization Tools for Statistical Analysis Results*. R package version 0.4.17. 2024.

[71] Melanie A. Huntley et al. «ReportingTools: an automated result processing and presentation toolkit for high-throughput genomic analyses». En: *Bioinformatics* 29.24 (2013), págs. 3220-3221. ISSN: 1367-4803. DOI: 10.1093/bioinformatics/btt551.

[72] R.A. Irizarry et al. «Exploration, normalization, and summaries of high density oligonucleotide array probe level data». En: *Biostatistics* 4.2 (2003), págs. 249-264. DOI: 10.1093/biostatistics/4.2.249. eprint: http://biostatistics.oxfordjournals.org/content/4/2/249.full.pdf+html.

[73] R.A. Irizarry et al. «Gene set enrichment analysis made simple». En: *Statistical Methods in Medical Research* 18.6 (2009). Gene set analysis, págs. 565-575.

[74] Rafael A. Irizarry et al. *affy: Methods for Affymetrix Oligonucleotide Arrays*. R package version 1.85.1. 2025. DOI: 10.18129/B9.bioc.affy.

[75] Arnoud J. Kal et al. «Dynamics of Gene Expression Revealed by Comparison of Serial Analysis of Gene Expression Transcript Profiles from Yeast Grown on Two Different Carbon Sources». En: *Molecular Biology of the Cell* 10.6 (1999), págs. 1859-1872. DOI: 10.1091/mbc.10.6.1859. eprint: http://www.molbiolcell.org/content/10/6/1859.full.pdf+html.

[76] Audrey Kauffmann y Wolfgang Huber. *arrayQualityMetrics: Quality metrics report for microarray data sets*. R package version 3.63.0. 2024. DOI: 10.18129/B9.bioc.arrayQualityMetrics.

[77] Audrey Kauffmann et al. *ArrayExpress: Access the ArrayExpress Collection at EMBL-EBI Biostudies and build Bioconductor data structures: ExpressionSet, AffyBatch, NChannelSet*. R package version 1.67.1. 2025. DOI: 10.18129/B9.bioc.ArrayExpress.

[78] L. Kaufman y P. J. Rousseeuw. *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, 1990.

[79] Matt Kline. *Modern LaTeX*. BookPatch LLC, The, 2022. ISBN: 9798885674485.

[80] Eija Korpelainen et al. *RNA-seq Data Analysis A Practical Approach*. CRC Press, 2015.

[81] Nan M. Laird y Christoph Lange. *The Fundamentals of Modern Statistical Genetics*. Springer New York, 2011. DOI: 10.1007/978-1-4419-7338-2.

[82] Jun Li y Robert Tibshirani. «Finding consistent patterns: A nonparametric approach for identifying differential expression in RNA-Seq data». En: *Statistical Methods in Medical Research* 22.5 (2013), págs. 519-536. DOI: 10.1177/0962280211428386. eprint: http://smm.sagepub.com/content/22/5/519.full.pdf+html.

[83] Xiaochun Li. *ALL: A data package*. R package version 1.49.0. 2024. DOI: 10.18129/B9.bioc.ALL.

[84] Q. Liu et al. «Comparative evaluation of gene-set analysis methods». En: *BMC Bioinformatics* 8.1 (2007), págs. 1-15. ISSN: 1471-2105. DOI: 10.1186/1471-2105-8-431.

[85] Michael Love. *parathyroidSE: RangedSummarizedExperiment for RNA-Seq of primary cultures of parathyroid tumors by Haglund et al., J Clin Endocrinol Metab 2012.* R package version 1.45.1. 2024. DOI: 10.18129/B9.bioc.parathyroidSE.

[86] Michael Love, Wolfgang Huber y Simon Anders. «Moderated estimation of fold change and dispersion for RNA-seq data with DESeq2». En: *Genome Biology* 15.12 (2014), pág. 550. ISSN: 1465-6906. DOI: 10.1186/s13059-014-0550-8.

[87] A. T. L. Lun, Y. Chen y G. K. Smyth. «It's DE-licious: A Recipe for Differential Expression Analyses of RNA-seq Experiments Using Quasi-Likelihood Methods in edgeR.» En: *Statistical Genomics. Methods and Protocols*. Ed. por E. Mathé y S. Davis. Vol. 1418. 2016. Cap. 19. DOI: 10.1007/978-1-4939-3578-9_19.

[88] Steven P. Lund et al. «Detecting Differential Expression in RNA-sequence Data Using Quasi-likelihood with Shrunken Dispersion Estimates». En: *Statistical Applications in Genetics and Molecular Biology* 11.5 (2012). DOI: doi:10.1515/1544-6115.1826.

[89] James W. MacDonald. *affycoretools: Functions useful for those doing repetitive analyses with Affymetrix GeneChips*. R package version 1.79.0. 2024. DOI: 10.18129/B9.bioc.affycoretools.

[90] Henryk Maciejewski. «Gene set analysis methods: statistical models and methodological differences». En: *Briefings in Bioinformatics* (2013). DOI: 10.1093/bib/bbt002. eprint: http://bib.oxfordjournals.org/content/early/2013/02/09/bib.bbt002.full.pdf+html.

[91] S. C. Madeira y A. L. Oliveira. «Biclustering algorithms for biological data analysis: a survey». En: *IEEE/ACM Transactions on Computational Biology and Bioinformatics* 1.1 (2004), págs. 24-45. DOI: 10.1109/TCBB.2004.2.

[92] Martin Maechler et al. *cluster: "Finding Groups in Data": Cluster Analysis Extended Rousseeuw et al.* R package version 2.1.8.1. 2025.

[93] Bryan F. J. Manly. *Randomization, Bootstrap and Monte Carlo Methods in Biology*. Taylor & Francis Ltd, 152006. 480 págs. ISBN: 1584885416.

[94] Paolo Martini et al. «Along signal paths: an empirical gene set approach exploiting pathway topology». En: *Nucleic Acids Research* 41.1 (2013), e19. DOI: 10.1093/nar/gks866. eprint: http://nar.oxfordjournals.org/content/41/1/e19.full.pdf+html.

[95] Davis J. McCarthy, Yunshun Chen y Gordon K. Smyth. «Differential expression analysis of multifactor RNA-Seq experiments with respect to biological variation». En: *Nucleic Acids Research* 40.10 (2012), págs. 4288-4297. DOI: 10.1093/nar/gks042. eprint: http://nar.oxfordjournals.org/content/40/10/4288.full.pdf+html.

[96] P. McCullagh y John A. Nelder. *Generalized Linear Models*. Taylor & Francis Ltd, 11989. 532 págs. ISBN: 0412317605.

[97] Peter McCullagh. «Quasi-Likelihood Functions». En: *The Annals of Statistics* 11.1 (1983), págs. 59-67. DOI: 10.1214/aos/1176346056.

[98] Qinxue Meng et al. «DBNorm: normalizing high-density oligonucleotide microarray data based on distributions». En: *BMC Bioinformatics* 18.1 (2017). DOI: 10.1186/s12859-017-1912-5.

[99] V.K . Mootha et al. «PGC-1alpha-responsive genes involved in oxidative phosphorylation are coordinately downregulated in human diabetes». En: *Nature Genetics* 34 (2003), págs. 267-73.

[100] Martin Morgan. *ShortRead: FASTQ input and manipulation*. R package version 1.65.0. 2024. DOI: 10.18129/B9.bioc.ShortRead.

[101] Martin Morgan, Seth Falcon y Robert Gentleman. *GSEABase: Gene set enrichment data structures and methods*. R package version 1.69.1. 2025. DOI: 10.18129/B9.bioc.GSEABase.

[102] Martin Morgan y Marcel Ramos. *BiocManager: Access the Bioconductor Project Package Repository*. R package version 1.30.25. 2024.

[103] Martin Morgan et al. *Rsamtools: Binary alignment (BAM), FASTA, variant call (BCF), and tabix file import*. R package version 2.23.1. 2024. DOI: 10.18129/B9.bioc.Rsamtools.

[104] Martin Morgan et al. *SummarizedExperiment: A container (S4 class) for matrix-like assays*. R package version 1.37.0. 2024. DOI: 10.18129/B9.bioc.SummarizedExperiment.

[105] Ali Mortazavi et al. «Mapping and quantifying mammalian transcriptomes by RNA-Seq». En: *Nat Meth* 5.7 (2008), págs. 621-628. ISSN: 1548-7091.

[106] Haroon Naeem et al. «Rigorous assessment of gene set enrichment tests». En: *Bioinformatics* 28.11 (2012), págs. 1480-1486. DOI: 10.1093/bioinformatics/bts164. eprint: http://bioinformatics.oxfordjournals.org/content/28/11/1480.full.pdf+html.

[107] Richard M. Neve et al. «A collection of breast cancer cell lines for the study of functionally distinct cancer subtypes». En: *Cancer cell* 10.6 (2006), págs. 515-527. DOI: 10.1016/j.ccr.2006.10.008.

[108] Michael A. Newton et al. «Random-set methods identify distinct aspects of the enrichment signal in gene-set analysis». En: *Annals of Appied Statistics* 1.1 (2007), págs. 85-106. DOI: 10.1214/07-AOAS104.

[109] Tobias Oetiker et al. *La introducción no-tan-corta a LATeX 2e*. 2010.

[110] Assaf Oron y Robert Gentleman. *GSEAlm: Linear Model Toolset for Gene Set Enrichment Analysis*. R package version 1.67.0. 2024. DOI: 10.18129/B9.bioc.GSEAlm.

[111] A. Oshlack y M. J. Wakefield. «Transcript length bias in RNA-seq data confounds systems biology». En: *Biol Direct* 4.1 (2009), págs. 1-10. ISSN: 1745-6150. DOI: 10.1186/1745-6150-4-14.

[112] Alicia Oshlack, Mark Robinson y Matthew Young. «From RNA-seq reads to differential expression results». En: *Genome Biology* 11.12 (2010), pág. 220. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-12-220.

[113] Hervé Pagès et al. *AnnotationDbi: Manipulation of SQLite-based annotations in Bioconductor*. R package version 1.69.1. 2025. DOI: 10.18129/B9.bioc.AnnotationDbi.

[114] J. K. Pickrell et al. «Understanding mechanisms underlying human gene expression variation with RNA sequencing». En: *Nature* 464 (2010). DOI: 10.1038/nature08872.

[115] Katherine S. Pollard et al. *multtest: Resampling-based multiple hypothesis testing*. R package version 2.63.0. 2024. DOI: 10.18129/B9.bioc.multtest.

[116] «Preliminary Report: Findings from the Aspirin Component of the Ongoing Physicians' Health Study». En: *New England Journal of Medicine* 318.4 (1988), págs. 262-264. DOI: 10.1056/nejm198801283180431.

[117] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing. Vienna, Austria, 2025.

[118] Yasir Rahmatallah, Frank Emmert-Streib y Galina Glazko. «Comparative evaluation of gene set analysis approaches for RNA-Seq data». En: *BMC Bioinformatics* 15.1 (2014), págs. 1-15. ISSN: 1471-2105. DOI: 10.1186/s12859-014-0397-8.

[119] C.R. Rao. *Linear Statistical Inference and Its Applications*. Wiley, 1967.

[120] Anat Reiner, Daniel Yekutieli y Yoav Benjamini. «Identifying differentially expressed genes using false discovery rate controlling procedures». En: *Bioinformatics* 19.3 (2003), págs. 368-375. DOI: 10.1093/bioinformatics/btf877. eprint: http://bioinformatics.oxfordjournals.org/content/19/3/368.full.pdf+html.

[121] Tyler Rinker y Dason Kurkiewicz. *pacman: Package Management Tool*. R package version 0.5.1. 2019.

[122] David Robinson, Alex Hayes y Simon Couch. *broom: Convert Statistical Objects into Tidy Tibbles*. R package version 1.0.8. 2025.

[123] M. D. Robinson y A. Oshlack. «A scaling normalization method for differential expression analysis of RNA-seq data». En: *Genome Biol* 11.3 (2010), R25. ISSN: 1465-6906. DOI: 10.1186/gb-2010-11-3-r25.

[124] Mark D. Robinson y Gordon K. Smyth. «Moderated statistical tests for assessing differences in tag abundance». En: *Bioinformatics* 23.21 (2007), págs. 2881-2887. DOI: 10.1093/bioinformatics/btm453. eprint: http://bioinformatics.oxfordjournals.org/content/23/21/2881.full.pdf+html.

[125] Mark D. Robinson y Gordon K. Smyth. «Small-sample estimation of negative binomial dispersion, with applications to SAGE data». En: *Biostatistics* 9.2 (2008), págs. 321-332. DOI: 10.1093/biostatistics/kxm030. eprint: http://biostatistics.oxfordjournals.org/content/9/2/321.full.pdf+html.

[126] Gabriele Sales et al. «Graphite Web: web tool for gene set analysis exploiting pathway topology». En: *Nucleic Acids Research* (2013). DOI: 10.1093/nar/gkt386. eprint: http://nar.oxfordjournals.org/content/early/2013/05/10/nar.gkt386.full.pdf+html.

[127] P.P. Sinha. *Bioinformatics with R Cookbook*. Packt Publishing, 2014.

[128] Gordon Smyth et al. *limma: Linear Models for Microarray and Omics Data*. R package version 3.63.13. 2025. DOI: 10.18129/B9.bioc.limma.

[129] Gordon K. Smyth. «Linear Models and Empirical Bayes Methods for Assessing Differential Expression in Microarray Experiments». En: *Statistical Applications in Genetics and Molecular Biology* 1 (2004), pág. 3.

[130] Peter H.A. Sneath y Rober R. Sokal. *Numerical Taxonomy. The principes and practice of numerical Classification.* San Francisco, USA: W.H. Freeman y Company, 1973.

[131] Phillip Stafford, ed. *Methods in Microarray Normalization*. CRC Press, 2008.

[132] John D. Storey y Robert Tibshirani. «Statistical significance for genomewide studies». En: *Proceedings of the National Academy of Sciences* 100.16 (2003), págs. 9440-9445. DOI: 10.1073/pnas.1530509100. eprint: http://www.pnas.org/content/100/16/9440.full.pdf+html.

[133] John D. Storey et al. *qvalue: Q-value estimation for false discovery rate control*. R package version 2.39.0. 2024. DOI: 10.18129/B9.bioc.qvalue.

[134] Aravind Subramanian et al. «Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles». En: *Proceedings of the National Academy of Sciences of the United States of America* 102.43 (2005), págs. 15545-15550. DOI: 10.1073/pnas.0506580102. eprint: http://www.pnas.org/content/102/43/15545.full.pdf+html.

[135] Dan Tenenbaum y Bioconductor Package Maintainer. *KEGGREST: Client-side REST access to the Kyoto Encyclopedia of Genes and Genomes (KEGG)*. R package version 1.47.1. 2025. DOI: 10.18129/B9.bioc.KEGGREST.

[136] Laura A. Thomson. *R (and S-PLUS) Manual to Accompany Agresti's Categorical Data Analysis (2002) 2nd edition*. 2009.

[137] Lu Tian et al. «Discovering statistically significant pathways in expression profiling studies». En: *Proceedings of the National Academy of Sciences of the United States of America* 102.38 (2005), págs. 13544-13549. DOI: 10.1073/pnas.0506577102. eprint: http://www.pnas.org/content/102/38/13544.full.pdf+html.

[138] Shailesh Tripathi, Galina V. Glazko y Frank Emmert-Streib. «Ensuring the statistical soundness of competitive gene set approaches: gene filtering and genome-scale coverage are essential». En: *Nucleic Acids Research* 41.7 (2013), e82. DOI: 10.1093/nar/gkt054. eprint: http://nar.oxfordjournals.org/content/41/7/e82.full.pdf+html.

[139] Olga Troyanskaya et al. «Missing value estimation methods for DNA microarrays». En: *Bioinformatics* 17.6 (2001), págs. 520-525. DOI: 10.1093/bioinformatics/17.6.520. eprint: http://bioinformatics.oxfordjournals.org/content/17/6/520.full.pdf+html.

[140] Virginia Goss Tusher, Robert Tibshirani y Gilbert Chu. «Significance analysis of microarrays applied to the ionizing radiation response». En: *Proceedings of the National Academy of Sciences* 98.9 (2001). Gene set analysis, págs. 5116-5121. DOI: 10.1073/pnas.091062498. eprint: http://www.pnas.org/content/98/9/5116.full.pdf+html.

[141] Marie Verbanck, Sebastien Le y Jerome Pages. «A new unsupervised gene clustering algorithm based on the integration of biological knowledge into expression data». En: *BMC Bioinformatics* 14.1 (2013), pág. 42. ISSN: 1471-2105. DOI: 10.1186/1471-2105-14-42.

[142] John Verzani. *UsingR: Data Sets, Etc. for the Text "Using R for Introductory Statistics", Second Edition*. R package version 2.0-7. 2022. DOI: 10.32614/CRAN.package.UsingR.

[143] Zhong Wang, Mark Gerstein y Michael Snyder. «RNA-Seq: a revolutionary tool for transcriptomics». En: *Nat Rev Genet* 10.1 (2009), págs. 57-63. ISSN: 1471-0056.

[144] R. W. M. Wedderburn. «Quasi-likelihood functions, generalized linear models, and the Gauss—Newton method». En: *Biometrika* 61.3 (1974), págs. 439-447. DOI: 10.1093/biomet/61.3.439.

[145] H. Wickham. *Advanced R*. CRC, 2014.

[146] Hadley Wickham. *R Packages: Organize, Test, Document, and Share Your Code*. O'Reilly Media, 2015. ISBN: 978-1-49191-059-7.

[147] Hadley Wickham. *reshape: Flexibly Reshape Data*. R package version 0.8.9. 2022.

[148] Hadley Wickham. «Reshaping Data with the reshape Package». En: *Journal of Statistical Software* 21.1 (2007), págs. 1-20. ISSN: 1548-7660. DOI: 10.18637/jss.v021.i12.

[149] Hadley Wickham. «Tidy Data». En: *Journal of Statistical Software* 59.1 (2014), págs. 1-23. ISSN: 1548-7660. DOI: 10.18637/jss.v059.i10.

[150] Hadley Wickham et al. *ggplot2: Create Elegant Data Visualisations Using the Grammar of Graphics*. R package version 3.5.2. 2025.

[151] Hadley Wickham et al. *roxygen2: In-Line Documentation for R*. R package version 7.2.3. 2022.

[152] G. N. Wilkinson y C. E. Rogers. «Symbolic Description of Factorial Models for Analysis of Variance». En: *Applied Statistics* 22.3 (1973), pág. 392. DOI: 10.2307/2346786.

[153] E. Wit y J.D. McClure. *Statistics for microarrays: design, analysis, and inference*. Wiley, 2004.

[154] Daniela M Witten y Robert Tibshirani. «Scientific research in the age of omics: the good, the bad, and the sloppy». En: *Journal of the American Medical Informatics Association* 20.1 (2013), págs. 125-127. DOI: 10.1136/amiajnl-2012-000972. eprint: http://jamia.bmj.com/content/20/1/125.full.pdf+html.

[155] Di Wu y Gordon K. Smyth. «Camera: a competitive gene set test accounting for inter-gene correlation». En: *Nucleic Acids Research* 40.17 (2012), e133. DOI: 10.1093/nar/gks461. eprint: http://nar.oxfordjournals.org/content/40/17/e133.full.pdf+html.

[156] Jean Wu y Rafael Irizarry with contributions from James MacDonald Jeff Gentry. *gcrma: Background Adjustment Using Sequence Information*. R package version 2.79.0. 2024. DOI: 10.18129/B9.bioc.gcrma.

[157] Jianguo Xia, Erin E Gill y Robert E W Hancock. «NetworkAnalyst for statistical, visual and network-based meta-analysis of gene expression data». En: *Nat. Protocols* 10.6 (2015), págs. 823-844. ISSN: 1754-2189.

[158] Yihui Xie. *Dynamic Documents with R and knitr*. 2nd. Chapman & Hall/CRC The R Series. Chapman y Hall/CRC, 2015.

[159] Yihui Xie. *knitr: A General-Purpose Package for Dynamic Report Generation in R*. R package version 1.50. 2025.

[160] Yihui Xie, Joe Cheng y Xianying Tan. *DT: A Wrapper of the JavaScript Library DataTables*. R package version 0.33. 2024.

[161] Qing Xiong et al. «Integrating genetic and gene expression evidence into genome-wide association analysis of gene sets». En: *Genome Research* 22.2 (2012), págs. 386-397. DOI: 10.1101/gr.124370.111. eprint: http://genome.cshlp.org/content/22/2/386.full.pdf+html.

[162] Matthew Young y Nadia Davidson. *geneLenDataBase: Lengths of mRNA transcripts for a number of genomes*. R package version 1.43.0. 2024. DOI: 10.18129/B9.bioc.geneLenDataBase.

[163] Guangchuang Yu. *clusterProfiler: A universal enrichment tool for interpreting omics data*. R package version 4.15.1. 2024. DOI: 10.18129/B9.bioc.clusterProfiler.

[164] Yi-Hui Zhou, William T. Barry y Fred A. Wright. «Empirical pathway analysis, without permutation». En: *Biostatistics* (2013). DOI: 10.1093/biostatistics/kxt004. eprint: http://biostatistics.oxfordjournals.org/content/early/2013/02/20/biostatistics.kxt004.full.pdf+html.

[165] Hao Zhu. *kableExtra: Construct Complex Table with kable and Pipe Syntax*. R package version 1.4.0. 2024.

Índice alfabético

Affymetrix
  Fichero CDF, 10
  Fichero CEL, 10
  Fichero DAT, 10
  hgu133a.db, 233
Affymetrix GeneChip, 9
ALL, 38, 236
annotate, 342
  annotation(), 342
  getSYMBOL(), 343
  lookUp(), 343
AnnotationDbi
  select(), 49
Análisis cluster, 5
Análisis de la varianza, 112, 125

base
  apply, 85
  as.character(), 344
Biobase
  AnnotatedDataFrame, 35
  ExpressionSet, 32
    exprs, 91
    exprs(), 29
    fData(), 30, 47
    pData(), 29
    phenoData(), 28
    sampleNames, 28
    varMetadata(), 28
  sample.ExpressionSet, 27
Bioconductor, 379
  affy, 41
  ALL, 379
  Biobase, 27
  BiocManager, 379
  GSEABase, 220
  multtest
    golub, 362

Category, 232
Coeficiente de correlación múltiple, 118
Coeficiente de determinación, 118
Coeficiente de determinación ajustado, 118
coldata, 66
Comparaciones múltiples (multiple comparisons), 149
Control débil del error, 152
Control fuerte del error, 152
CpG, 77

Datos
  GEO
    GSE21779, 40
  GSE37211, 68
  gse44456, 185
  GSE64099, 67
Desviación, 133
Desviación escalada, 133
Diagrama de cajas, 91
Distribución binomial, 128
Distribución normal, 129
Distribución Poisson, 129
DT, 344
  datatable(), 344

edgeR
  binomTest, 192
  binomTest(), 192
  TMM
    calcNormFactors, 74, 75
emacs, 340
  ESS Emacs Speaks Statistics, 340
  polymode, 340
  Quarto, 340
EnrichmentBrowser, 238
  deAna(), 238
  gsRanking(), 239
  makeSummarizedExperiment-FromExpressionSet(), 238
  sbea(), 239
Ensembl, 343
Error cuadrático medio, 271
Error estándar, 271
Espacio paramétrico, 270
Especificidad, 284
Estimador, 271
Estimador máximo verosímil, 273
ExpressionSet

featureNames(), 342

Familia de dispersión exponencial, 128
Familia exponencial natural, 128
FDR
Tasa de falsamente rechazados (false discovery rate), 151
Tasa de falso rechazo positivo (Positive false discovery rate), 152
Filtrado independiente, 84
Fold-change, 89
Función de enlace, 130
Función logit, 129
FWER
tasa de error global (familywise error rate, 151

Gene Ontology: GO, 343
Gene set analysis, 5
Gene set enrichment analysis, 5
genefilter, 87
kOverA, 87
nsFilter, 88
nsFilter(), 85
rowFtests(), 125
rowttests, 93, 157, 342
GenomicRanges
assay, 65
coldata, 66
rowRanges, 66
GEOquery, 44
getGEOSuppFiles, 44
ggplot2, 40
geom$_boxplot()$, 91
qplot
facets, 365
GO.db, 233
golubEsets
golub, 38
GOstats, 232
hyperGTest, 233
GSE1397, 42
gse183019, 404
GSE20986, 44, 220, 234, 342
gse20986, 342
gse21443, 403
gse21942, 263
gse25171, 175, 176
GSE34764, 45
GSE37211, 65
GSE64099, 67
GSEABase
GeneSetCollection, 224
GSEAbase, 220
details, 221
geneIds, 220
GeneSet, 220
GeneSetCollection(), 221

hgu133plus2.db, 234, 342, 343

IDE
RStudio, 357
Intervalo de confianza, 276

knitr, 339

limma
contrasts.fit(), 187
eBayes(), 186
geneSetTest(), 250
lmFit(), 186
makeContrasts(), 187
normalizeBetweenArrays(), 26, 27
topTable(), 186
wilcoxGST(), 251
limma::contrasts.fit(), 188
limma::eBayes(), 188
logit, 137
Logverosimilitud, 269

MA plot, 14
Markdown, 339
MAS5, 18
Matriz de correlaciones muestral, 273
Matriz de covarianzas muestral, 273
Matriz hat, matriz de influencia, 115
max T por pasos de bajada, 156
Median polish, 23
Microarray, 9
Modelo lineal generalizado, 128
Componente aleatoria, 128
Componente sistemática, 128, 130
Desviación, 132
Ecuaciones de estimación, 131
Enlace canónico, 130
Estimadores máximo verosímiles, 131
Función de enlace, 128, 130
Función respuesta, 130
Residuos de la desviación, 133
Residuos de Pearson, 133
Modelo lineal generalizado GLM, 127
multtest
golub, 36

mt.rawp2adjp(), 235
Método de Benjamini y Hochberg, 157
Método de Bonferroni, 155
Método de Hochberg, 156
Método de Holm, 156
Método de Šídák, 155

Nivel de confianza, 277
Normalización de cuantiles, 21
nsFilter, 235

org.Sc.sgd.db, 222

p-valor ajustado (adjusted p-value), 154
Pandoc, 339
Paquete R
parathyroidSE, 65
Parámetro de dispersión, 128
Parámetro natural, 128
PRJNA297664, 264

q-valor, 158
Quarto, 339
qvalue, 160

R
all.equal(), 402
as.matrix, 33
base
apply, 366
class, 363
colnames, 366
dim, 364
factor, 363
help, 357
help.start, 357
mean, 367
ncol, 364
nrow, 364
rownames, 366
sort, 359
browseURL(), 237, 264
class, 33
data.frame, 33
fisher.test, 231
function, 371
grep, 312
Instalación, 355
library(), 356
Listas, 369
Logical Operators, 359
matplot, 12
Matriz inversa, 116
model.matrix(), 116
Paquetes
ggplot2, 363
UsingR, 356
paste0, 370
Producto de matrices, 116
read.csv, 32
read.table, 32
resid(), 116
return, 371
sapply, 34
save(), 41
setwd, 41
Short-refcard, 356
solve(), 116
stats
fisher.test(), 230
table, 363
typeof(), 28
which, 359
Región crítica, 275
Regresión logística, 137
Error de clasificación, 139
Función de enlace, 135
Regresión Poisson, 142
Regresión probit, 137
ReportingTools, 344
finish(), 344
HTMLReport(), 344
htmlReport(), 236
publish(), 344
reshape
melt, 12, 40
Residuo de Pearson, 133
RMarkdown, 339
Robust multichip average (RMA), 20
rowRanges, 66
RStudio, 340, 357

S4, 28
SAM, 165, 166
samr, 168
Selección no específica, 84
Sensibilidad, 284
Sesgo, 271
stats
ks.test(), 98
na.omit(), 344
p.adjust, 158
quantile(), 85
t.test(), 93
Sumas de cuadrados
Regresión, 117

Residual, 117
Total, 117
SummarizedExperiment
RangedSummarizedExperiment, 65

t-test, 92
Tabla de contingencia, 283
tami
ensembl2url(), 344
entrezid2url(), 344
GeneSetTest(), 264
glimpse(), 264
go2url(), 344
kegg2url(), 344
tidy(), 264
tamidata
gse20986, 234
gse21779, 49
gse21942, 90, 238
gse44456, 185
PRJNA297664, 73, 75
TCGAbiolinks
GDCdownload, 69
GDCprepare, 69
GDCquery, 69
Test de Fisher, 236
Test de Fisher unilateral, 229
Test de Kolmogorov-Smirnov, 98
Test de Wald, 276
Test del cociente de verosimilitudes, 275
TMM, 72
topGO, 266
Transformación integral de la probabilidad, 22

Verosimilitud, 269

Glosario

Affymetrix http://www.affymetrix.com/. 36, 224, 346, 385, 386

Agilent https://www.agilent.com/. 379

ArrayExpress https://www.ebi.ac.uk/biostudies/arrayexpress. Es una colección de bancos de datos para estudios de genómica funcional dependiente del EBI.. 27, 40, 42

ASCII Sistema de codificación de caracteres: American Standard Code for Information Interchange. https://en.wikipedia.org/wiki/ASCII. 57

Bash https://en.wikipedia.org/wiki/Bash_(Unix_shell). xi, 339

Bioconductor Red de espejos con paquetes de R para análisis de datos ómicos: https://www.bioconductor.org/. xi–xiii, 28, 36, 48, 54, 65, 175, 220, 249, 350, 379, 381

BioMart http://www.biomart.org Es un sistema de almacenamiento de datos orientado a las consultas. Ha sido desarrollado por el European Bioinformatics Institute (EBI) y el Cold Spring Harbor Laboratory (CSHL). . 396

Bowtie2 Alineador de secuencias. http://bowtie-bio.sourceforge.net/bowtie2/index.shtml . 58–61

CDS Es la región codificante de un gen: **C**oding **D**NA **S**equence. Es la parte del gen (DNA o RNA) compuesta por los exones que codifican proteína. El CDS es la porción de un transcrito que es trasladado por un ribosoma. . 393, 394

CpG La metilación del DNA es una modificación en donde se produce una modificación de una base citosina para formar una metilcitosina. Esta modificación ocurre casi exclusivamente cuando la citosina está seguida por una guanina en la dirección 5' a 3'. Esto se denota como **C**pG. https://en.wikipedia.org/wiki/CpG_site#CpG_islands. 77

CRAN Red de espejos con paquetes de R: https://cran.r-project.org/ . 379

DDBJ https://www.ddbj.nig.ac.jp/index-e.html. 55

Debian Una distribución Linux. Posiblemente la mejor de todas ellas y la que está debajo de otras muchas. Consultar https://www.debian.org/ y https://en.wikipedia.org/wiki/Debian.. xiii, 426, 427

emacs Editor de textos que puede ser configurado para trabajar con R y Markdown o LaTeXy otros lenguajes de programación. https://www.gnu.org/software/emacs/ . 339, 340

EMBL https://www.embl.org/. 55

Ensembl https://en.wikipedia.org/wiki/Ensembl. 51, 224, 343, 344, 346, 388, 398–400, 402

Entrez https://en.wikipedia.org/wiki/Entrez. 49, 222, 224, 227, 343, 346, 386, 388, 398–400

ExpressionSet Clase S4 para almacenar datos de expresión de microarrays. Definida en [53].. 88, 306, 369

FASTA https://en.wikipedia.org/wiki/FASTA_format. 55

FASTQ Es un formato para guardar datos de secuencias. Consiste de cuatro líneas. La primera contiene el nombre de la secuencia. La segunda línea contiene a la propia secuencia. La tercera línez contiene información opcional sobre la secuencia. La cuarta línea cuantifica la confianza o calidad en la determinación de cada base recogida en la segunda línea. https://en.wikipedia.org/wiki/FASTQ_format . 56, 60, 62

Gene Ontology Abreviadamente GO. Definen conceptos o clases que son utilizados para describir la función de un gen y relaciones entre estos conceptos. Se clasifican las funciones atendiendo a tres aspectos: MF o función molecular que describe las actividades moleculares de los productos de los genes; CC o componente celular indicando dónde los productos de los genes son activos; BP o procesos biológicos que indican las rutas y procesos indicando actividad de los productos de varios genes: https://en.wikipedia.org/wiki/Gene_ontology y en http://geneontology.org/. Los términos GO están organizados en un grafo acíclico dirigido. . 35, 219, 221, 222, 224, 226–228, 233–235, 238, 240, 243, 244, 247, 255, 257, 326, 343, 344, 346, 388, 389, 400

GEO NCBI Gene Expression Omnibus. Su dirección es http://www.ncbi.nlm.nih.gov/geo/. Es un repositorio público con datos experimentales de alto rendimiento. Tenemos experimentos basados en microarrays de uno o dos canales que miden mRNA, DNA genómico, presencia de proteinas. También hay otras técnicas no basadas de arrays como análisis serial de expresión de genes (SAGE), datos proteómicos obtenidos son espectrometría de masas o datos de secuenciación de alto rendimiento. Hay cuatro tipos de entidades básicas en GEO: Sample, Platform, Series, DataSet. En esta sección mostramos cómo obtener datos de esta gran base de datos pública. Los datos que nos bajamos los utilizaremos en los siguientes temas. Cuando accedemos a una entrada de GEO podemos ver el enlace **Analyze with GEO2R**. Si accedemos a este enlace podemos ver que nos ofrece algunos análisis de los datos utilizando R/Bioconductor.. 10, 26, 27, 40, 41, 44, 55, 65, 67, 78

GTEx Herramienta de análisis en línea que incluye análisis transcriptómico y estudios de asociación genética. Los datos son propios de la herramienta. https://www.gtexportal.org/home/.. 6

HTML https://en.wikipedia.org/wiki/HTML. 264, 339, 341, 425

i.i.d. Independientes y con la misma distribución. En definitiva que tenemos una muestra aleatoria de una variable o vector aleatorio.. 272, 279

IDAT https://datatypes.net/open-idat-files. 78

KEGG Es una colección de mapas de rutas representando interacciones moleculares y grafos de interacción. Estas rutas cubren muchos procesos bioquímicos que se pueden dividir en: metabolismo, proceso de información genética, proceso de información medioambiental, procesos celulares, sistemas de organismos, enfermedades humanas, desarrollo de medicamentos: http://www.genome.jp/kegg/, https://en.wikipedia.org/wiki/KEGG. . 35, 219, 228, 240, 258, 344

Markdown Es un lenguaje de marcas ligero (minimal sería más correcto). Pretende ser una forma rápida de escribir HTML. Según su autor, John Gruber, "HTML is a publishing format; Markdown is a writing format." La dirección indicada contiene una exposición de este lenguaje de marcas. Al ser tan limitado en su sintaxis han aparecido diferentes variaciones que lo extienden (entre otras cosas para tablas) de las cuales la más interesante y usada es Pandoc Markdown.. xii

muestra aleatoria S X denota una variable o vector aleatorio, se dice que $X_1, \ldots, X_n$ es una muestra aleatoria si son independientes y con la misma distribución que X.. 92

NCBI Es una colección de bases de datos y utilizados en línea para el análisis de datos ómicos. También contiene el Pubmed. http://www.ncbi.nlm.nih.gov https://en.wikipedia.org/wiki/National_Center_for_Biotechnology_Information National Center for Biotechnology Information. . 55

NCBI-SRA NCBI SRA (**S**equence **Read A**rchive) es una base de datos con datos de secuenciación de DNA en forma de lecturas cortas generadas mediante secuenciación de alto rendimiento. Su dirección es http://www.ncbi.nlm.nih.gov/sra. En http://www.ncbi.nlm.nih.gov/books/NBK47528/ tenemos más información. Si no funciona el enlace buscamos en Google *SRA handbook*.. 55, 60, 65, 426

Pandoc Es un programa que convierte entre distintos lenguajes de marcas inicialmente desarrollado por John McFarlane. En particular, los paquetes [159], [8, rmarkdown] y Quarto lo utilizan. https://pandoc.org/.. 426

pdf https://en.wikipedia.org/wiki/PDF.. 341

Python https://en.wikipedia.org/wiki/Python_(programming_language). xi

Quarto Es un sistema de escritura de documentos científicos en la forma de documentos dinámicos y que utiliza Pandoc y se puede https://quarto.org/.. xii, 339, 340, 425

R R es un entorno de programación estadístico. https://cran.r-project.org/, https://www.r-project.org/ . xi–xiii, 34, 36, 40, 41, 54, 60, 61, 102, 103, 105, 193, 220, 249, 279, 338–340, 350, 355, 373, 374, 379, 381, 426

Retraction Watch Revista en línea dedicada al estudio de los problemas en las publicaciones científicas: https://retractionwatch.com/.. 348

RMA Robust multichip average § 2.2.9. Ver § 2.2.9.. 39, 41, 42, 51

RMarkdown El paquete [8, rmarkdown] incorpora una implementación del lenguaje de marcas Markdown que utilizado conjuntamente con R permite generar informes. La página http://rmarkdown.rstudio.com/ es el mejor lugar para aprender a manejarlo. . 340

RNA-Seq https://en.wikipedia.org/wiki/RNA-Seq. . 53

RPKM Reads per kilobase per million reads. C es el total de lecturas alineadas sobre la característica. El total de lecturas que podemos alinear es N (o tamaño de la librería), L la longitud de la característica de interés en bp. $RPKM = \frac{10^9 C}{NL}$. . 72

SAM Sequence Aligment Map http://genome.sph.umich.edu/wiki/SAM y la especificación del formato la tenemos en http://samtools.sourceforge.net/SAM1.pdf o https://samtools.github.io/hts-specs/SAMv1.pdf.. 58, 59, 247

Samtools Es un conjunto de programas para trabajar con datos de secuenciación. Ver http://www.htslib.org/. En Debian/Ubuntu se instala con *a*pt-get install samtools. En R/Bioconductor tenemos el paquete [103, Rsamtools].. 60, 61

SRA-Toolkit Es un conjunto de funciones para trabajar con la base de datos NCBI-SRA. Para su instalación es conveniente consultar consultar https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=software así como la documentación en https://trace.ncbi.nlm.nih.gov/Traces/sra/sra.cgi?view=toolkit_doc. También es de interés https://github.com/ncbi/sra-tools/wiki/Downloads.. En Debian/Ubuntu hemos de instalar el paquete **sra-toolkit**.. 60

STAR Alineador de secuencias cortas. https://github.com/alexdobin/STAR. En Debian/Ubuntu tenemos el paquete rna-star.. 60, 61

t-test https://en.wikipedia.org/wiki/Student%27s_t-test. 111

tasa de falso rechazo (FDR) Tasa de falso rechazo (false discovery rate), Realizamos diversos contrastes de hipótesis. Denotamos por V el número de los falsamente rechazados (hipótesis nula cierta pero es rechazada) y por $Relnmerodehiptesisnulasrechazadas.sedefinelatasadefalsorechazooFDRcomo$FI = E $\left(\frac{V}{R}|R > 0\right) P(R > 0)$. . 152, 251, 342

TCGA **T**he **C**ancer **G**enome **A**tlas es una plataforma creada por el NCI (*National Cancer Institute*) y la National Human Genome Research Institute. Contiene bases de datos online con estudios de cáncer utilizando distintas técnicas https://cancergenome.nih.gov/. . 6, 69

Ubuntu Distribución Linux basada en la distribución Debian y que resulta de más fácil instalación. https://www.ubuntu.com/ y https://en.wikipedia.org/wiki/Ubuntu.. xiii, 426