

# Sobre qué es R y Bioconductor

Guillermo Ayala Gallego

## Sobre qué es R y Bioconductor

Guillermo Ayala Gallego

2025-02-27

## Introducción

### Instalando R

- Primero instalamos R
- Una vez instalado el R base la instalación de algun paquete adicional

```
1 install.packages("UsingR")
```

### R

- RStudio
  - emacs con el modo Emacs Speaks Statistics ESS
  - Instalación
- apt update apt install r-base r-base-dev apt install libatlas3-base

### Markdown

- Daring Fireball by John Gruber
- Markdown en español
- pandoc

### Instalando los paquetes de la asignatura

- tami
- Instalación
- Instalar la versión más reciente de R: Ubuntu, Debian.

- [Paquetes Debian/Ubuntu]

```
apt update
apt upgrade
apt install libcurl4 libcurl4-openssl-dev libssl-dev libxml2-dev
libcairo2-dev libxt-dev
```

## Una sola muestra

### Construyendo y jugando con un vector

```
1 x = c(3,6,3,6,7,1,4,7,4,4)
1 class(x)
[1] "numeric"
1 x
[1] 3 6 3 6 7 1 4 7 4 4
1 x[1]
[1] 3
1 3:7
[1] 3 4 5 6 7
1 x[3:7]
[1] 3 6 7 1 4
1 c(1,3,5:8)
[1] 1 3 5 6 7 8
1 x[c(1,3,5:8)]
[1] 3 3 7 1 4 7
1 x[x >= 4]
[1] 6 6 7 4 7 4 4
1 x[x > 6]
[1] 7 7
1 y = x[x > 4 & x <= 6]
```

### De cómo ordenar un vector

```
1 sort(x)
[1] 1 3 3 4 4 4 6 6 7 7
```

```

1 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

1 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

```

## Varias muestras

### Matrices

```

1 x = matrix(rpois(10*4,lambda=5),nrow=10)
1 class(x)
[1] "matrix" "array"
1 colnames(x) = c("sample1","sample2","sample3","sample4")
1 colnames(x) = paste0("sample",1:4)
1 rownames(x) = paste0("gene",1:10)
1 x

      sample1 sample2 sample3 sample4
gene1      7       6       4       6
gene2      6       4       4       3
gene3      2       6       9       8
gene4      5       4       4       4
gene5      5       6       7       5
gene6      6       5       9       7
gene7      2       3       7       7
gene8      7       4       4       4
gene9      4       3       2       6
gene10     2       8       5       6

```

La expresión del gen en la fila 6 y en la columna 2.

```

1 x[6,2]
[1] 5

```

```
1 x["gene6", "sample2"]
```

[1] 5

La expresión de los genes en la muestra 2 podemos hacerlo con

```
1 x[,2]
```

gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8	gene9	gene10
6	4	6	4	6	5	3	4	3	8

```
1 x[, "sample2"]
```

gene1	gene2	gene3	gene4	gene5	gene6	gene7	gene8	gene9	gene10
6	4	6	4	6	5	3	4	3	8

Borramos el espacio de trabajo.

```
1 rm(list=ls())
```

Datos golub

- 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).

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

- Hemos leído

1 golub.cl

¿Qué tipo de dato tenemos?

```
1 class(golub.cl)
```

```
[1] "numeric"
```

Es más cómodo trabajarla como **factor**.

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

Veamos un resumen.

```
1 summary(golub.fac)
```

ATT. AMT.

27 11

Gráficamente,

```

1 pacman::p_load(ggplot2)
2 df0 = data.frame(golub.fac)
3 ggplot(df0,aes(x=golub.fac))+geom_bar()
• Los datos de expresión están en el objeto golub.
1 class(golub)
[1] "matrix" "array"
• Veamos las expresiones de los primeros genes
1 head(golub,n=2)
[,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]      [,8]
[1,] -1.45769 -1.39420 -1.42779 -1.40715 -1.42668 -1.21719 -1.37386 -1.36832
[2,] -0.75161 -1.26278 -0.09052 -0.99596 -1.24245 -0.69242 -1.37386 -0.50803
      [,9]      [,10]     [,11]     [,12]     [,13]     [,14]     [,15]     [,16]
[1,] -1.47649 -1.21583 -1.28137 -1.03209 -1.36149 -1.39979  0.17628 -1.40095
[2,] -1.04533 -0.81257 -1.28137 -1.03209 -0.74005 -0.83161  0.41200 -1.27669
      [,17]     [,18]     [,19]     [,20]     [,21]     [,22]     [,23]     [,24]
[1,] -1.56783 -1.20466 -1.24482 -1.60767 -1.06221 -1.12665 -1.20963 -1.48332
[2,] -0.74370 -1.20466 -1.02380 -0.38779 -1.06221 -1.12665 -1.20963 -1.12185
      [,25]     [,26]     [,27]     [,28]     [,29]     [,30]     [,31]     [,32]
[1,] -1.25268 -1.27619 -1.23051 -1.43337 -1.08902 -1.29865 -1.26183 -1.44434
[2,] -0.65264 -1.27619 -1.23051 -1.18065 -1.08902 -1.05094 -1.26183 -1.25918
      [,33]     [,34]     [,35]     [,36]     [,37]     [,38]
[1,] 1.10147 -1.34158 -1.22961 -0.75919  0.84905 -0.66465
[2,] 0.97813 -0.79357 -1.22961 -0.71792  0.45127 -0.45804
• El número de filas corresponde con el número de genes.
1 nrow(golub)
[1] 3051
• El número de columnas corresponde con el número de individuos o muestras.
1 ncol(golub)
[1] 38
O las dimensiones de la matriz.
1 dim(golub)
[1] 3051   38

Un perfil de expresión
El nivel de expresión del gen que está en la fila 2000 y ha sido observado en la muestra 3 (columna 3).
1 golub[2000,3]

```

```
[1] 0.51981
```

Y todos los niveles de este gen lo tendremos con

```
1 golub[2000,]  
[1] 0.73124 -0.19598 0.51981 -0.54371 0.55596 1.40683 0.79772 0.59493  
[9] 0.99503 0.39529 0.09834 0.19595 0.85017 -1.39979 1.09789 -0.74362  
[17] 0.44207 0.27698 -0.04128 -1.60767 -1.06221 -1.12665 0.47863 -0.44014  
[25] 0.22286 0.42795 0.65427 0.07257 -0.28093 -0.20985 0.05160 -1.44434  
[33] -0.17118 -1.34158 0.92325 -0.21462 -1.34579 1.17048
```

A estos niveles los llamaremos **perfil de expresión**.

```
1 muestra = 1:ncol(golub) ## En abscisas el número de la muestra  
2 y2000 = golub[2000,] ## En ordenadas su expresión
```

## Dibujando el perfil de expresión

```
1 golub[2000,12]  
[1] 0.19595  
1 golub[2000,]  
[1] 0.73124 -0.19598 0.51981 -0.54371 0.55596 1.40683 0.79772 0.59493  
[9] 0.99503 0.39529 0.09834 0.19595 0.85017 -1.39979 1.09789 -0.74362  
[17] 0.44207 0.27698 -0.04128 -1.60767 -1.06221 -1.12665 0.47863 -0.44014  
[25] 0.22286 0.42795 0.65427 0.07257 -0.28093 -0.20985 0.05160 -1.44434  
[33] -0.17118 -1.34158 0.92325 -0.21462 -1.34579 1.17048  
1 pacman::p_load(ggplot2)  
2 muestra = 1:ncol(golub) ## En abscisas el número de la muestra  
3 y2000 = golub[2000,] ## En ordenadas su expresión  
4 df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,])  
5 ggplot(df,aes(x = muestra,y= y2000))+geom_point()  
1 golub[2000,golub.fac == "ALL"]  
[1] 0.73124 -0.19598 0.51981 -0.54371 0.55596 1.40683 0.79772 0.59493  
[9] 0.99503 0.39529 0.09834 0.19595 0.85017 -1.39979 1.09789 -0.74362  
[17] 0.44207 0.27698 -0.04128 -1.60767 -1.06221 -1.12665 0.47863 -0.44014  
[25] 0.22286 0.42795 0.65427  
1 golub[2000,golub.fac == "AML"]  
[1] 0.07257 -0.28093 -0.20985 0.05160 -1.44434 -0.17118 -1.34158 0.92325  
[9] -0.21462 -1.34579 1.17048  
1 df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],  
2 tipo = golub.fac)  
3 ggplot(df,aes(x = muestra,y= y2000,colour=tipo))+geom_point() +
```

```

4     xlab("Número de muestra") + ylab("Gen en fila 2000") +
5     facet_grid(tipo~.)
1 df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],
2                  tipo = golub.fac)
3 ggplot(df,aes(x = muestra,y= y2000,colour=tipo))+geom_point() +
4     xlab("Número de muestra") + ylab("Gen en fila 2000") +
5     facet_grid(tipo~.)
1 df = data.frame(muestra = 1:ncol(golub),y2000= golub[2000,],
2                  tipo = golub.fac)
3 ggplot(df,aes(x = muestra,y= y2000,colour=tipo))+geom_point() +
4     xlab("Número de muestra") + ylab("Gen en fila 2000") +
5     facet_grid(. ~ tipo) #Modificamos esta línea

```

## De qué genes estamos hablando

```

1 golub.gnames[2000,]
[1] "4544"           "CDC21 HOMOLOG" "X74794_at"
1 class(golub.gnames)
[1] "matrix" "array"
1 dim(golub.gnames)
[1] 3051      3
1 golub.gnames[2000,2]
[1] "CDC21 HOMOLOG"
1 golub.gnames[2000,3]
[1] "X74794_at"
1 colnames(golub) = golub.fac

```

## apply

```

1 mean(golub[2000,])
[1] 0.02080211
1 mean(golub[2000,])
[1] 0.02080211
    • Seleccionamos las submatrices correspondientes a cada leucemia.
1 golub.ALL = golub[,golub.fac == "ALL"]
2 golub.AML = golub[,golub.fac == "AML"]
    • Calculamos la media y la mediana para cada submatriz de golub.

```

```

1 ALL.mean = apply(golub[,golub.fac == "ALL"],1,mean)
2 AML.mean = apply(golub[,golub.fac == "AML"],1,mean)
3 ALL.median = apply(golub[,golub.fac == "ALL"],1,median)
4 AML.median = apply(golub[,golub.fac == "AML"],1,median)

```

```

1 df = data.frame(x0 = ALL.mean, y0 = AML.mean)
2 p = ggplot(df,aes(x=x0,y=y0))+geom_point()
3 p + xlab("Medias ALL") + ylab("Medias AML")

```

Medias de ALL (abscisas) frente a las medias AML (ordenadas)

## Listas

### Listas

- Tenemos un grupo de 10 genes que podemos identificar por un número.

```

1 seq_len(10)
[1] 1 2 3 4 5 6 7 8 9 10

• Tenemos tres grupos.

1 geneset1 = c(1,4)
2 geneset2 = c(2,5,7,8)
3 geneset3 = c(1,5,6)

1 (genesets = vector("list",3))
[[1]]
NULL

[[2]]
NULL

[[3]]
NULL

1 genesets[[1]] = geneset1
2 genesets[[2]] = geneset2
3 genesets[[3]] = geneset3

1 genesets
[[1]]
[1] 1 4

[[2]]

```

```

[1] 2 5 7 8

[[3]]
[1] 1 5 6

1 names(genesets) = paste0("set",1:3)
1 genenames = paste0("gene",1:10)
1 genesets[[1]] = genenames[geneset1]
2 genesets[[2]] = genenames[geneset2]
3 genesets[[3]] = genenames[geneset3]

```

### **lapply y sapply**

- ¿Qué longitud tiene cada grupo de genes?

La primera opción con `lapply`.

```

1 lapply(genesets,function(i) length(i))

$set1
[1] 2

$set2
[1] 4

$set3
[1] 3

```

Notar que hemos utilizado una función **anónima**.

Podemos conseguir un vector con

```

1 unlist(lapply(genesets,function(i) length(i)))

set1 set2 set3
2     4     3

```

También lo podemos obtener con `sapply`.

```

1 sapply(genesets,function(i) length(i))

set1 set2 set3
2     4     3

```

## **Bioconductor**

### **Instalación**

- Y después Bioconductor

```
1 install.packages("BiocManager")
2 BiocManager::install(version = "3.16")
• Una vez instalado la parte básica de Bioconductor podemos instalar algún paquete adicional (como ALL).
1 BiocManager::install("ALL", version = "3.16", dependencies=TRUE)
```