# Análisis de conjuntos de genes

Guillermo Ayala Gallego

5/5/23

## Table of contents

# El problema

## Planteamiento

- Estudiamos si hay relación entre **conjuntos de genes** previamente definidos y fenotipo.
- Estos grupos vendrán definidos según distintos criterios.
- Por ejemplo, corresponder a una ruta metabólica,
  localizados en un mismo cromosoma o bien definidos utilizando términos de Gene Ontology o KEGG.
- De un modo genérico: un conjunto de genes que represente un un hecho biológico relevante.

## Diferencias con la aproximación gen-a-gen o marginal

- 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 tests múltiples nos produce una clasificación en genes significativos y no significativos, un valor de corte de la lista.
- En esta aproximación **no usamos ningún conocimiento previo sobre relaciones conocidas entre genes**.

## Conjunto(s) de genes

- $G = \{1, \dots, N\}$ es el conjunto total de genes considerado (o universo de genes) entonces los conjuntos de genes serán $S_1, \dots, S_K$.
- El conjunto $S_k$ tiene cardinal $n_{S_k}$.

### ¿Hay asociación entre un conjunto de genes y el fenotipo?

- Es una pregunta muy vaga.
- **Competitive test**
  - Hypothesis Q1: The genes in a gene set show the same pattern of associations with the phenotype compared with the rest of the genes.
- **Self-contained test**
  - Hypothesis Q2: The gene set does not contain any genes whose expression levels are associated with the phenotype of interest.

## Ejemplos

### Un ejemplo simulado de Efron y Tibshirani: et1

- 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 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.

```
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
y.et = factor(rep(1:2,rep(25,2)),levels = 1:2,
              labels = c("Normal","Tratamiento"))
```

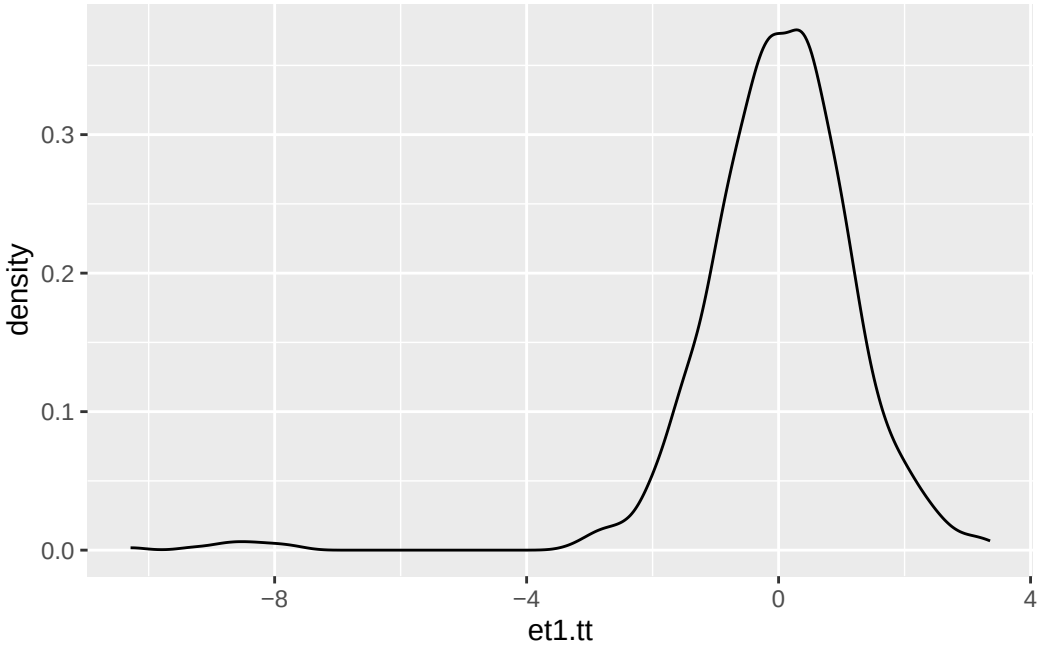
Cuantificamos una medida de asociación fenotipo-expresión.

```
et1.tt = genefilter::rowttests(et1,y.et)$statistic
```

```
Warning: replacing previous import 'utils::findMatches' by
'S4Vectors::findMatches' when loading 'AnnotationDbi'
```

- En figura 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())
```




## Un ejemplo simulado de Efron y Tibshirani: et2

- Este ejemplo se define exactamente como el ejemplo et1 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).

```
N = 1000; n = 50
set.seed(280562)
```

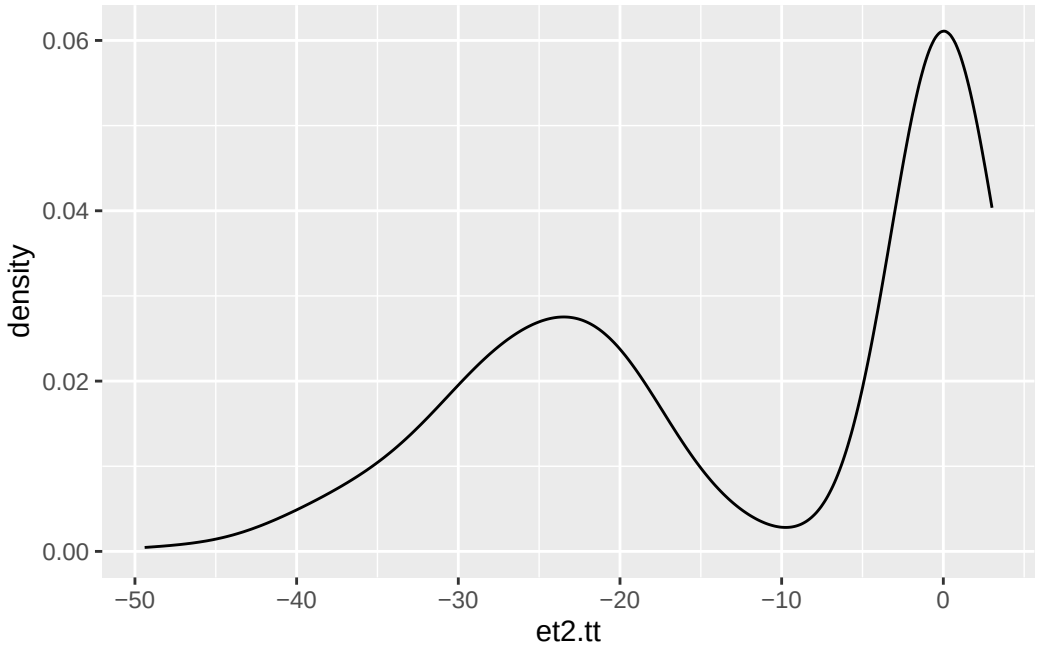
```
indices.temp  = (0:49)*20 + 1
indices = NULL
for(i in indices.temp) indices = c(indices,i:(i+9))
et2  = matrix(rnorm(N*n),nrow = N,ncol = n)
et2[indices,26:50] = et1[1:10,26:50] + 2.5
```

Cuantificamos la medida de asociación.

```
et2.tt = genefilter::rowttests(et2,y.et)$statistic
```

Vemos la densidad.

```
pacman::p_load("ggplot2")
df = data.frame(et2.tt)
(p = ggplot(df,aes(x=et2.tt))+geom_density())
```



### Grupos

```
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)
```

## Enriquecimiento

### Cuantificando asociación gen-fenotipo

- Esta cuantificación será un estadístico cuya definición dependerá del tipo de información fenotípica disponible.
- En lo que sigue a la medida de asociación gen-fenotipo la denotaremos genéricamente por $t_i$ (aunque no sea un t estadístico).

### Enriqueciendo el conjunto de genes

- El vector $t = (t_1, \dots, t_N)$ constituye una descripción de la asociación **marginal** gen-fenotipo.
- Enriquecer el estadístico $t$ para el conjunto consiste en considerar un resumen de $t$ sobre $S$.
- Si denotamos el estadístico de enriquecimiento por

1. $t(S)$ algunas opciones son
$$t(S) = \sum_{i \in S} \frac{t_i}{n_S}.$$
2. Definimos:
   - $t^+ = \max\{t, 0\}$
   - $t^- = -\min\{t, 0\}$
   - $\bar{t}_S^+ = \sum_{i \in S} \frac{t_i^+}{n_S}$
   - $\bar{t}_S^- = \sum_{i \in S} \frac{t_i^-}{n_S}$ La medida de enriquecimiento **maxmean** es
$$t(S) = \max\{\bar{t}_S^+, \bar{t}_S^-\}$$
   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.

## Distribuciones condicionadas a los datos

### Aleatorizando columnas (muestras)

- Para un gen tenemos su perfil de expresión $u = (u_1, \dots, u_n)$ y tenemos el vector $y$ asociado a las muestras.
- Sea $\pi$ denota una permutación aleatoria de $(1, \dots, n)$ de forma que los $\pi(i)$ será el índice que ocupa la posición $i$-ésima en la permutación $\pi$.
- Denotaremos por $\pi_0$ 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 $y$) $y_\pi = (y_{\pi(1)}, \dots, y_{\pi(n)})$.
- Obviamente, $y = y_{\pi_0}$.
- El valor observado de la asociación gen-fenotipo para $u$ e $y$ será $t_{\pi_0}$.
- Si consideramos $B$ permutaciones aleatorias entonces tendremos los valores $t_{\pi_1}, \dots, t_{\pi_B}$ para las $B$ permutaciones aleatorias.
- Si no hay asociación gen-fenotipo entonces el valor de $t_{\pi_0}$ debe de ser **como** los valores $t_{\pi_1}, \dots, t_{\pi_B}$.
- De hecho, cualquier ordenación de $t_{\pi_0}, t_{\pi_1}, \dots, t_{\pi_B}$
  tiene la misma probabilidad.
- El p-valor sería la proporción de $t_{\pi_b}$s mayores que $t_{\pi_0}$, es decir,

$$p_r = \frac{|\{b : t_{\pi_b} > t_{\pi_0}\}|}{B+1}.$$

- 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

$$p_r = \frac{|\{b : |t_{\pi_b}| > |t_{\pi_0}|\}|}{B+1},$$

  ya que tendríamos un test bilateral.

### Aleatorizando filas (genes o características)

- Denotemos por $S_0$ el conjunto de genes original en el que tenemos interés.
- Elegimos 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 $t(S)$ (ahora las muestras tienen su ordenación original).
- A la distribución de probabilidad de $t(S)$ se le llama **distribución de aleatorización** del estadístico de enriquecimiento.
- Tomamos $B$ selecciones aleatorias de $n_{S_0}$ genes.
- Tenemos los conjuntos $S_b$ con $b = 1, \dots, B$ y los valores del estadístico de enriquecimiento observados son $t(S_0)$ (para el grupo original) y $t(S_b)$ con $b = 1, \dots, B$ para los seleccionados al azar.
- El p-valor se define análogamente como

$$p_c = \frac{|\{b : t(S_b) > t(S_0)\}|}{B+1}.$$

  si solamente rechazamos para valores grandes (positivos).
- En el caso bilateral tendremos

$$p_{ae} = \frac{|\{b : |t(S_b)| > |t(S_0)|\}|}{B+1}.$$

## ¿Cómo hacerlo?

```
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 p-valor será

```
sum(tb > t0) / B+1
```

```
[1] 1
```

## Realizando un análisis de grupos de genes

### Limma::genSetTest

- 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.

```
pacman::p_load(limma)
geneSetTest(1:20,et1.tt,alternative ="up")
```

```
[1] 0.9984722
```

¿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.

### Limma::wilcoxGST

- 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.

- 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.
- Asume independencia entre estos estadísticos que *no* se verifica ya que las expresiones de los genes son interdependientes.

```
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
```

```
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
```

### Colecciones de grupos de genes

- Tenemos que elegir una colección de grupos de genes
- El paquete `EnrichmentBrowser` tiene algunas funciones útiles para bajar colecciones de Gene Ontology (GO) y KEGG

```
pacman::p_load(EnrichmentBrowser,tamidata)
```

### Gene Ontology

- Construimos la colección de Gene Ontology para homo sapiens y para la levadura en la ontología `BP` (biological process) utilizando el paquete `GO.db`.

```
hsa_go = EnrichmentBrowser::getGenesets(org = "hsa",db="go",
                                        onto ="BP")
names(hsa_go) = sapply(names(hsa_go),
                       function(x)unlist(strsplit(x,split="_"))[1])
save(hsa_go,file=paste(dirTamiData,"hsa_go.rda",sep=""))
```

```
load("hsa_go.rda")
```

```
sce_go = getGenesets(org="sce", onto="BP")
names(sce_go) = sapply(names(sce_go),
                       function(x)unlist(strsplit(x,split="_"))[1])
save(sce_go,file="sce_go.rda")
```

```
load("sce_go.rda")
```

### Explorando las colecciones de genes

¿Qué tipo de datos tenemos?

```
class(hsa_go)
```

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

¿Cuántos grupos tenemos?

```
length(hsa_go)
```

```
[1] 12417
```

Podemos acceder a cada elemento de la lista con la posición

```
hsa_go[1]
```

```
$`GO:0000002`
 [1] "291"   "1890"  "4205"  "4358"  "4976"  "9361"  "10000" "55186" "80119"
[10] "84275" "92667"
```

o el nombre

```
hsa_go["GO:0000002"]
```

```
$`GO:0000002`
 [1] "291"   "1890"  "4205"  "4358"  "4976"  "9361"  "10000" "55186" "80119"
[10] "84275" "92667"
```

Los elementos que la componen los tenemos con

```
hsa_go[[1]]
```

```
 [1] "291"   "1890"  "4205"  "4358"  "4976"  "9361"  "10000" "55186" "80119"
[10] "84275" "92667"
```

o con

```
hsa_go[["GO:0000002"]]
```

```
 [1] "291"   "1890"  "4205"  "4358"  "4976"  "9361"  "10000" "55186" "80119"
[10] "84275" "92667"
```

Los primeros elementos de la lista son

```
head(hsa_go)
```

```
$`GO:0000002`
 [1] "291"   "1890"  "4205"  "4358"  "4976"  "9361"  "10000" "55186" "80119"
[10] "84275" "92667"

$`GO:0000003`
[1] "2796"   "2797"   "8510"   "286826"

$`GO:0000012`
 [1] "1161"      "2074"      "3981"      "7141"      "7515"      "23411"
 [7] "54840"     "54840"     "55775"     "55775"     "55775"     "200558"
[13] "100133315"

$`GO:0000017`
[1] "6523" "6523" "6523" "6524"

$`GO:0000018`
[1] "3575"  "3836"  "3838"  "9984"  "10189" "56916"

$`GO:0000019`
[1] "4361"  "10111"
```

### Using `lapply` and `sapply`

Si queremos conocer el número de elementos de cada grupo de genes tenemos

```
ngs = lapply(hsa_go,length)
```

¿Qué es `ngs`?

```
class(ngs)
unlist(ngs)
```

Otra opción para conocer el número de elementos de cada grupo sería

```
ngs = sapply(hsa_go,length)
```

Ahora `ngs` es

```
class(ngs)
```

```
[1] "integer"
```

Podemos seleccionar los grupos de genes con más de 10 elementos con

```
hsa_go[ngs>10]
```

## Análisis de gse21942 con paquete tami

### Cargamos paquetes y datos

```
library(Biobase) ## For microarrays
library(SummarizedExperiment) ## For RangedSummarizedExperiment
library(tami) ## Our own package
```

```
Attaching package: 'tami'
```

```
The following objects are masked from 'package:BiocGenerics':

    type, type<-
```

```
library(EnrichmentBrowser) ## For gene set collections
library(ggplot2) ## Plots
```

### Un primer análisis de gse21942

- Utilizamos t-test moderado (Limma) para el análisis de expresión diferencial marginal: `test = rowt`
- 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"`
- 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")
```

### Un segundo análisis

```
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")
```

### Informes con los resultados

- 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      adjp
GO:0000002 GO:0000002 -0.8156418 0.64646465 0.8237807
GO:0000012 GO:0000012 -1.7272039 0.08080808 0.1848677
GO:0000018 GO:0000018 -4.4991525 0.00000000 0.0000000
GO:0000027 GO:0000027 -1.8897751 0.00000000 0.0000000
GO:0000028 GO:0000028 -3.2247175 0.00000000 0.0000000
GO:0000038 GO:0000038  1.1246832 0.24242424 0.4172263
```

- Un informe en un fichero lo tenemos con

```
glimpse(gse21942_self)
```

```
Warning: replacing previous import 'utils::findMatches' by
'S4Vectors::findMatches' when loading 'AnnotationForge'
```

```
Registered S3 method overwritten by 'GGally':
  method from
  +.gg   ggplot2
```

```
glimpse(gse21942_comp)
```

- El fichero podemos generarlo y abrirlo con

```
browseURL(glimpse(gse21942_self))
browseURL(glimpse(gse21942_comp))
```

**PRJNA297664**

- Leemos los datos.

```
data(PRJNA297664,package = "tamidata")
```

```
PRJNA297664_self = GeneSetTest(x = PRJNA297664,y="treatment",gsc=sce_go,
         test = edgercommon,association="statistic",correction="BH",
```

```
          GeneNullDistr = "randomization",
          GeneSetNullDistr = "self-contained",
          alternative="two-sided",nmax = 100,
          id = "ENTREZID",descriptive=mean,
          foutput = "PRJNA297664_self")
```

```
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")
```

```
PRJNA297664_self_df = tidy(PRJNA297664_self)
PRJNA297664_comp_df = tidy(PRJNA297664_comp)
```

```
glimpse(PRJNA297664_self)
```

```
[1] "./reports/PRJNA297664_self.html"
```

```
glimpse(PRJNA297664_comp)
```

```
[1] "./reports/PRJNA297664_comp.html"
```

```
browseURL(glimpse(PRJNA297664_self))
browseURL(glimpse(PRJNA297664_comp))
```