Guía para la realización de análisis de enriquecimiento funcional en R (parte uno)

Simposio anual Ómicas - Modalidad talleres

1. Análisis de enriquecimiento funcional para una lista de genes única (Instalación y carga de archivos)

En esta sesión se realizará análisis de enriquecimiento funcional para una única lista de genes usando R.

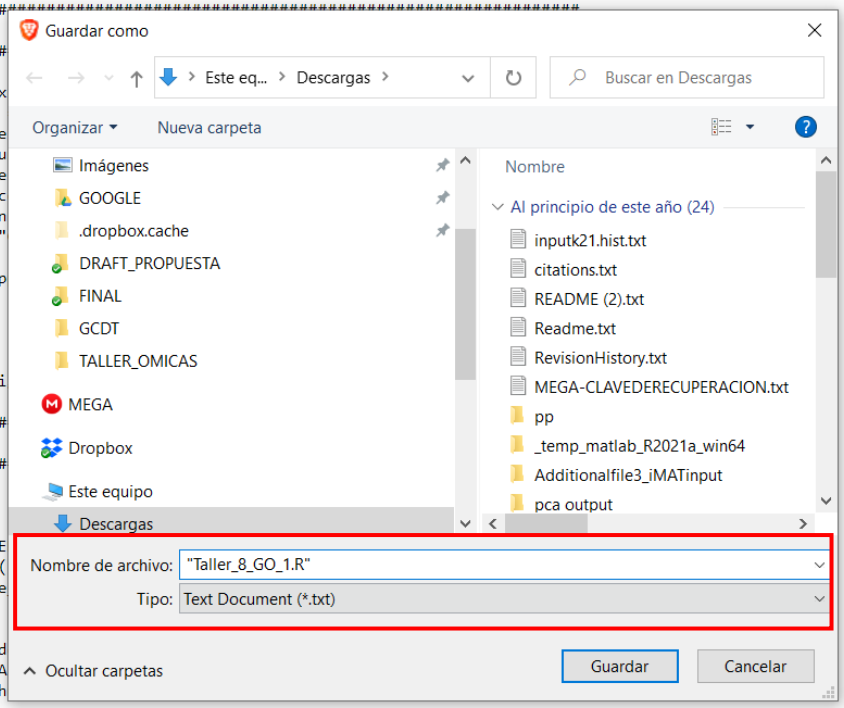
- El archivo para usar es el mismo que el que se ha usado en la sesión anterior. Este se encuentra disponible en el siguiente enlace: https://raw.githubusercontent.com/ccsosa/R_Examples/master/GIM.csv, sin embargo, se llamará directamente desde R.
- Para esta sesión necesitará instalar las siguientes librerías

I. **gprofiler2**
II. **biomaRt**
III. **topGO**
IV. **GOsummaries**
V. **Enrichplot**
VI. **org.Hs.eg.db**

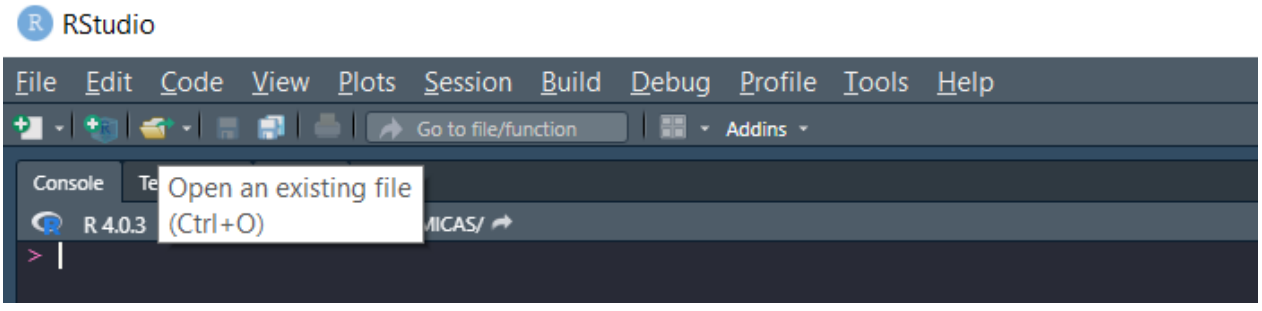
```
if (!requireNamespace("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("topGO")

BiocManager::install("biomaRt")

BiocManager::install("org.Hs.eg.db")

BiocManager::install("enrichplot")

install.packages("gprofiler2")

install.packages("ggnewscale")
```

• Para este ejercicio, el script con las instrucciones se puede descargar del enlace: https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/Taller_8_GO_1.R y puede descargarse localmente dando clic derecho y luego usando la opción Guardar como. Posteriormente guarde el archivo como "Taller_8_GO_1.R" entre comillas.

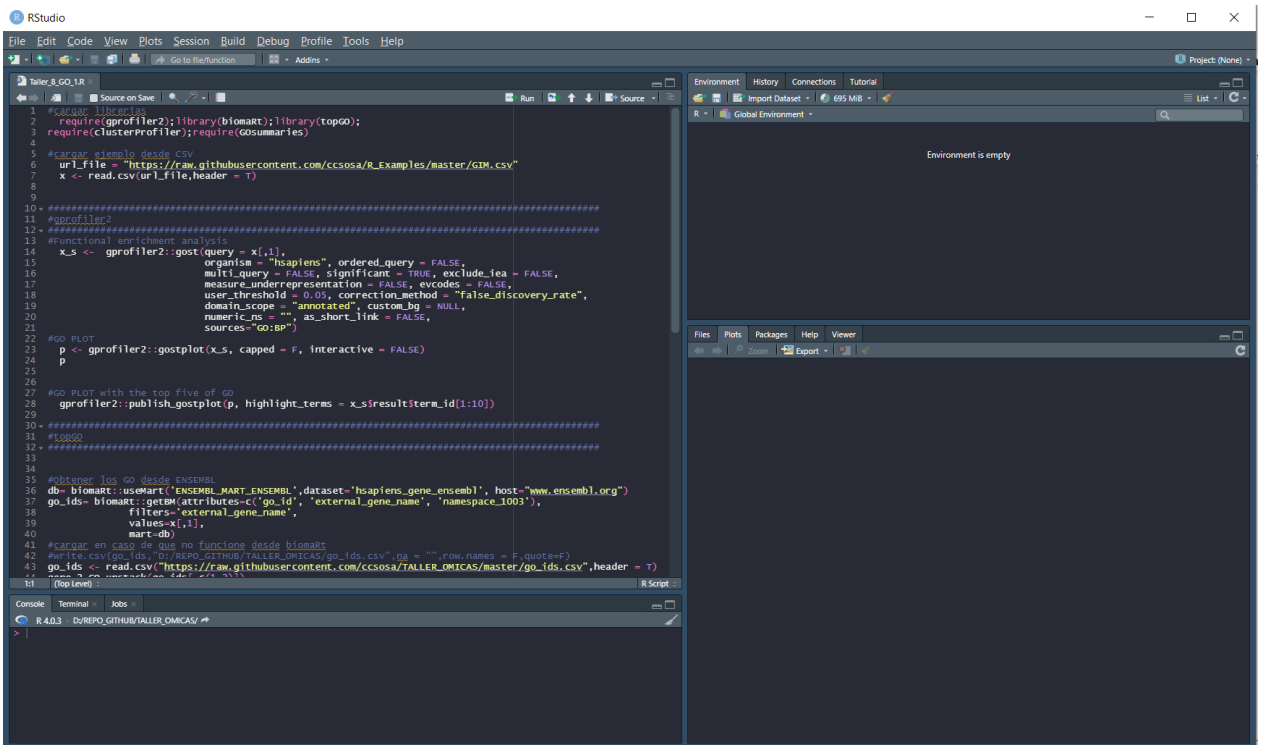
```
#cargar librerias
  require(gprofiler2);library(biomaRt);library(topGO);
require(clusterProfiler);require(GOsummaries)

#cargar ejemplo desde CSV
  url_file = "https://raw.githubusercontent.com/ccsosa/R_Examples/master/GIM.csv"
  x <- read.csv(url_file,header = T)


##########################################################################################
#gprofiler2
##################################
#Functional enrichment analysis
  x_s <-  gprofiler2::gost(query = x
                           organism
                           multi_que
                           measure_u
                           user_thre
                           domain_sc
                           numeric_n
                           sources="
#GO PLOT
  p <- gprofiler2::gostplot(x_s, cap
  p


#GO PLOT with the top five of GO
  gprofiler2::publish_gostplot(p, hi

##################################
#topGO
##################################


#Obtener los GO desde ENSEMBL
db= biomaRt::useMart('ENSEMBL_MART_E
go_ids= biomaRt::getBM(attributes=c(
              filters='external_gene
              values=x[,1],
              mart=db)
#cargar en caso de que no funcione d
#write.csv(go_ids,"D:/REPO_GITHUB/TA
go_ids <- read.csv("https://raw.gith
gene_2_GO=unstack(go_ids[,c(1,2)])
```



• Abra el archivo Taller_8_GO_1.R usando el icono 



• Al abrir el archivo con extensión .R en RStudio, este será abierto en el área del editor, como se ve a continuación:



• Para correr los análisis propuestos, primero se deben correr las líneas 2 y 3, en las cuales se cargarán las librerías gprofiler2, biomaRt, topGO, clusterProfiler y GOsummaries.

```
library (gprofiler2);library(biomaRt);library(topGO),

library (clusterProfiler); library (GOsummaries)
```

• Posteriormente se cargará el archivo con la lista de genes pertenecientes a inestabilidad genómica en humano que se encuentra localizado en GitHub (líneas 5 a la 7)

```
#cargar ejemplo desde CSV

 url_file = "https://raw.githubusercontent.com/ccsosa/R_Examples/master/GIM.csv"

 x <- read.csv(url_file,header = T)
```

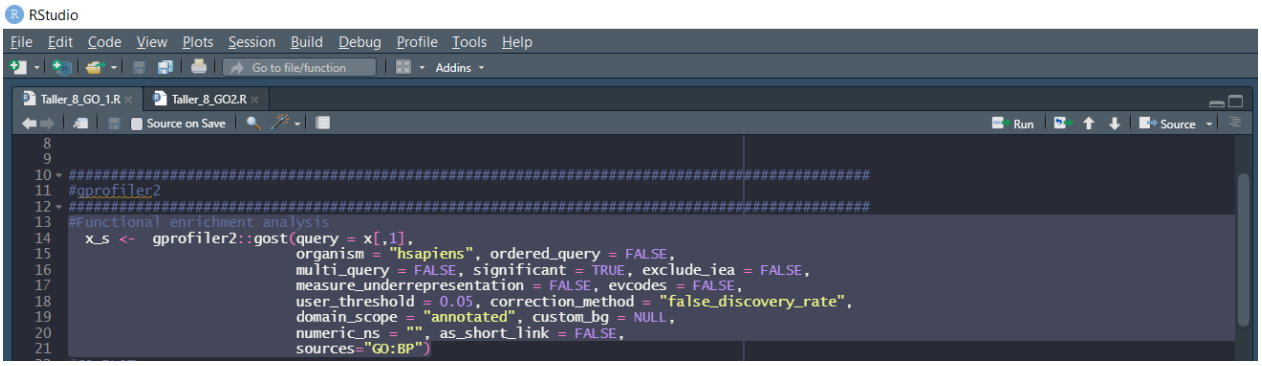
2. Análisis de enriquecimiento funcional para una lista de genes única

I. gprofiler2

• Ahora que ha cargado el archivo con la lista de genes, haremos un análisis de enriquecimiento funcional con el paquete gprofiler2, usaremos como organism= “hsapiens”, un punto de corte de 0.05, como método de corrección false_discovery_rate. Este análisis se guardará en el objeto x_s y se encuentra entre las líneas 13 a 21 del script

```
#Analisis de enriquecimiento funcional

 x_s <- gprofiler2::gost(query = x[,1],

             organism = "hsapiens", ordered_query = FALSE,

              multi_query = FALSE, significant = TRUE, exclude_iea = FALSE,

              measure_underrepresentation = FALSE, evcodes = FALSE,

              user_threshold = 0.05, correction_method = "false_discovery_rate",

              domain_scope = "annotated", custom_bg = NULL,

              numeric_ns = "", as_short_link = FALSE,

              sources="GO:BP")
```

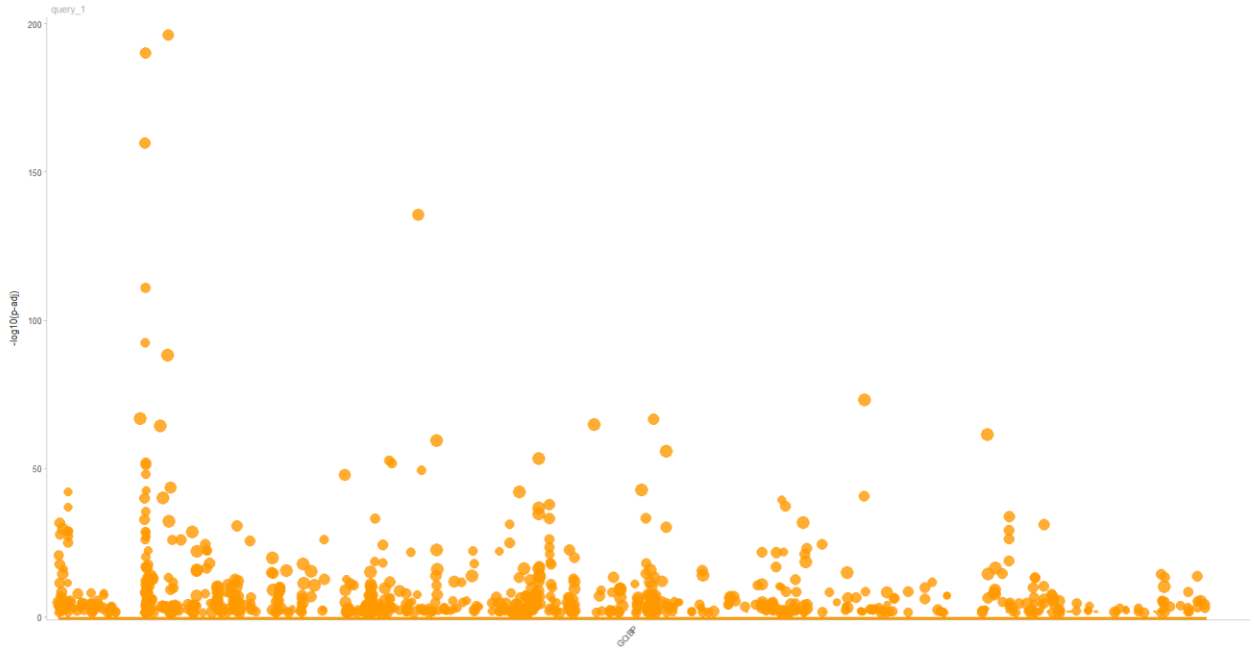
NOTA: Recuerde que para correr esta línea de código debe señarla y dar clic en Run, como se muestra abajo:



• Ahora haremos un *Manhattan plot,* en este gráfico se usa el $-log_{10}(p\ valor)$ para ver fácilmente qué términos GO tuvieron un valor de p más cercano a cero (líneas 22 a la 24)

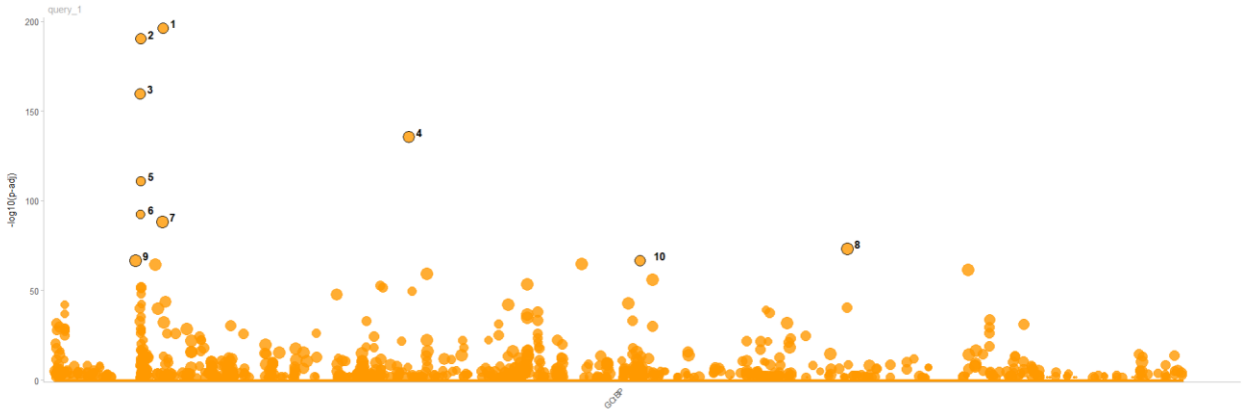
```
#GO PLOT

p <- gprofiler2::gostplot(x_s, capped = F, interactive = FALSE)

p
```

• El resultado del *Manhattan plot,* debe verse así, si quiere un gráfico interactivo, por favor cambie interactive=FALSE a interactive=TRUE.



• Si desea un gráfico en el que solo se muestre el top diez de GO enriquecidos, corra las líneas 27 a 29, como se muestra a continuación. El gráfico resultante mostrará que el término GO más enriquecido fue respuesta celular a daño de ADN.

```
#GO PLOT con el top diez

 gprofiler2::publish_gostplot(p, highlight_terms = x_s$result$term_id[1:10])

 p
```



id	source	term_id	term_name	term_size	p_value
1	GO:BP	GO:0006974	cellular response to DNA damage stimulus	894	8.0e-197
2	GO:BP	GO:0006281	DNA repair	586	5.7e-191
3	GO:BP	GO:0006259	DNA metabolic process	976	1.8e-160
4	GO:BP	GO:0033554	cellular response to stress	2142	3.2e-136
5	GO:BP	GO:0006289	nucleotide-excision repair	107	1.0e-111
6	GO:BP	GO:0006283	transcription-coupled nucleotide-excision repair	72	4.4e-93
7	GO:BP	GO:0006950	response to stress	4321	5.4e-89
8	GO:BP	GO:0090304	nucleic acid metabolic process	5234	8.5e-74
9	GO:BP	GO:0006139	nucleobase-containing compound metabolic process	5752	1.8e-67
10	GO:BP	GO:0051276	chromosome organization	1270	2.4e-67

g:Profiler (biit.cs.ut.ee/gprofiler)

II. topGO

• Para correr un análisis de enriquecimiento funcional en topGO, primero necesitará obtener los términos GO desde ENSEMBL, para esto deberá usar las líneas 35 a la 40 en donde se usará la librería biomaRt para obtener esa información. La primera parte del código permitirá establecer una conexión con el servidor de ENSEMBL. Ya que usaremos la información de humano, en el parámetro biomart, estará la opción 'ENSEMBL_MART_ENSEMBL' y en el parámetro dataset usaremos la opción 'hsapiens_gene_ensembl'.

NOTA: Si desea extraer la información en plantas, debe usar la siguiente línea de código en azul y cambie el dataset dependiendo de que especie guste usar.

```
db=     biomaRt::useMart(biomart="plants_mart",host="plants.ensembl.org",     dataset="athaliana_eg_gene")
```

```
#Obtener los GO desde ENSEMBL

db=           biomaRt::useMart('ENSEMBL_MART_ENSEMBL',dataset='hsapiens_gene_ensembl', host="www.ensembl.org")

go_ids= biomaRt::getBM(attributes=c('go_id', 'external_gene_name', 'namespace_1003'),

       filters='external_gene_name',

       values=x[,1],

       mart=db)
```

NOTA: Si en su computador surge un error a la hora de obtener la información de los GO asociados a la lista de genes, esto se debe a las restricciones que posee el muro de fuego de la universidad, por favor corra la siguiente línea de código (línea 42 a la 44) para corregirlo.

```
go_ids <- read.csv("https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/go_ids.csv",header = T)
```

- Corra la línea 46 para crear una lista de los términos GO por cada gen

```
gene_2_GO=unstack(go_ids[,c(1,2)])
```

- Las líneas 49 a la 54 le permitirán remover genes que no tengan anotaciones y marcar genes de interés como 1 en una lista

```
#Remover genes sin anotacion

keep = x[,1] %in% go_ids[,2]

keep =which(keep==TRUE)

candidate_list=x[,1][keep]

geneList=factor(as.integer(x[,1] %in% candidate_list),levels = c(0,1))

names(geneList)= x[,1]
```

- Posteriormente las líneas 56 a la 59 le permitirán crear un objeto, donde se obtendrán procesos biológicos mediante el parámetro ontology='BP'

```
#crear un objeto topGOdata

GOdata=new('topGOdata', ontology='BP', allGenes = geneList, annot = topGO::annFUN.gene2GO, gene2GO = gene_2_GO)
```

- Las líneas 63 a 72 le permitirán calcular las pruebas estadísticas de Fisher, Kosmogorov y eliminación combinada con Kosmogorov, respectivamente y, reportar los diez nodos más relevantes en una tabla resumen que puede llamar usando View(allRes)

```
#Run stastical tests

resultFisher <- runTest(GOdata, algorithm = "classic", statistic = "fisher")

resultKS <- runTest(GOdata, algorithm = "classic", statistic = "ks")

resultKS.elim <- runTest(GOdata, algorithm = "elim", statistic = "ks")

#summarize in a table

allRes <- topGO::GenTable(GOdata, classicFisher = resultFisher,

          classicKS = resultKS, elimKS = resultKS.elim,

          orderBy = "elimKS", ranksOf = "classicFisher", topNodes = 10)
```

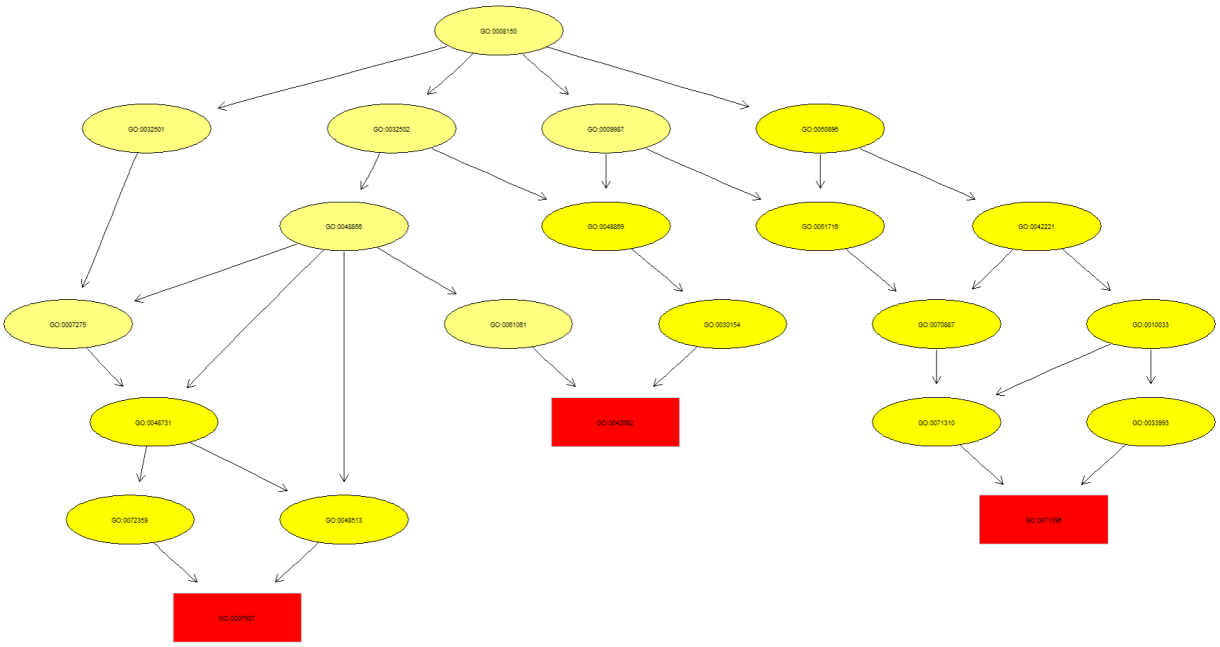
	GO.ID	Term	Annotated	Significant	Expected	Rank in classicFisher	classicFisher	classicKS	elimKS
1	GO:0042692	muscle cell differentiation	5	5	5	1	1	0.0019	0.0019
2	GO:0071396	cellular response to lipid	8	8	8	2	1	0.0021	0.0021
3	GO:0007507	heart development	4	4	4	3	1	0.0027	0.0027
4	GO:0051240	positive regulation of multicellular org...	24	24	24	4	1	0.0040	0.0040
5	GO:0022411	cellular component disassembly	7	7	7	5	1	0.0046	0.0046
6	GO:2000026	regulation of multicellular organismal d...	23	23	23	6	1	0.0063	0.0063
7	GO:0031124	mRNA 3'-end processing	4	4	4	7	1	0.0070	0.0070
8	GO:0034504	protein localization to nucleus	4	4	4	8	1	0.0075	0.0075
9	GO:0006611	protein export from nucleus	3	3	3	9	1	0.0077	0.0077
10	GO:2001251	negative regulation of chromosome organi...	12	12	12	10	1	0.0081	0.0081

- Si desea obtener los valores p asociados a las pruebas estadísticas realizadas podrá usar la función score, a su vez, podrá ver los resultados de todos los términos GO mediante la función termStat (líneas 75 a 78).

```
#diversas formas de obtener p valores

pValue.classic <- score(resultKS)

pValue.elim <- score(resultKS.elim)[names(pValue.classic)]

gstat <- topGO::termStat(GOdata, names(pValue.classic))
```

- El paquete topGO, le permitirá obtener un grafo dirigido con los nodos más significantes, en este caso usaremos tres (líneas 80 y 81).

```
par(cex = 1)

showSigOfNodes(GOdata, score(resultKS.elim), firstSigNodes = 3)
```



III.GOsummaries

- Una última opción para análisis de enriquecimiento funcional es presentada a través del paquete GOSummaries. Para esta opción usaremos los datos originales que se cargaron en el CSV y usaremos la función gosummaries (líneas 88-92).

```
#alternativa usando gosummaries

gl = list(GIM = x[,1]) # Two lists per component

gs = GOsummaries::gosummaries(gl)

plot(gs, fontsize = 8)
```