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

Simposio anual Ómicas - Modalidad talleres

1. Análisis de enriquecimiento funcional para varias listas de genes (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/TALLER_OMICAS/master/Hallmarks_of_Cancer_AT.csv sin embargo, se llamará directamente desde R.
- Para esta sesión necesitará instalar las siguientes librerías

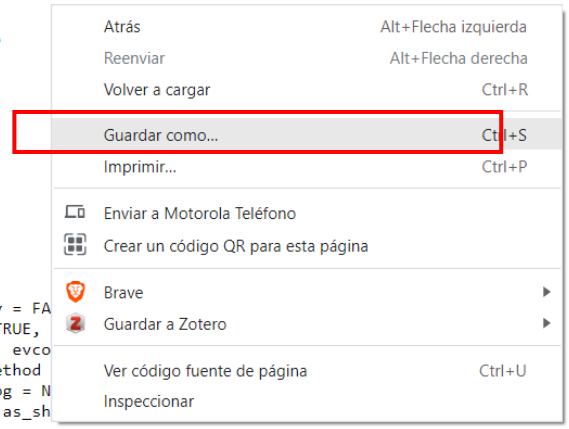
I. **gprofiler2**
II. **biomaRt**
III. **topGO**
IV. **clusterProfiler**
V. **GOsummaries**
VI. **Enrichplot**
VII. **org.Hs.eg.db**
VIII. **GOCompare**
IX. **ggplot2**

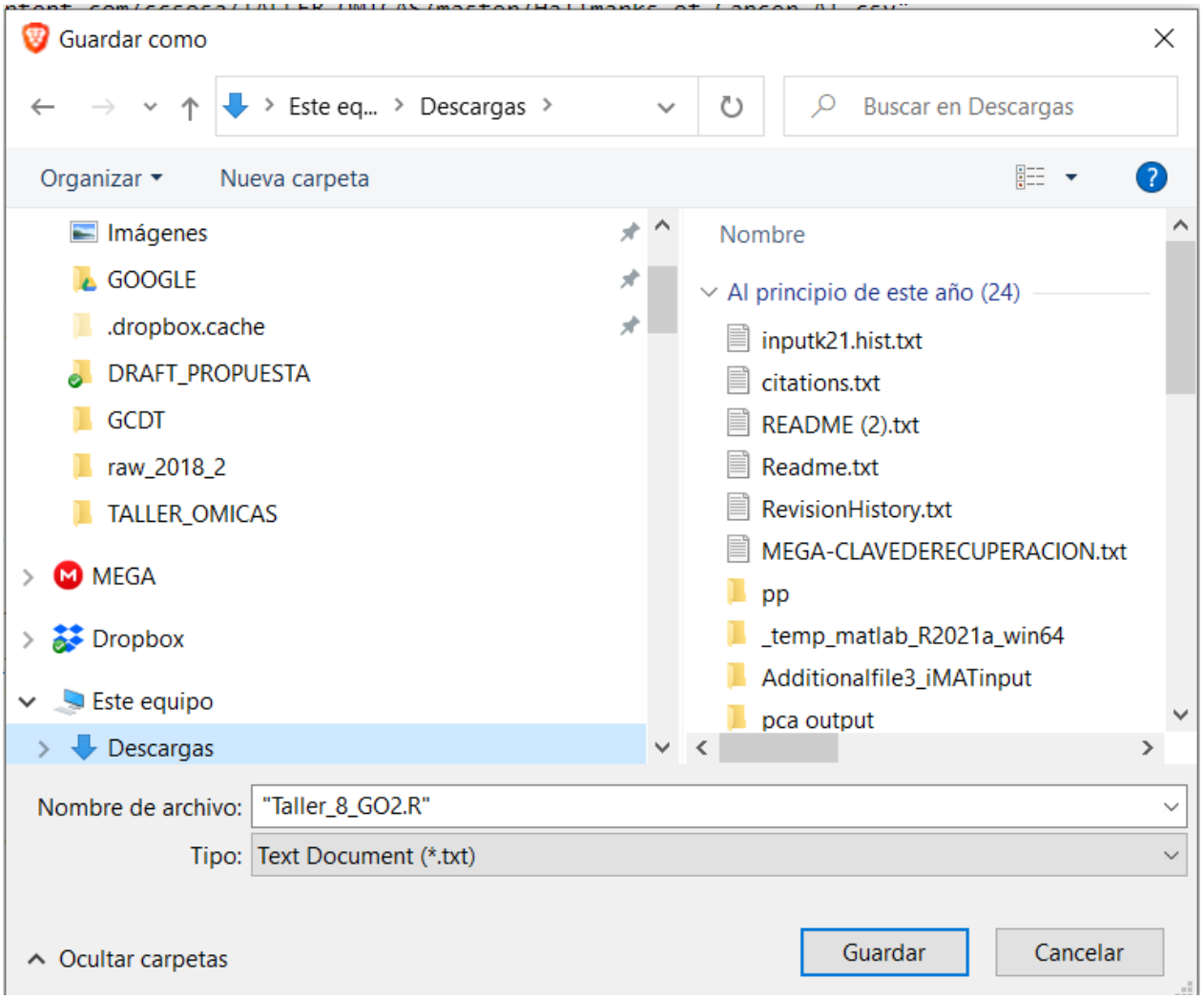
```
if (!requireNamespace("BiocManager", quietly = TRUE))

install.packages("BiocManager")

BiocManager::install("topGO")

BiocManager::install("clusterProfiler")

BiocManager::install("biomaRt")

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

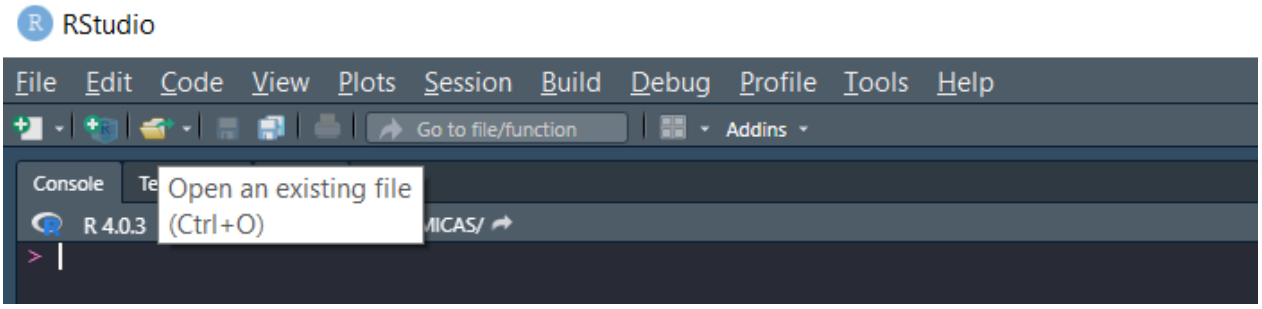
```
BiocManager::install("enrichplot")

install.packages("gprofiler2")

install.packages("ggnewscale")

install.packages("devtools")

library(devtools)

remotes::install_github("ccsosa/GOCompare")
```

- Para este ejercicio, el script con las instrucciones se puede descargar del enlace: https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/Taller_8_GO2.R y puede descargarse localmente dando clic derecho y luego usando la opción Guardar como. Posteriormente guarde el archivo como "Taller_8_GO_2.R" entre comillas.

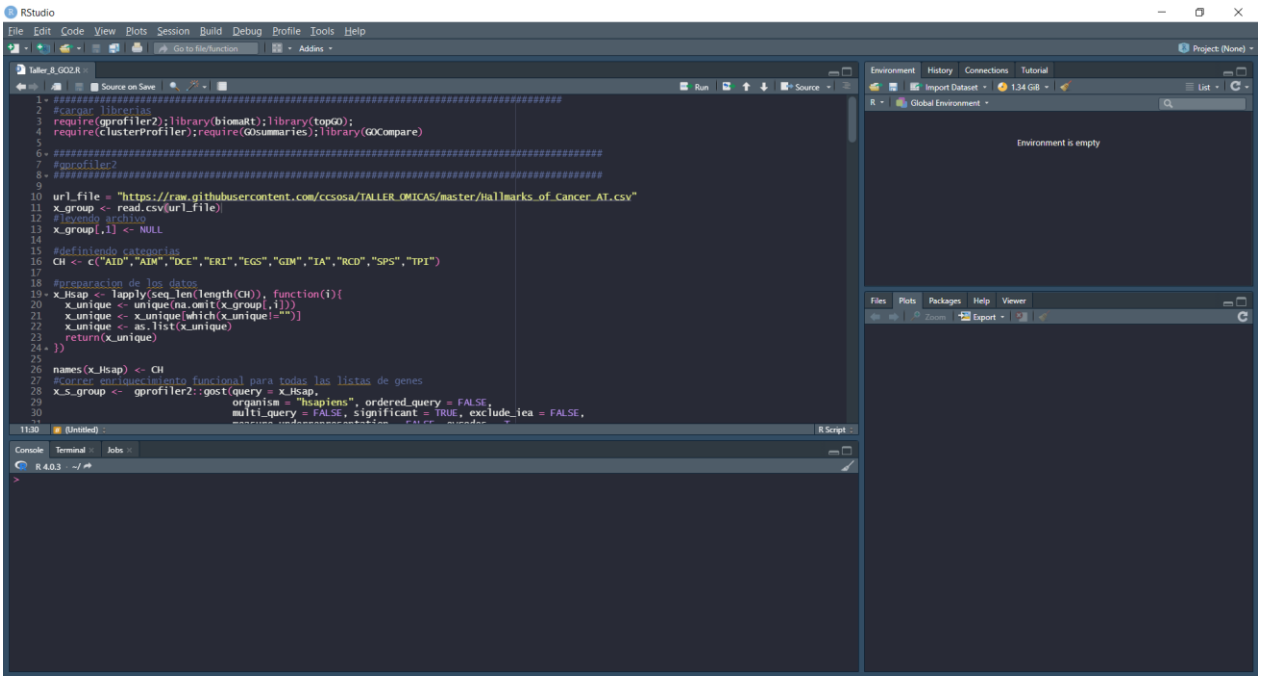
```
##############################################################################################
#cargar librerias
require(gprofiler2);library(biomaRt);library(topGO);
require(clusterProfiler);require(GOsummaries)

#############################################################################################
#gprofiler2
#############################################################################################

url_file = "https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/Hallmarks_of_Cancer_AT.csv"
x_group <- read.csv(url_file)
#leyendo archivo
x_group[,1] <- NULL

#definiendo categorias
CH <- c("AID","AIM","DCE","ERI","EGS","GIM","IA","RCD","SPS","TPI")

#preparacion de los datos
x_Hsap <- lapply(seq_len(length(CH)), function(i){
  x_unique <- unique(na.omit(x_group[,i]))
  x_unique <- x_unique[which(x_unique!="")]
  x_unique <- as.list(x_unique)
  return(x_unique)
})

names(x_Hsap) <- CH
#Correr enriquecimiento funcional para todas las listas de genes
x_s_group <-  gprofiler2::gost(query = x_Hsap,
                               organism = "hsapiens", ordered_query = FA
                               multi_query = FALSE, significant = TRUE,
                               measure_underrepresentation = FALSE, evco
                               user_threshold = 0.05, correction_method
                               domain_scope = "annotated", custom_bg = N
                               numeric_ns = "", sources = "GO:BP", as_sh

res_group <- x_s_group$result
```







• 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 3 y 4, en las cuales se cargarán las librerías gprofiler2, biomaRt, topGO, clusterProfiler y GOsummaries.

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

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

• Posteriormente se cargará el archivo con la lista de genes pertenecientes a diez categorías (*cancer hallmarks)* en humano que se encuentra localizado en GitHub y se introducen nombres cortos que se usarán más adelante (líneas 10 a la 16)

```
url_file                                                              =
"https://raw.githubusercontent.com/ccsosa/TALLER_OMICAS/master/Hallmarks_of_Cancer_AT.
csv"

x_group <- read.csv(url_file)

#leyendo archivo

x_group[,1] <- NULL

#definiendo categorias

CH <- c("AID","AIM","DCE","ERI","EGS","GIM","IA","RCD","SPS","TPI")
```

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

I. gprofiler2

- Ya que se requiere calcular análisis de enriquecimiento funcional para diez listas, procederemos a organizar los datos ya que originalmente estos estan en una matriz de diez columnas como se ve a continuación:

	AvoidingImmuneDesctruction	ActivatingInvasionMotility	DeregulationgCellularEnergetics	EnablingReplicativeImmortality	EvadingGrowthSuppressors	GenomeInstabilityMutation	InducingAngiogenes
1	CYBB	TSTA3	SUCLG1	PRIM2	VPS53	CDK5RAP3	SLC12A6
2	PSMC1	DUSP26	CHMP4C	PCNA	BARD1	CNOT7	ACTA2
3	POLR3C	BZW1	VPS36	CCT4	RBL1	RNASEH2A	AQP1
4	PSMD12	GSK3B	CYP2C9	NSMCE2	STK11	PCNA	HMGB1
5	POLR3F	BZW2	ADIPOR1	TFIP11	SIN3A	POLR2D	PRKX
6	CYFIP1	DENND6A	SORD	POLD1	CDC73	NSMCE2	AAMP
7	SLC11A1	RPL23A	GSK3B	GAR1	MYBBP1A	POLR2G	SAT1
8	SERINC3	OLA1	VPS35	UPF1	NDRG1	CDK1	NCL
9	RPS19	DOCK7	YIPF1	MAPK14	MSH2	COPS7A	CTH
10	PSME4	CYFIP1	PDK4	DKC1	FHIT	CUL4A	PDCL3
11	GNL1	RPS19	TAZ	RFC3	RAD51C	BARD1	NOX1
12	PIK3R4	CDK1	MLYCD	FEN1	PLCD1	GTF2H1	ANXA3
13	POLR3A	RPL34	CHAF1B	CALR	HINT1	POLR2I	EDF1
14	POLR3B	MMP7	ACSL5	RAD50	RBM5	POLD1	CHI3L1
15	CHIA	ATP6V1D	PDK2	TERT	ASS1	CCNB1	ACVR1
16	CTSH	GTPBP4	AKR1A1	MIF	CUL3	HMGB1	SPHK1
17	PSMD1	CAMK1D	NDUFAF1	XRCC5	PPP2R1B	CUL4B	PDCD6
18	MAPK10	ENTPD1	CHMP4A	MAPKAPK5	WWOX	UPF1	SETSIP

- El paquete gprofiler requiere que los genes sean organizados por listas y que se tenga un nombre para cada lista ingresada, para esto correremos las líneas 18 a la 26 para generar el formato necesario para nuestros análisis.

```
#preparacion de los datos

x_Hsap <- lapply(seq_len(length(CH)), function(i){

 x_unique <- unique(na.omit(x_group[,i]))

 x_unique <- x_unique[which(x_unique!="")]

 x_unique <- as.list(x_unique)

 return(x_unique)

})

names(x_Hsap) <- CH
```

- Ahora se correrá el análisis de enriquecimiento funcional, para este fin, se usará 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_group y la tabla de resultados estará disponible en el objeto res_group (líneas 28 a la 36).

```
x_s_group <- gprofiler2::gost(query = x_Hsap,
                 organism = "hsapiens", ordered_query = FALSE,
                 multi_query = FALSE, significant = TRUE, exclude_iea = FALSE,
                 measure_underrepresentation = FALSE, evcodes = T,
                 user_threshold = 0.05, correction_method = "fdr",
                 domain_scope = "annotated", custom_bg = NULL,
                 numeric_ns = "", sources = "GO:BP", as_short_link = FALSE)


res_group <- x_s_group$result
```

- Ahora obtendremos el número de términos GO enriquecidos por cada lista de genes. Al correr la línea 39 se obtendrá que la lista de genes asociados a la resistencia a muerte celular tiene más términos enriquecidos.

```
#Obtener numero de GO enriquecidos por lista de genes

tapply(res_group$query,res_group$query,length)
```

```
> tapply(res_group$query,res_group$query,length)
 AID  AIM  DCE  EGS  ERI  GIM   IA  RCD  SPS  TPI
1102 1605 1067  351  717  874  822 1784 1492  803
```

- Si quiere saber el top tres de términos GO enriquecidos y una tabla de frecuencias con esta información, corra las líneas 41 a 52.

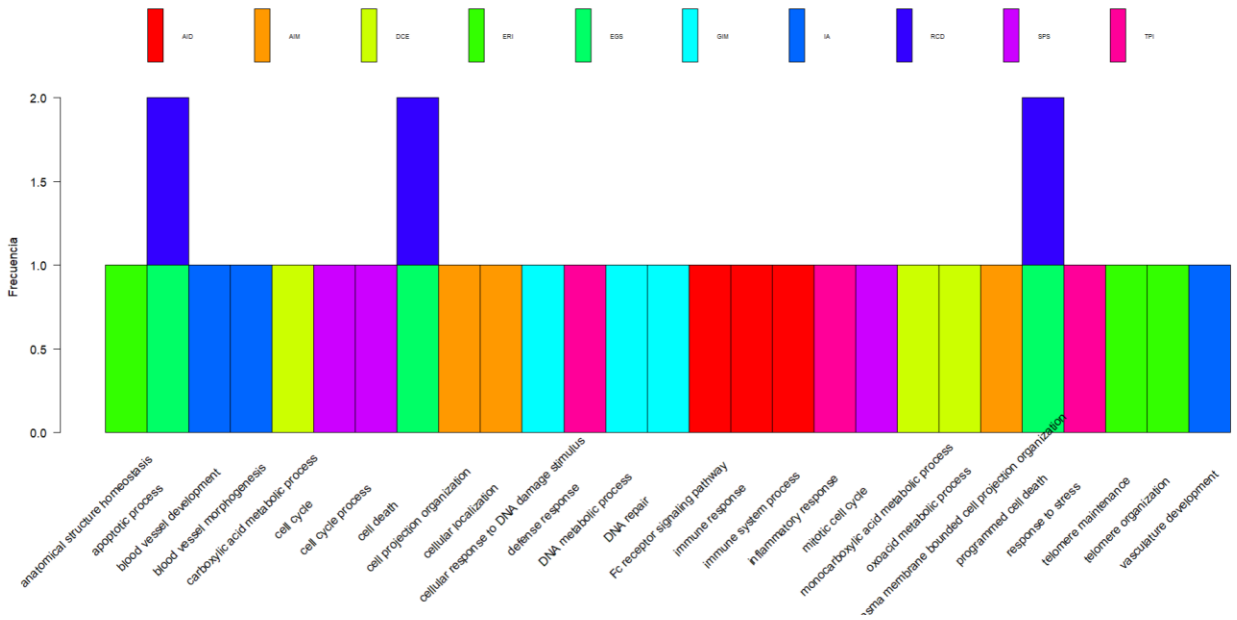
```
#Obtener top tres GO por categoria

x_res <- list()

for(i in 1:length(CH)){

 x_res[[i]] <- res_group[which(res_group$query==CH[[i]]),]

 x_res[[i]] <- x_res[[i]][c(1:3),]
```

```
}

x_res <- do.call(rbind, x_res)




#obtener la tabla de frecuencias para graficar

x_res$query <- factor(x_res$query,levels = CH)

other_table <- table( x_res$query,x_res$term_name)
```

- Ahora se generará un gráfico de barras con este top tres de términos, corriendo las líneas 54 a la 72. En este gráfico observará que hay varios términos GO compartidos entre *cancer hallmarks.*

```


par(mar = c(13, 4, 11, 1) + 0.2) #add room for the rotated labels

#par(mar=c(1,1,1,1))



bar <- barplot(other_table,

   main = "",

   xlab = "", ylab = "Frecuencia",

   col = rainbow(10),

   xaxt="n",

   axes=T,

   # legend.text = rownames(other_table),

   beside = F,

   las=2,horiz = F,

   space=0,cex.names = 0.8)


labs <- paste(colnames(other_table))
```

```
text(cex=1, x=bar-1, y=-.52,labs, xpd=TRUE, srt=45)

#axis(2, at = 0:5, labels = 0:5)

legend("top", rownames(other_table), fill = rainbow(10), bty = "n",horiz = T,inset = c(0,-0.5),xpd =
T,

   cex = 0.5)
```



II. clusterprofiler

- En esta parte del taller se usará el paquete clusterprofiler, a diferencia de gprofiler, los análisis requieren que los ids sean de ENTREZ, es decir que sean de NCBI. Para esto las líneas 79 a 87 permite obtener los ids usando la función *bitr*.

```
#Organizar la lista de genes para usarse en clusterProfiler

x_Hsap_2 <- list()

for(i in 1:length(x_Hsap)){

 x_Hsap_2[[i]] <- clusterProfiler::bitr(as.character(unlist(x_Hsap[[i]])),

                fromType = "SYMBOL",

                toType = c("ENTREZID"),

                OrgDb = "org.Hs.eg.db")[,2]

}

names(x_Hsap_2) <-CH
```

- Ahora que nuestros genes están con el identificador ENTREZ, se usará la función *compareCluster* para realizar el enriquecimiento funcional junto con la función *enrichGO*. Los resultados se guardarán en el objeto clust_results y el número de términos GO enriquecidos se obtendrá con la función *tapply* (líneas 90 a 93).

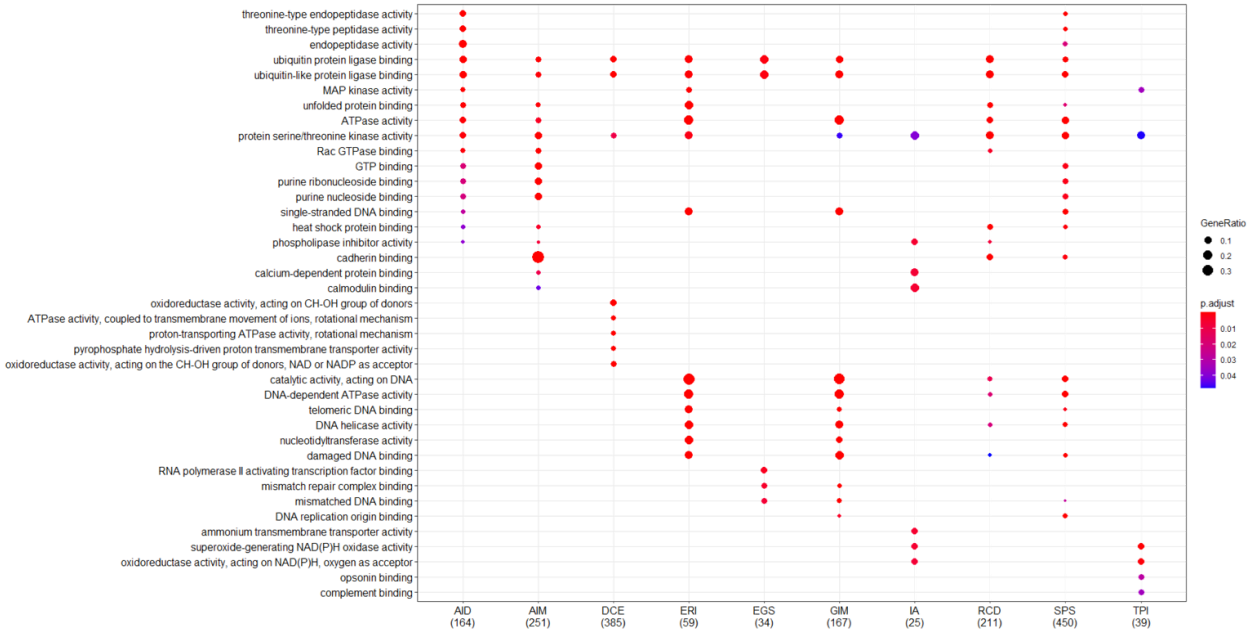
```
x_compare <- clusterProfiler::compareCluster(geneClusters=x_Hsap_2,enrichGO, OrgDb = org.Hs.eg.db)



clust_results <- x_compare@compareClusterResult

tapply(clust_results$Cluster,clust_results$Cluster,length)
```

- Para este caso, la lista de genes asociada a la desregulación de energética celular (DCE), posee un mayor número de términos GO enriquecidos.

```
> tapply(clust_results$Cluster,clust_results$Cluster,length)
AID AIM DCE ERI EGS GIM  IA RCD SPS TPI
 66  81 141  64   9 108  15  75 112  11
```

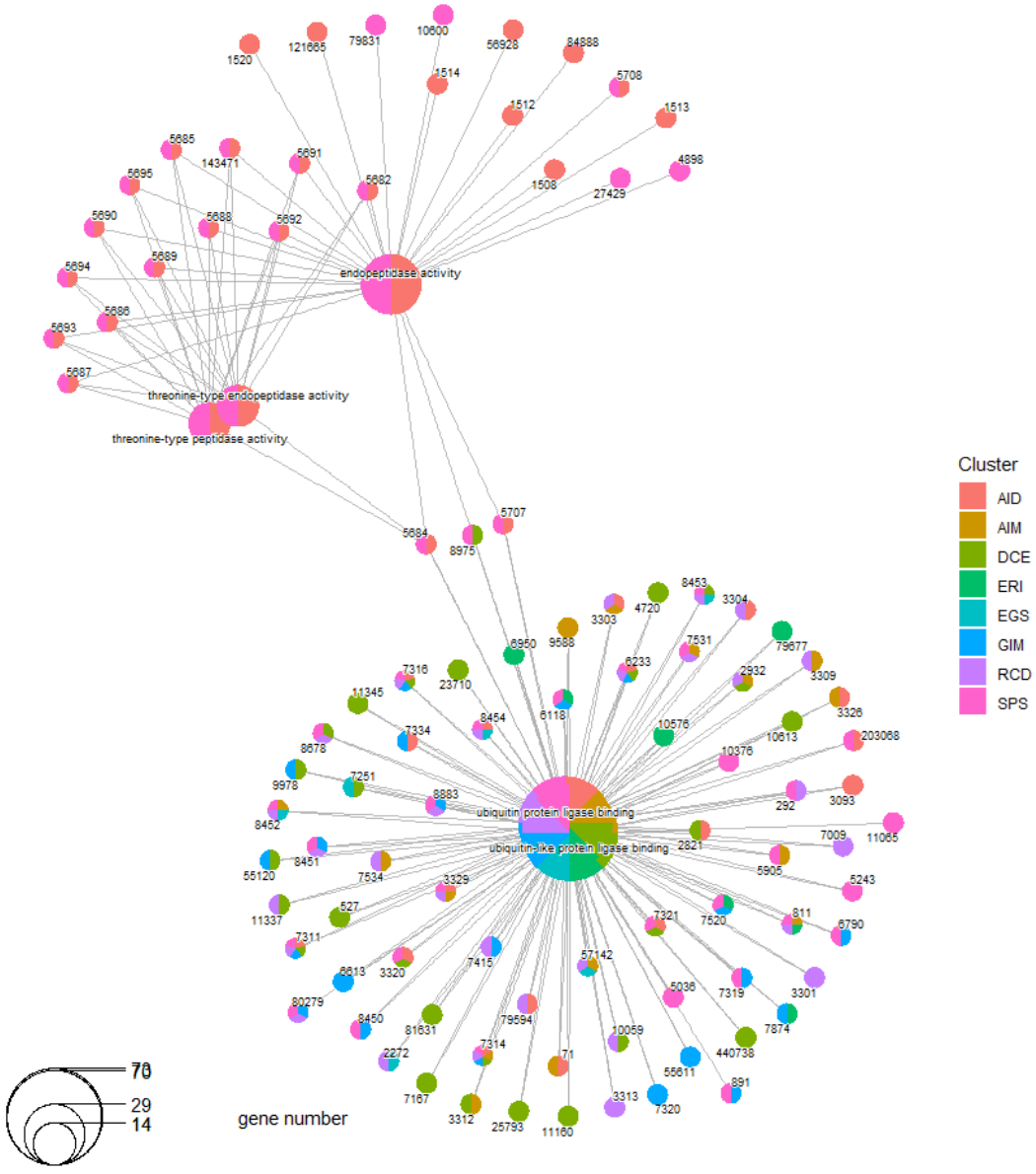
- Ahora usaremos la función *dotplot* para visualizar el resultado del análisis de enriquecimiento. En este gráfico, los puntos representan términos GO enriquecidos, el tamaño de los puntos representa la proporción de genes de cada lista y los colores representan los valores p ajustados para cada una de las diez listas de genes (línea 97).

```
#Graficar en un dotplot

enrichplot::dotplot(x_compare)
```



- Como alternativa se puede usar la función *cnetplot* (linea 100), para graficar las interacciones entre genes y términos GO enriquecidos. En este grafo se pueden visualizar el número de genes como tamaño de los nodos; los genes asociados a los términos GO, los cuales son representados como nodos centrales y, finalmente cada lista de genes es representada por un color diferente (línea 100).

```
#Graficar una red

enrichplot::cnetplot(x_compare)
```


endopeptidase activity
threonine-type endopeptidase activity
threonine-type peptidase activity
ubiquitin protein ligase binding
ubiquitin-like protein ligase binding
Cluster
AID
AIM
DCE
ERI
EGS
GIM
RCD
SPS
gene number
70
29
14

III. Adaptando resultados de otros paquetes o softwares a clusterprofiler

- Para este ejercicio usaremos los datos que obtuvimos en el paquete gprofiler, ahora correremos las líneas 106 a 116 para generar una tabla con las columnas necesarias para visualizarse en clusterprofiler.

```
#Modificar archivo de resultados para correr enrichplot

gp_mod =      res_group[,c("query",

              "source",

              "term_id",

              "term_name",

              "p_value",

              "query_size",

              "intersection_size",

              "term_size",

              "effective_domain_size",

              "intersection")]
```

- A pesar de que los resultados de gprofiler son consistentes necesitamos calcular la frecuencia de genes y la frecuencia de genes con relación a la distribución *background.* También, cambiaremos los nombres de las columnas y los nombres de listas serán convertidas en factores para usarse en los gráficos (líneas 118 a 124).

```
gp_mod$GeneRatio = paste0(gp_mod$intersection_size, "/", gp_mod$query_size)

gp_mod$BgRatio = paste0(gp_mod$term_size, "/", gp_mod$effective_domain_size)

names(gp_mod) = c("Cluster", "Category", "ID", "Description", "p.adjust",

       "query_size", "Count", "term_size", "effective_domain_size",
```

```
      "geneID", "GeneRatio", "BgRatio")

gp_mod$geneID = gsub(",", "/", gp_mod$geneID)

gp_mod$Cluster <- factor(gp_mod$Cluster)
```

- Ahora se usará la función *new* para generar un objeto de clase *compareClusterResult* y otro de clase *enrichResult* (líneas 126 a 128).

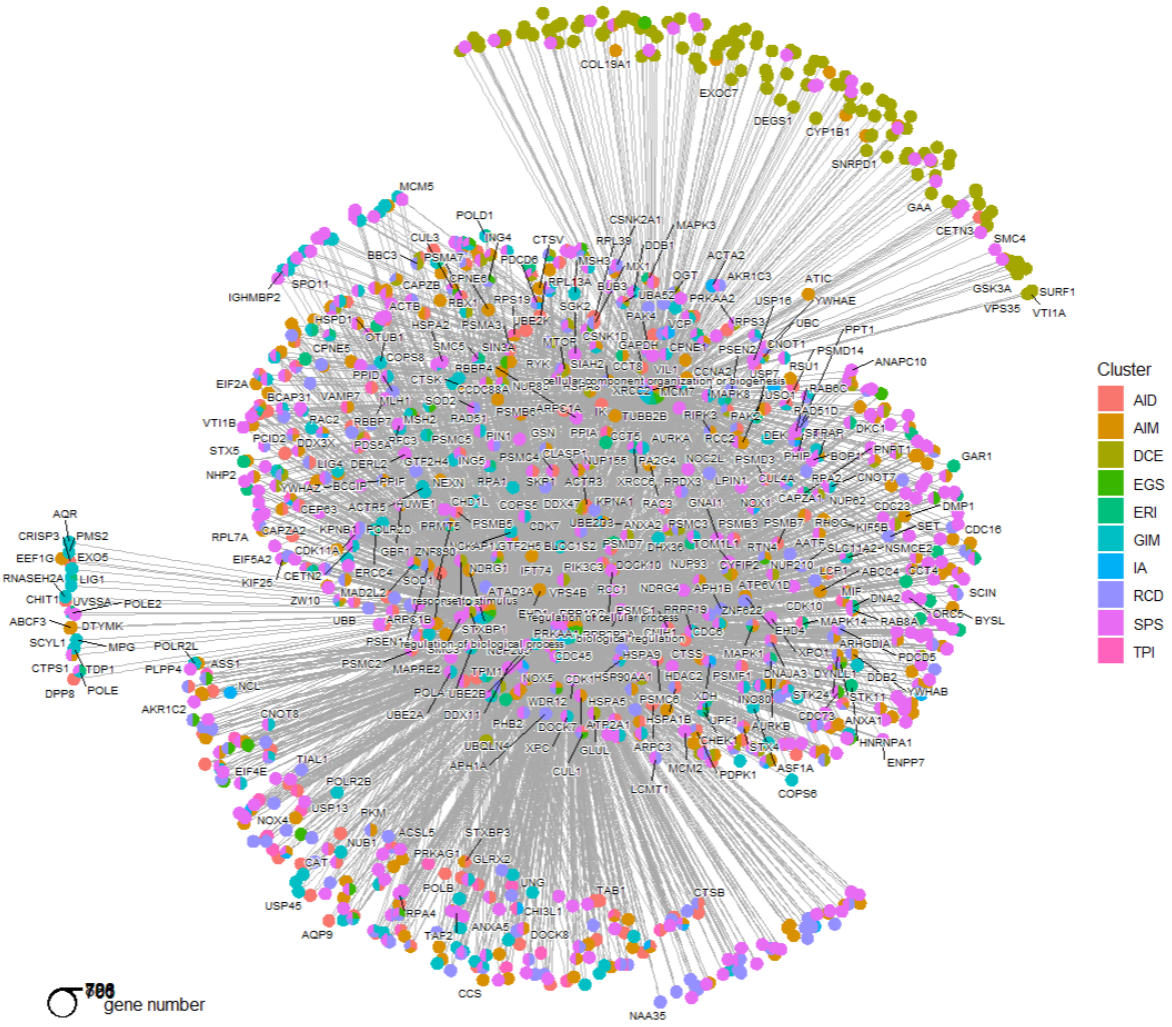
```
 # Convertir de gprofiler a clusterprofiler

gp_mod_cluster = new("compareClusterResult", compareClusterResult = gp_mod)

gp_mod_enrich = new("enrichResult", result = gp_mod)
```

- Ahora graficaremos el *dotplot* y el grafo de conectividad como lo hicimos anteriormente (líneas 131 a 135).

```
#dotplot

enrichplot::dotplot(gp_mod_cluster)

#red de interacciones

p2 <- enrichplot::cnetplot(gp_mod_cluster)

p2
```

- Los gráficos se verán como se muestra a continuación, como podrá observar, los resultados entre aproximaciones pueden variar.


response to stimulus
biological regulation
regulation of biological process
cellular component organization or biogenesis
regulation of cellular process
macromolecule metabolic process
cellular macromolecule metabolic process
programmed cell death
multicellular organism development
multicellular organismal process
organic substance metabolic process
developmental process
anatomical structure development
nitrogen compound metabolic process
metabolic process
primary metabolic process
cellular metabolic process
cellular process
AID (163)
AIM (248)
DCE (385)
EGS (34)
ERI (59)
GIM (166)
IA (24)
RCD (210)
SPS (449)
TPI (38)
GeneRatio
0.2
0.4
0.6
0.8
1.0
p.adjust
0.01
0.02
0.03
0.04


Cluster
AID
AIM
DCE
EGS
ERI
GIM
IA
RCD
SPS
TPI
gene number

IV. GOsumaries

- 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, pero los adaptaremos a un formato que lea el paquete GOSummaries (líneas 142 a 147).

```
#Ajustar el formato a GOSummaries

x_Hsap3 <- lapply(seq_len(length(CH)), function(i){

 x_unique <- as.character(x_Hsap[[i]])

 return(x_unique)

})



names(x_Hsap3) <- CH
```

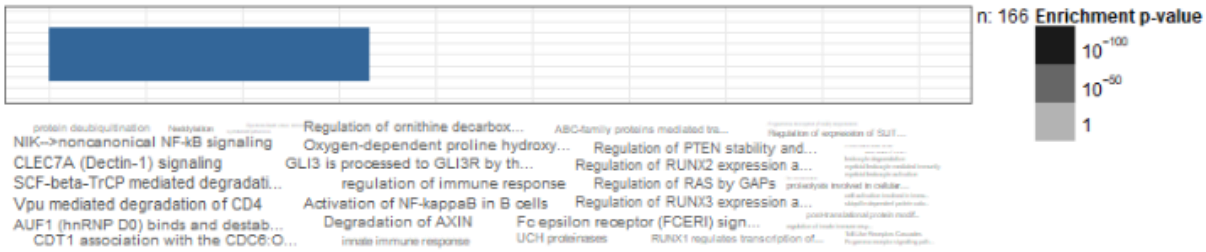
- Ahora graficaremos los resultados y se harán en dos grupos de cinco categorías (líneas 149 a 152). Los gráficos están disponibles en las siguientes dos páginas.

```
#Correr el analisis y graficar

gs1 = gosummaries(x_Hsap3)

plot(gs1[1:5])

plot(gs1[6:10])
```

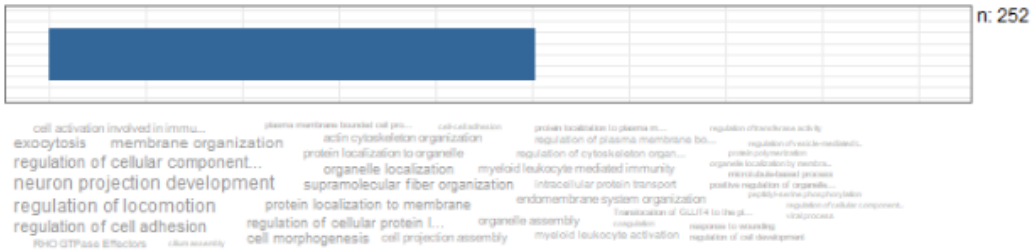
NOTA: Si desea ver los resultados, use la siguiente línea de código, ya que son diez listas de genes reemplace el número uno por el de la lista que requiera.

```
gs1[[1]]$WCD
```

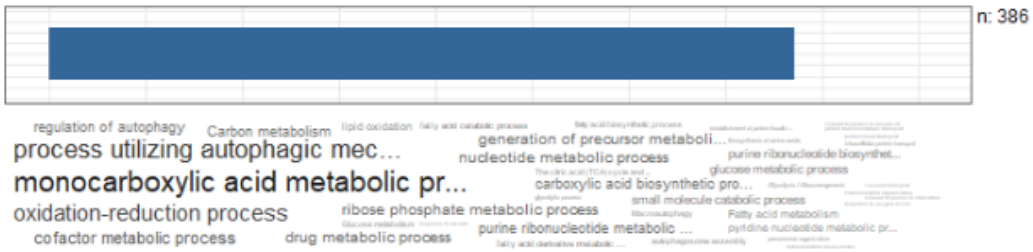
AID



AIM

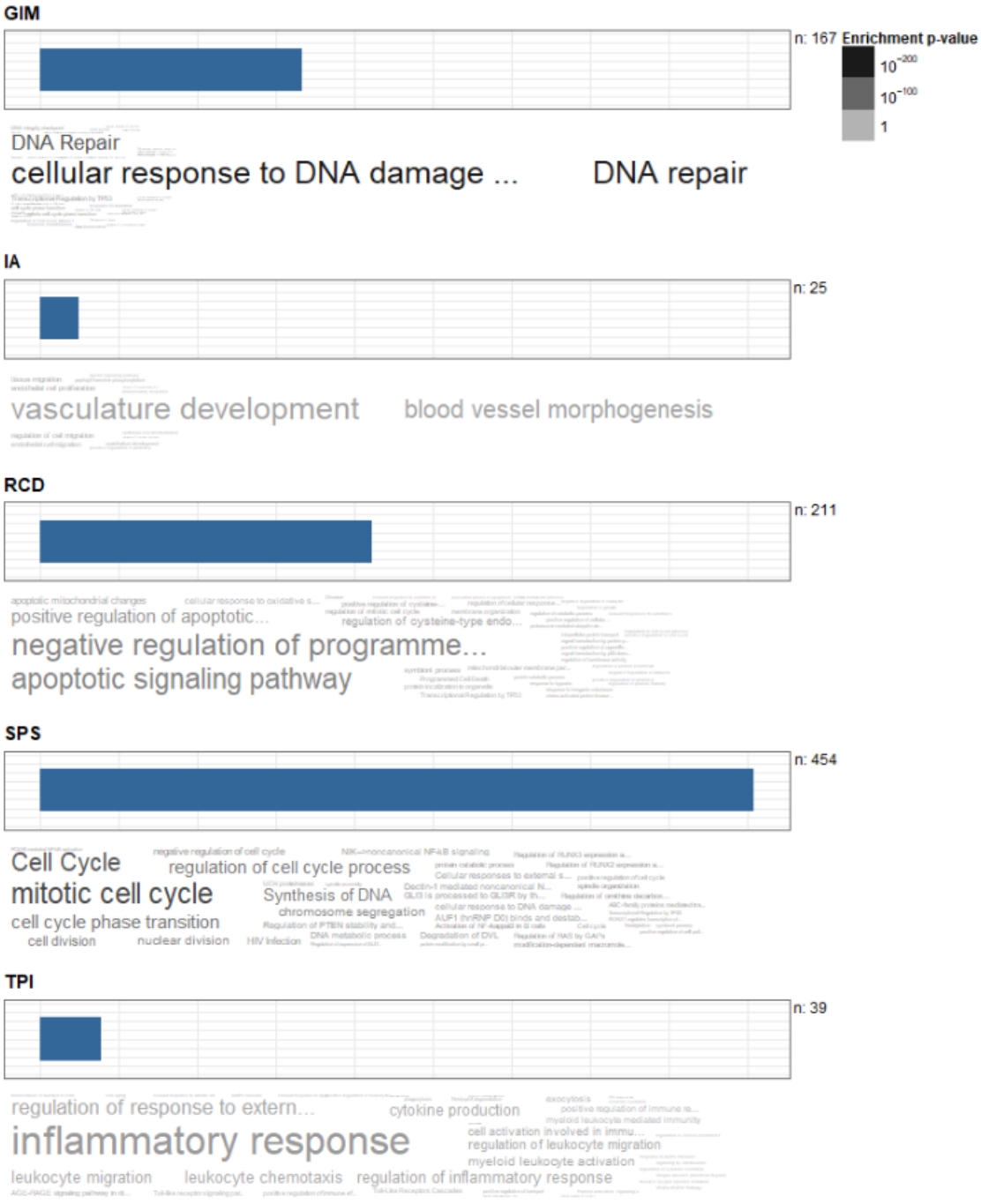


DCE



ERI

EGS


GIM
n: 167
Enrichment p-value
10^-200
10^-100
1
DNA Repair
cellular response to DNA damage ...
DNA repair
IA
n: 25
vasculature development
blood vessel morphogenesis
RCD
n: 211
positive regulation of apoptotic...
negative regulation of programme...
apoptotic signaling pathway
SPS
n: 454
Cell Cycle
regulation of cell cycle process
mitotic cell cycle
Synthesis of DNA
chromosome segregation
cell cycle phase transition
cell division
nuclear division
TPI
n: 39
regulation of response to extern...
cytokine production
inflammatory response
cell activation involved in immu...
regulation of leukocyte migration
myeloid leukocyte activation
leukocyte migration
leukocyte chemotaxis
regulation of inflammatory response

3. Comparación de listas de genes para una especie y/o entre dos especies usando GOCOmpare

Para esta parte final del taller usaremos el paquete GOCompare, el cual tiene como finalidad permitir la comparación entre listas de genes entre una especie o dos especies a través de la implementación de grafos no dirigidos con pesos para informar acerca de los procesos biológicos más relevantes, proveer un marco de análisis usando distancias de Jaccard y por último usar pruebas de hipótesis para las comparaciones. Actualmente el paquete cuenta con cinco funciones:

Función	**Foco**	**Descripción**
mostFrequentGOs	Una especie	procesos biológicos más frecuentes por categoría
graphGOspecies	Una especie	Conversión una lista procesos biológicos y categorías a grafos
compareGOspecies	Dos especies	Comparación de dos listas de procesos biológicos y categorías usando PCoA y distancias de Jaccard
evaluateGO_species	Dos especies	Evaluación de la importancia de un proceso biológico para n categorías usando pruebas Chi cuadrado
graph_two_GOspecies	Dos especies	Conversión una lista procesos biológicos y categorías en dos especies a grafos

- Para este ejemplo usaremos las funciones *compareGOspecies*, *graphGOspecies* y graph_two_GOspecies usando dos datasets de ejemplos: data(H_sapiens_compress) y data(A_thaliana_compress) los cuales son términos GO enriquecidos para genes asociados a cuatro *categorías* de características asociadas a cáncer (*cancer hallmarks)* en humano y sus respectivos genes ortólogos en *Arabidopsis thaliana.*

- Para iniciar este ejercicio cargaremos los datos de ejemplo usando las líneas 158 a 160.

```
#Cargar datos de ejemplo (cuatro cancer hallmarks)
```

```
data(H_sapiens_compress)

data(A_thaliana_compress)
```

- Los datos estan guardados como matrices con la siguiente estructura:

	Enrichment_FDR	Genes_in_list	Total_genes	Functional_Category	Genes	feature
1	0.000000e+00	778	4507	Response to stress	ACTB FGA ACTG1 TAC1 CX3CL1 CCL26 NOS2 PGLYRP1 HFE ...	AID
2	0.000000e+00	712	2062	Defense response	TAC1 CX3CL1 CCL26 NOS2 PGLYRP1 NR1H4 NR1H3 CD44 S...	AID
3	0.000000e+00	972	2602	Immune response	CX3CL1 CCL26 CD79B PGLYRP1 BTK FYN BTN3A1 SBNO2 PV...	AID
4	0.000000e+00	426	968	Regulation of defense response	NR1H4 NR1H3 SBNO2 PVR MAVS SAMHD1 TLR8 C1QBP U...	AID
5	0.000000e+00	504	1392	Immune effector process	PGLYRP1 SBNO2 PVR MAVS OAS1 SAMHD1 TLR8 C1QBP IL2...	AID
6	0.000000e+00	423	763	Activation of immune response	CD79B BTK FYN BTN3A1 TXK MAVS TLR8 CSK CD276 CD79A...	AID
7	0.000000e+00	976	3539	Immune system process	CD38 CX3CL1 CCL26 CD79B PGLYRP1 CD4 BTK FYN BTN3A1...	AID
8	0.000000e+00	590	1242	Innate immune response	CX3CL1 CCL26 PVR MAVS SAMHD1 TLR8 CCL22 MEFV RIPK...	AID
9	0.000000e+00	718	1909	Regulation of immune system p...	CD38 CD79B PGLYRP1 BTK FYN BTN3A1 PVR TXK PAG1 MAV...	AID
10	0.000000e+00	587	1301	Positive regulation of immune s...	CD38 CD79B BTK FYN BTN3A1 PVR TXK MAVS IL12RB1 MA...	AID
11	0.000000e+00	807	4820	Regulation of response to stim...	TSPAN6 FKBP1A MAVS MUL1 TRIM38 BST2 SECTM1 LGALS9...	AID
12	0.000000e+00	656	2621	Positive regulation of response ...	TSPAN6 FKBP1A MAVS MUL1 TRIM38 BST2 SECTM1 LGALS9...	AID
13	0.000000e+00	658	1325	Regulation of immune response	CD79B PGLYRP1 BTK FYN BTN3A1 PVR TXK MAVS SAMHD1 ...	AID
14	0.000000e+00	519	991	Positive regulation of immune r...	CD79B BTK FYN BTN3A1 PVR TXK MAVS TLR8 CSK CD276 C...	AID
15	0.000000e+00	372	662	Immune response-activating si...	CD79B BTK FYN BTN3A1 TXK TLR8 CSK CD276 CD79A DUSP...	AID

- Dado que compararemos listas de genes y términos GO enriquecidos, debemos establecer las categorías en la columna *feature* y definir en el código el nombre de columna que tiene los términos GO, que en este caso es “Functional_Category”. Para este fin usaremos la línea 163.

```
#Definir la columna que tiene la info de los GO enriquecidos

GOterm_field <- "Functional_Category"
```

- Ya que tenemos dos especies y dos matrices que tienen la misma estructura, debemos decirle al programa el nombre de las especies (líneas 165 a 167).

```
#Nombrar las especies

species1 <- "H. sapiens"

species2 <- "A. thaliana"
```

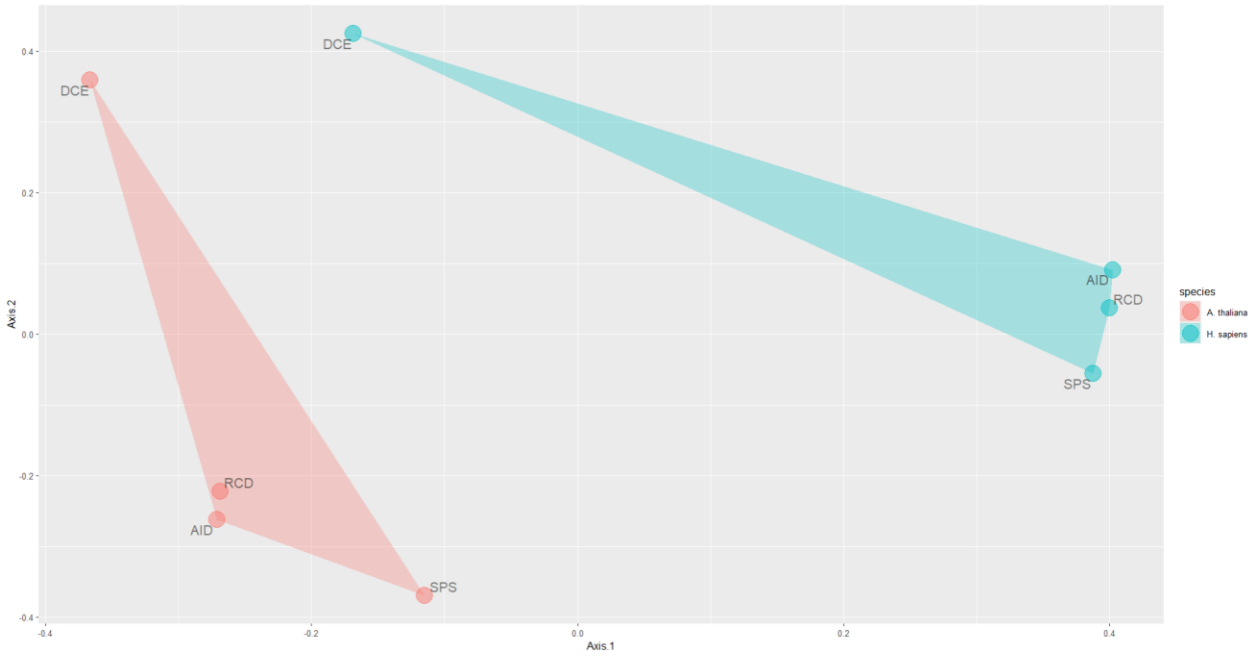
- Ahora procederemos a comparar las dos matrices de términos GO y las dos especies, es decir cuatro categorías y dos especies (ocho grupos para comparar). La función *compareGOspecies* permitirá está comparación realizando: (i) distancias de Jaccard, las cuales se usarán para un (ii) análisis de coordenadas principales (*PCoA)*, y (iii) permitirá extraer los términos GO compartidos y únicos por cada especie (líneas 169 a 176).

```
x <- compareGOspecies(df1=H_sapiens_compress,
```

```
        df2=A_thaliana_compress,

        GOterm_field=GOterm_field,

        species1=species1,

        species2=species2)
```

- Para visualizar el PcoA debe usar el código de la línea 176. Para este *PCoA,* cada conjunto de categoría y especie es visto como un individuo. Así, el *PCoA* permite ver que tan cercanos son dos especies con relación a la presencia o ausencia de un término GO.

```
#graficar el PCoA

x$graphics
```



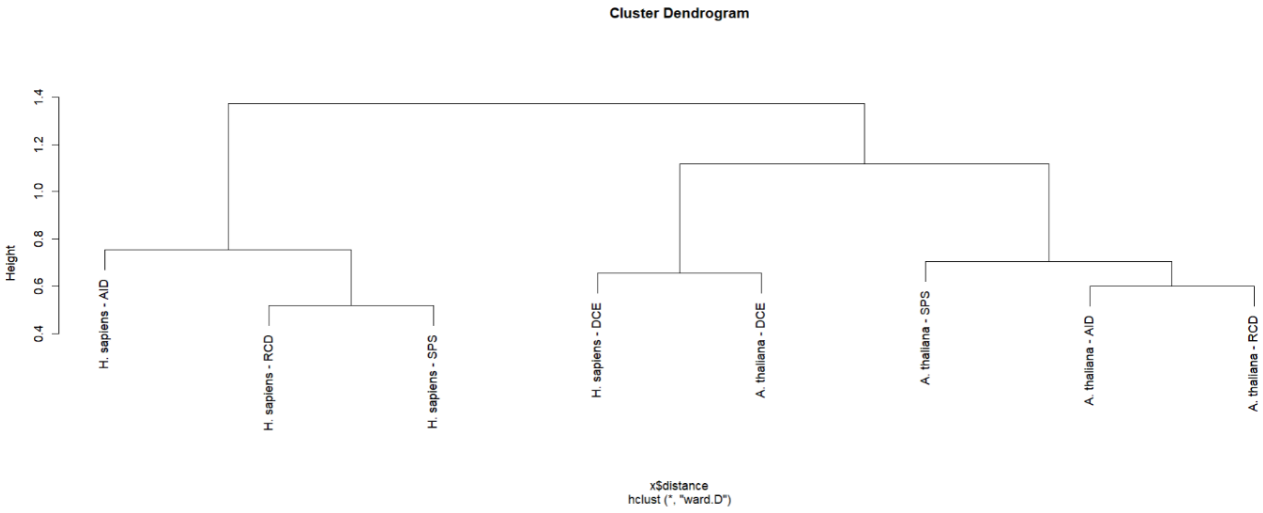
NOTA: Si desea ver los términos GO compartidos o únicos, use los comandos x$shared_GO_list y x$unique_GO_list

feature	GO	species
AID	Regulation of defense response	H. sapiens
AID	Immune effector process	H. sapiens
AID	Activation of immune response	H. sapiens
AID	Regulation of immune system process	H. sapiens
AID	Positive regulation of immune system process	H. sapiens
AID	Regulation of response to stimulus	H. sapiens
AID	Positive regulation of response to stimulus	H. sapiens
AID	Regulation of immune response	H. sapiens
AID	Positive regulation of immune response	H. sapiens
AID	Immune response-activating signal transduction	H. sapiens
AID	Immune response-regulating signaling pathway	H. sapiens

feature	GO	species
AID	Response to stress	Shared
AID	Defense response	Shared
AID	Immune response	Shared
AID	Immune system process	Shared
AID	Innate immune response	Shared
AID	Response to external stimulus	Shared
AID	Response to biotic stimulus	Shared
AID	Response to other organism	Shared
AID	Response to external biotic stimulus	Shared
AID	Response to organic substance	Shared
AID	Cellular response to organic substance	Shared

- Si desea ver que tan parecidos son la combinación de categorías y especies, use la línea 179. Para este caso verá que la desregulación de energética celular es muy similar entre *H. sapiens* y *A. thaliana.*

```
#cluster con las distancias

plot(hclust(x$distance,"ward.D"))
```



- Si desea obtener la importancia de cada término GO use la función *graphGOspecies*. Esta función creará un grafo no dirigido con pesos para este fin. La función posee dos opciones para crear grafos, (i) “Categories”, la cual usará las categorías, en este caso (*cancer hallmarks)* como nodos y las aristas representan la interacción entre términos. (ii) “GO”, usará los procesos biológicos como nodos y la pertenencia a las categorías como aristas. Esta función está paralelizada y generará grafos muy densos, por lo tanto, debe ser paciente y usar un número de procesadores (*numCores*) que sea adecuado, a su vez, permite guardar el resultado como un archivo *graphML* que puede verse en *Cytoscape* (líneas 182 a 187).

```
#Extraer pesos para GO enriquecidos en una sola especie

x_graph <- graphGOspecies(df=H_sapiens_compress,

        GOterm_field=GOterm_field,

        option = "GO",

        numCores=2,

        saveGraph=FALSE,

        outdir = NULL)
```

- Después de correr la función obtendrá una lista con 1,257 nodos y 433,285 aristas. También notará que los términos GO asociados a estrés y a respuesta a estímulos tienen mayores pesos.

	GO	GO_WEIGHT
1	Response to stress	1.587908
11	Regulation of response to stimulus	1.587908
19	Regulation of response to stress	1.587908
26	Response to external stimulus	1.587908
34	Response to organic substance	1.587908
36	Cellular response to organic substance	1.587908
37	Cellular response to chemical stimulus	1.587908
41	Multi-organism process	1.587908
47	Positive regulation of cell communication	1.587908
48	Positive regulation of signaling	1.587908
49	Positive regulation of signal transduction	1.587908
57	Intracellular signal transduction	1.587908
59	Regulation of cell communication	1.587908
60	Regulation of signaling	1.587908
79	Regulation of protein metabolic process	1.587908
81	Positive regulation of intracellular signal transduction	1.587908
84	Positive regulation of protein metabolic process	1.587908
86	Interspecies interaction between organisms	1.587908

- Ahora para finalizar el taller usaremos la función *graph_two_Gospecies,* la cual comparará listas de genes entre dos especies. Esta función posee las mismas opciones que la función *graphGOspecies*, sin embargo, generará un grafo compuesto de tres subgrafos (uno por especie y uno con los términos compartidos entre especies) (líneas 190 a 198).

```
#Extraer pesos para GO enriquecidos entre dos especies y categorias

x_graph_two <- graph_two_GOspecies(x=x,

            species1=species1,

            species2=species2,

            GOterm_field=GOterm_field,

            numCores=2,

            saveGraph = FALSE,

            option= "GO",

            outdir = NULL)
```

- Después de correr la función, obtendrá una lista con 1,948 nodos y 456,674 aristas. También notará que los términos GO asociados a procesos catabólicos y transporte tienen mayores pesos.

	GO	GO_WEIGHT
123	Organic substance catabolic process	1.818718
217	Establishment of localization in cell	1.818718
121	Organic substance transport	1.795058
138	Nitrogen compound transport	1.795058
222	Intracellular transport	1.795058
225	Intracellular protein transport	1.795058
279	Response to temperature stimulus	1.795058
283	Response to heat	1.795058
285	Organonitrogen compound catabolic process	1.795058
297	Response to acid chemical	1.795058
78	Positive regulation of phosphorylation	1.779604
82	Cellular catabolic process	1.670305
84	Catabolic process	1.670305
190	Macromolecule localization	1.670305
193	Establishment of protein localization	1.670305