MEMORIA DE PRÁCTICAS

FCO JAVIER FONTENLA LABRADOR

11/2020-02/2021

- Instalación de paquetes y carga de librerías

```
install.packages("UniprotR")
install.packages("RCurl")
install.packages("bitops")
install.packages("devtools")
install.packages("dplyr", dependencies = TRUE)
install.packages("tidyverse")
install.packages("BiocManager")
install.packages("writexl")
install.packages("ALL")

BiocManager::install("UniProt.ws")
BiocManager::install("hipathia")
BiocManager::install("org.Hs.eg.db")
BiocManager::install("AnnotationDbi")
BiocManager::install("topGO")
BiocManager::install("GO.db")
BiocManager::install("ALL")

library(hipathia)
library(AnnotationDbi)
library(org.Hs.eg.db)
library(BiocManager)
library(UniProt.ws)
library(UniprotR)
library(devtools)
library(dplyr)
library(tidyverse)
library(writexl)
library(topGO)
library(GO.db)
library(ALL)
```

- Cargamos la base de datos “hsa” y obtenemos todos sus genes. Realizamos un mapeo asociando los “EntrezID” de cada gen con su identificador alojado en “Uniprot”. Obtenemos un DataFrame “uniprots” con todos los datos.

```
pathways <- load_pathways("hsa")

genes <- pathways$all.genes


uniprots <- data.frame( entrez = genes, uniprot = mapIds(x = org.Hs.eg.db, keys = genes, column = "UNIPROT", keytype = "ENTREZID"), stringsAsFactors = F)
```

- Buscaremos añadir al dataframe una columna correspondiente a un vector de NA con el fin de poder ir mirando uno a uno cada valor, relacionando aquellos identificadores que si que tienen un valor asociado del BiologicalProcess y sustituyendo el NA correspondiente por su valor. El objetivo es obtener finalmente el dataframe con toda la información útil de los biological processes que podemos sacar del paquete.

```
Obj_ProteinGOinfo <- GetProteinGOInfo(uniprots$uniprot)

BP_ProteinGOinfo <- select(Obj_ProteinGOinfo, Gene.ontology..biological.process.)


names(BP_ProteinGOinfo) #NOMBRE DE DATAFRAME

BP_ProteinGOinfo<-rename(BP_ProteinGOinfo, BiologicalProcess = Gene.ontology..biological.process.) #RENOMBRAR COLUMNA DE DATAFRAME


BPNA<-filter(BP_ProteinGOinfo, is.na(BiologicalProcess)) #VECTOR DE NULOS

BPVALOR<-filter(BP_ProteinGOinfo, !is.na(BiologicalProcess)) #VECTOR DE VALORES
```

- Creamos una columna que va a contener la información de anotaciones que está en BP_ProteinGOinfo$BiologicalProcess, y la llenamos de "NA" para que así las proteínas que no tengan anotación se queden simplemente con el "NA". Es la forma de evitar campos vacíos y diferente número de filas.

```
uniprotsfinal <- uniprots

uniprotsfinal$BiologicalProcess <- "NA"
```

- A %in% B: "%in%" lo que hace es buscar A en B y te da un vector de TRUE/FALSE, con "which" lo pasas a índice (ej. 1,5,9,7,19,30) en este caso una vez hagas el "which" el índice te va a indicar la posición de elementos en el vector uniprotsfinal$uniprot, primer argumento.

```
idx_sub <- which (uniprotsfinal$uniprot %in% rownames(BP_ProteinGOinfo))
```

- El match te da directamente el índice (ej. 1,4,8,7,12,34) con el orden en el que encuentra los identificadores uniprot$uniprot en rownames(BP_ProteinGOinfo). En este caso funciona al revés que "%in%", porque el índice corresponde con la posición en el vector rownames(BP_ProteinGOinfo), que es el segundo argumento de la función. Utilizamos "match" porque en este caso quieres acceder a los elementos de BP_ProteinGOinfo$BiologicalProcess, pero te interesa que estén (que son las anotaciones de las proteinas) en el orden del data.frame uniprotsfinal.

```
idx_replace <- match(uniprotsfinal$uniprot, rownames(BP_ProteinGOinfo))

v1= uniprotsfinal$BiologicalProcess[idx_sub]

v2= BP_ProteinGOinfo$BiologicalProcess[idx_replace]

uniprotsfinal$BiologicalProcess <- str_replace(v1, v2)
```

- Por otro, vamos a realizar exactamente el mismo proceso, pero esta vez nos interesa usar la función GetProteinFunction para obtener los valores de “FunctionCC”.

```
Obj_ProteinFunction <- GetProteinFunction(uniprots$uniprot)

FCC_ProteinFunction <- select(Obj_ProteinFunction, Function..CC.)

names(FCC_ProteinFunction) #NOMBRE DE DATAFRAME

FCC_ProteinFunction<-rename(FCC_ProteinFunction, FunctionCC = Function..CC.)
#RENOMBRAR COLUMNA DE DATAFRAME

FCCNA<-filter(FCC_ProteinFunction, is.na(FunctionCC)) #VECTOR DE NULOS

FCCVALOR<-filter(FCC_ProteinFunction, !is.na(FunctionCC)) #VECTOR DE VALORES

uniprotsfinal2 <- uniprots

uniprotsfinal$FuncionCC <- "NA"

idx_sub2 <- which (uniprotsfinal$uniprot %in% rownames(FCC_ProteinFunction))

idx_replace2 <- match(uniprotsfinal$uniprot, rownames(FCC_ProteinFunction))

v3= uniprotsfinal$FuncionCC[idx_sub2]

v4= FCC_ProteinFunction$FuncionCC[idx_replace2]

uniprotsfinal$FuncionCC <- str_replace(v3, v4)
```

- Pasamos ahora a la búsqueda de las anotaciones en el paquete “Uniprot.ws”. Para ello vamos a obtener los identificadores de Uniprot del paquete uniprot.ws y vamos a buscar la relación con los EntrezID de Hipathia.

```
#Paquete uniprot.ws


up <- UniProt.ws(taxId=9606)

uni<-up@taxIdUniprots

keytypes(up)

columns(up)


res <- UniProt.ws::select(up, keys = genes, columns = "UNIPROTKB", keytype =
"ENTREZ_GENE")

res2 <- UniProt.ws::select(up, keys = genes, columns = "KEYWORDS", keytype =
"ENTREZ_GENE")

vectorvalores<-filter(res, !is.na(res$FUNCTION))
```

- Pasamos ahora a la búsqueda de las anotaciones en GO. En primer lugar buscamos establecer la relación GO(GO.DB) – ENTREZ(HIPATHIA). Una vez realizada la relación y la creación de la tabla realizamos un filtrado con el fin de buscar solo aquella información que corresponde con los biological processes.

```
#Paquete go.db

columns(GO.db)

keytypes(GO.db)


res2 <- GO.db::select(up, keys = genes, columns = "GO.db", keytype =  "ENTREZ_GENE")


vec1<-list()

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

 if(xy[[i]]@Ontology=="BP"){

   vec1<-c(vec1, xy[[i]])

 }

}


GOID<-list()

for(i in 1:length(vec1)){

 GOID<-c(GOID,vec1[[i]]@GOID)

}


TERM<-list()

for(i in 1:length(vec1)){

 TERM<-c(TERM,vec1[[i]]@Term)

}


unlist(GOID)

unlist(TERMS)

goterms<-data.frame(GOID=unlist(GOID), TERM=unlist(TERMS))
```

- Finalmente buscamos realizar de igual forma que en “GO.db”, buscar en “topGO” la anotación de interés, en nuestro caso las relacionadas con los BP. Realizamos una nueva relación entre las anotaciones de GO y los EntrezID e invertimos el dataframe.

```
#Paquete topGO


BPterms <- ls(GOBPTerm)

head(BPterms)


xx <- annFUN.org("BP", mapping = "org.Hs.eg.db", ID = "entrez")

a<-annFUN.GO2genes("BP", feasibleGenes = genes, xx)

b<-stack(a)

b<-data.frame(values=b$ind, ind=b$values)

d<-unstack(b)
```