PEC 1

ARANTXA GARCIA REDON

2025-03-30

En primer lugar, vamos a cargar los paquetes que necesitamos para resolver la PEC y además vamos a cargar los datos en R para poder trabajar con ellos. En mi caso he decidido trabajar con el conjunto de datos human_cachexia. He decidido trabajar con este conjunto de datos porque es sencillo pero versatil en cuanto a los análisis que se puede hacer con él. Se trata de datos de los metabolitos de varios pacientes, un grupo control y un grupo caquéxico. En los metadatos de nos indica que las muestras no son pareadas, es decir, que no se trata de datos de una misma persona que en un momento era normal y despues desarrolla la enfermedad, por lo que nos da una guía de los tests estadísticos que podemos emplear y los que no. Nos dice además que todos los datos son numéricos y que no hay datos nulos.

```
# Se carga SummarizedExperiment. EN la cabecera se incluye:
# - warning=FALSE, message=FALSE: que permite que aunque se visualice el código, los
# warnings que aparecen al cargar la librería no ensucien el informe.
# - results='hide' porque la opción head también devuelve un resultado muy largo.
library(SummarizedExperiment)
# Se cargan los datos y los metadatos para trabajar con ellos
datos <-read.csv("human_cachexia.csv")
metadatos <-readLines("description.md")
# Se visualizan los datos.
# De nuevo he empleado
print(metadatos)
head(datos)
```

Este conjunto de datos tiene la siguiente estructura: - Las filas representan los diferentes pacientes del estudio. Algunos son caquéxicos y otros no lo son.

- En las columnas tenemos los diferentes datos que se recogen de cada paciente. La primera columna se trata de el identificador del paciente, la segunda su condicion de salud que puede ser normal o caquéxico y el resto de las columnas son diferentes metabolitos que se han recogido de cada uno de los pacientes.

1. Creación de un objeto SummarizedExperiment y estudio de las diferencias con la clase ExpressionSet

Para la creación del objeto SummarizedExperiment (de ahora en adelante, SE), he accedido a la documentación oficial de Bioconductor (https://www.bioconductor.org/packages/release/bioc/vignettes/SummarizedExperiment/inst/doc/SummarizedExperiment.html#subsetting).

En primer lugar, mirando el apartado de la anatomía de un objeto SE, lo que se puede observar es la necesidad de trasponer los datos daado que en un objeto SE las diferentes muestras(pacientes) están en las columnas y en las filas se deben recoger los features, que en este caso son los metabolitos. Además se admite que se tenga más de un ensayo para unas mismas samples y features y unos metadatos, que se almacenen por separado pero ligados al conjunto de datos principal.

En el código de a continuación se va a crear el objeto SE tras preprocesar los datos para poder llevar a cabo su creación.

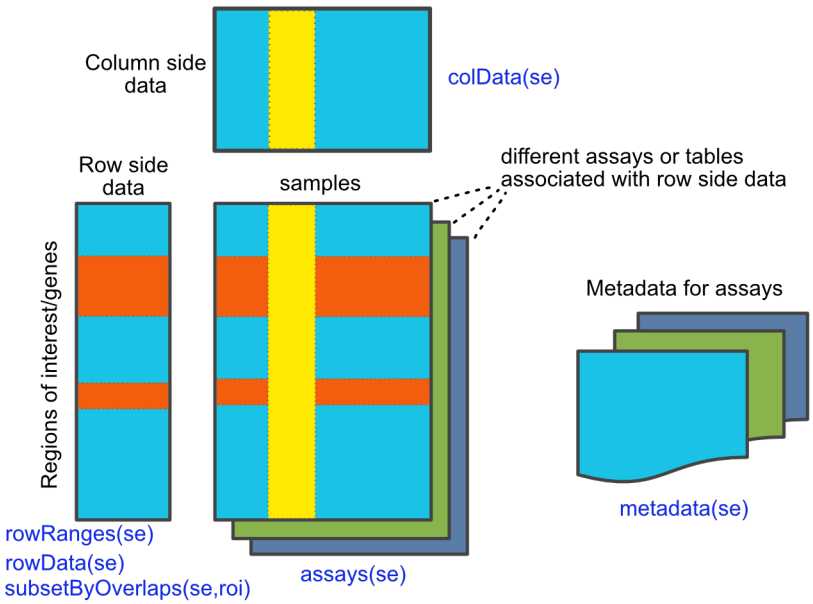


Figure 1: Estrucutra SE

```
# Creo una nueva matriz que es la traspesta de datos, en la cual elimino
# las dos primeras columnas para quedarme solo con los datos numéricos
datos_num<-t(datos[,-c(1,2)])
# El  coldatos contendrá los valores de la segunda columna de datos.
# Es decir el dato de control o enfermos guardado en un vector
coldatos<-datos[,2]
# Convertimos coldatos en un dataframe que contiene el ID del paciente y su condicion
coldatos<-data.frame(SampleName=datos[,1], Muscle.loss=datos[,2])
#Creamos el objeto SE.
se<-SummarizedExperiment(assays=list(counts=datos_num), colData=coldatos, metadata=metadatos)
```

El objeto SE contiene los siguientes componentes:

- `assays = list(counts = datos_num)`: esto contendrá los datos numéricos que hacen referencia a las cantidades de metabolitos de la matriz transpuesta datos_num.
- `colData = coldatos`: contiene los metadatos asociados a cada columna (es decir, a las muestras). En este caso, se incluye la información del dataframe coldatos, que contiene:
  - SampleName (nombre de la muestra)
  - Muscle.loss (información sobre la pérdida muscular)
- `metadata = metadatos`: Este aargumento nos permite introducir los metadatos en formato texto que hemos importado del archivo description.

En la PEC nos pide que comparemos SE con ExpressionSet. Ambas son clases, o estructuras de datos de Bioconductor que son frecuentemente utilizadas para representar y manejar datos experimentales de datos ómicos. Ambas se usan mucho en estudios de expresión génica.

Para comprender mejor la clase ExpressionSet, de nuevo se ha consultado la página oficial de bioconductor (https://www.bioconductor.org/packages/devel/bioc/vignettes/Biobase/inst/doc/ExpressionSetIntroduction.pdf). La principal diferencia entre los dos paquetes parece ser la flexibilidad en cuanto a los tipos de datos que pueden almacenarse en cada una de las estructuras de datos. Mirando diferentes repositorios en github y stackoverflow se puede ver que ExpressionSet se utiliza para estudios de expresión génica con microarrays de

forma casi exclusiva. Sin embargo, SummarizedExperiment es más flexible y permite trabajar con datos de RNAseq y otros datos ómicos.

Otra diferencia es como se estucturan los datos:

- `SummarizedExperiment` se estructura en:
  - assays: que es una matriz o dataframe con los datoss experimentales (por ejemplo los datos de expresión o de metabolómica como en el dataset que estamos usando)
  - `coldata`: un dataframe que contiene los metadatos de las muestras.
  - `rowdata`: un dataframe con los metadatos sobre las filas, como los genes
  - `metadata`: un apartado para metadatos generales del experimento, donde en este caso hemos almacenado la descripción.
- `ExpressionSet` tiene tres componentes principales:
  - `exprs`: una matriz de datos de expresión con los genes en filas y las muestras en columnas.
  - `phenodata`: un objeto que contiene metadatos sobre las muestras, como por ejemplo las condiciones del experimento.
  - `featureData`que contiene información sobre los genes del experimento.

En general, podemos decir que las diferencias en su estructura hacen que `ExpressionSet`es una estructura de datos más rígida que puede resultar muy útil en experimentos de expresión génica con microarrays, mientras que `SummarizedExperiment`tiene una estructura más flexible que nos permite almacenar más datos y es útil para otros tipos de datos ómicos, no solo de microarrays.

A continuación voy a guardar el objeto SummarizedExperiment en formato binario .Rda como se indica en las instrucciones de la PEC, para ello se usa la función save:

```
save(se, file = "cachexia_se.Rda")
```

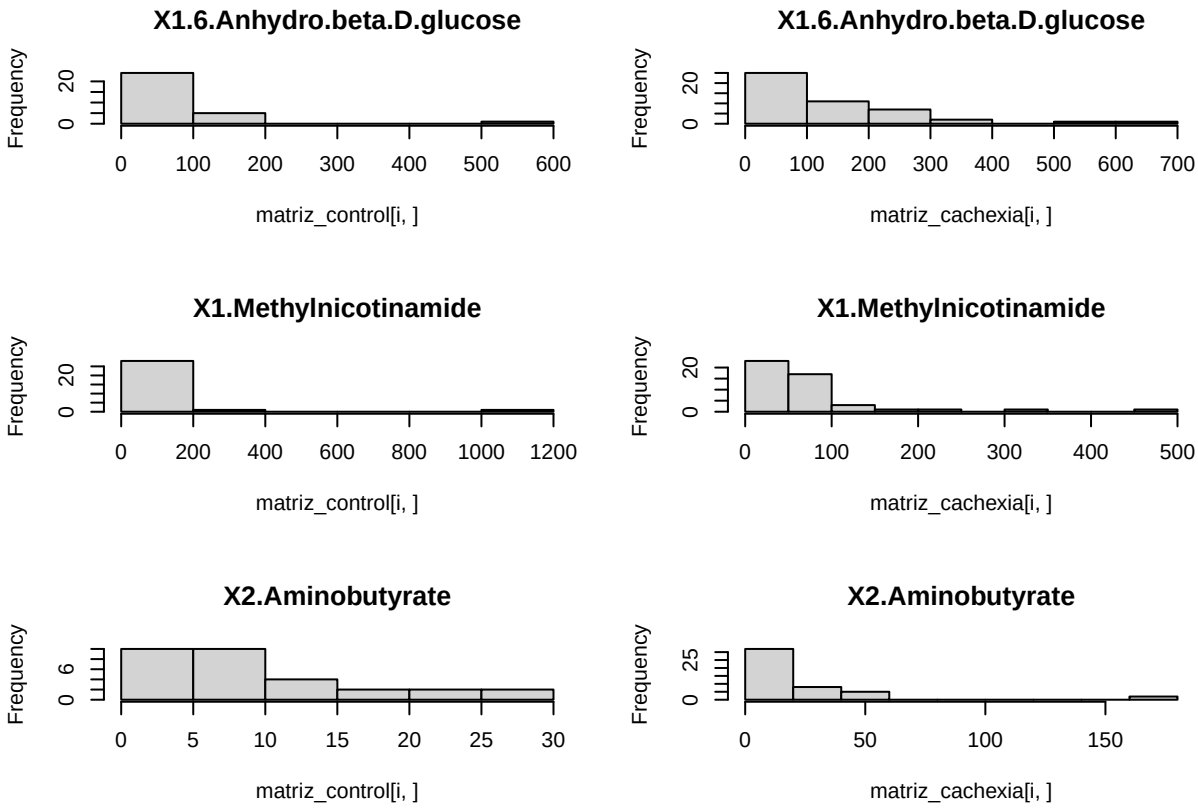
2. Análisis exploratorio de los datos del dataset cachexia.

En primer lugar se va a explorar algunas funciones que se puede hacer con el objeto `se`.

Para poder comparar los datos entre el grupo de caquexia y el grupo control, voy a crear dos matrices distintas con los datos de cada uno de los grupos.

Después voy a comenzar ocn el análisis exploratorio de alguno de los datos. En primer lugar, utilizo la función par para representar histogramas con los niveles de algunos de los 3 primeros metabolitos de las matrices de control y caquexia. Como todavía no se ha explorado en profundidad los datos, no obtenemos grandes diferencias. Podría ser más interesante repetir esto más tarde representando los metabolitos que se expresan de forma más diferente entre los dos grupos. Sin embargo, se incluye dado que es algo que se ha hecho en las actividades previas del reto.

```
matriz_control<-assays(se[,se$Muscle.loss=="control"])$counts
matriz_cachexia<-assays(se[,se$Muscle.loss=="cachexic"])$counts
par(mfrow=c(3,2))
for(i in 1:3){
  hist(matriz_control[i,], main=rownames(matriz_control)[i])
  hist(matriz_cachexia[i,], main=rownames(matriz_cachexia)[i])
}
```



En este punto, vamos a comenzar a dirigir el análisis exploratorio de los datos a encontrar diferencias entre el grupo control y el enfermo.

Para observar las diferencias entre un grupo y el otro, vamos a emplear el test T de student. Se trata de muestras lo suficientemente grandes, de manera que podemos asumir su normalidad, aunque podemos también hacer un test para comprobar la normalidad. Además debemos de saber que los datos no son pareados, lo que se nos ha indicado en los metadatos.

```
# Definimos una función 'ttest' que realiza una prueba t de Student
# La función toma un vector de datos 'x', divide el vector en dos partes (de 1 a 47 y de 48 a 77)
# y luego realiza la prueba t para comparar estas dos partes.
ttest = function(x) {
  tt = t.test(x[1:47], x[48:77])
  # La función devuelve tres valores:El estadístico t, el p-valor y el fold change en log2
  return(c(tt$statistic,
           tt$p.value,
           log(tt$estimate[1]/tt$estimate[2])))
}

# Aplicamos la función 'ttest' a cada fila de los datos (filas de la matriz 'datos_num')
# 'apply' aplica la función a las filas (dim = 1) de la matriz, y guarda el resultado en 'ans'
ans <- apply(datos_num, 1, ttest)


# Extraemos los valores del estadístico t para cada comparación (primera columna de 'ans')
ts <- ans[1,]
```

```r
# Extraemos los p-valores de las pruebas t para cada comparación (segunda columna de 'ans')
pvalues <- ans[2,]


fc <- ans[3,]  # El tercer valor de cada columna de 'ans' corresponde al fold change

# Identificamos las moléculas en las que la diferencia entre los dos grupos es significativa selecciona
sign_mol <- rownames(matriz_cachexia)[pvalues < 0.05]
print(sign_mol)
```

```
##  [1] "X1.6.Anhydro.beta.D.glucose" "X2.Aminobutyrate"
##  [3] "X2.Hydroxyisobutyrate"       "X3.Hydroxybutyrate"
##  [5] "X3.Hydroxyisovalerate"       "X3.Indoxylsulfate"
##  [7] "Acetate"                     "Adipate"
##  [9] "Alanine"                     "Asparagine"
## [11] "Betaine"                     "Carnitine"
## [13] "Citrate"                     "Creatine"
## [15] "Creatinine"                  "Dimethylamine"
## [17] "Ethanolamine"                "Formate"
## [19] "Fucose"                      "Fumarate"
## [21] "Glucose"                     "Glutamine"
## [23] "Glycine"                     "Glycolate"
## [25] "Hippurate"                   "Histidine"
## [27] "Leucine"                     "Methylamine"
## [29] "N.N.Dimethylglycine"         "O.Acetylcarnitine"
## [31] "Pyroglutamate"               "Pyruvate"
## [33] "Quinolinate"                 "Serine"
## [35] "Succinate"                   "Taurine"
## [37] "Threonine"                   "Trigonelline"
## [39] "Trimethylamine.N.oxide"      "Tryptophan"
## [41] "Tyrosine"                    "Valine"
## [43] "cis.Aconitate"               "myo.Inositol"
## [45] "trans.Aconitate"             "tau.Methylhistidine"
```

Se ha obtenido un listado de las moléculas que están diferencialmente presentes en los pacientes normales y con caquexia.

Ahora, se va a mostrar la matriz de correlación entre los diferentes metabolitos para cada uno de los dos grupos, de manera que podremos ver si algunos de los metabolitos et

```r
correlacio<-cor(assays(se)$counts)
```