PAC 1

Pau Pérez Rebassa

Preparació de l'objecte *SummarizedExperiment*

El *dataset* escollit és el *2023-CIMCBTutorial* del github que l'enunciat de la PAC ens ha facilitat. Com es pot veure, tenim un fitxer *.xlsx* anomenat *GastricCancer_NMR* que a dins té dos fulls de càlcul, el primer d'ells conté les dades i el segon les metadades. El primer que s'ha de fer és llegir cada una dels fulls i emmagatzemar-los dins dues variables separades. Gràcies al paquet *readxl* es pot llegir el fitxer.

```
library(readxl)
dades = read_excel("metaboData/Datasets/2023-CIMCBTutorial/GastricCancer_NMR.xlsx",
                   sheet = "Data")
meta = read_excel("metaboData/Datasets/2023-CIMCBTutorial/GastricCancer_NMR.xlsx",
                   sheet = "Peak")
```

Com es pot veure a la informació sobre les dades carregades, el *dataframe dades* té 140 observacions i 153 variables i el *meta* té 149 observacions i 5 variables. S'haurà de modificar les variables de *dades* perquè coincideixi amb les observacions de *meta*. També, el que veim, és que les 4 primeres variables són informació sobre les mostres i la resta de variables són pròpies dades en brut, per tant, el que hem de fer és separar aquest tipus d'informació en dues variable més, de la següent manera:

```
informacio_mostra = dades[, 1:4]
dades = dades[, -c(1:4)]
```

Després d'haver executat les dues intruccions anteriors, tenim separada la informació de la mostra i las dades en brut, d'aquesta manera es veu que ja coincideixen les variables de *dades* i les observacions de *meta*.

Per poder contruir l'objecte *SummarizedExperiment* hem de modificar la informació de la mostra. El primer que feim es convertir en factor els camps *SampleType* i *Class*.

```
informacio_mostra$SampleType = as.factor(informacio_mostra$SampleType)
informacio_mostra$Class = as.factor(informacio_mostra$Class)
```

Per poder seguir endavant en la creació de l'objecte, hem de relacionar d'alguna manera la informació de les mostres amb les dades. Per això cream una nova columna que contendrà un identificador que nosaltres crearem depenent de cada valor de *Idx*, *SampleType*, *Class* de la variable *informació_mostra*, per exmple, en el primer valor quedaria així: ***1__Q__QC***.

```
suppressMessages(suppressPackageStartupMessages(library(dplyr)))
library(dplyr)
informacio_mostra = informacio_mostra %>%
  mutate(new_Id = paste(informacio_mostra$Idx, substr(informacio_mostra$SampleType, 1, 1),
                        informacio_mostra$Class, sep = "_"))
```

```
informacio_mostra = as.data.frame(informacio_mostra)
rownames(informacio_mostra) = informacio_mostra$new_Id
informacio_mostra$new_Id = NULL
```

Eliminam la variable *Idx* i *SampleType* perquè una vegada creat el nou identificador per la mostra, les dues variables contenen informació redundant.

```
informacio_mostra$Idx = NULL
informacio_mostra$SampleID = NULL
head(informacio_mostra)
```

```
##        SampleType Class
## 1_Q_QC         QC    QC
## 2_S_GC     Sample    GC
## 3_S_BN     Sample    BN
## 4_S_HE     Sample    HE
## 5_S_GC     Sample    GC
## 6_S_BN     Sample    BN
```

Generam una matriu amb les dades en cru, per poder crear l'objecte final i afegim el nom de les files que hem creat a *informacio_mostra*, d'aquesta manera ja tenim tot relacionat entre si.

```
matriu_dades = as.matrix(dades)
rownames(matriu_dades) = rownames(informacio_mostra)
```

Ja podem crear l'objecte *SummarizedExperiment* i ho feim de la següent manera:

```
suppressMessages(suppressPackageStartupMessages(library(SummarizedExperiment)))
library(SummarizedExperiment)
experiment = SummarizedExperiment(assays = list(rawValues = matriu_dades),
                                  rowData = informacio_mostra, colData = meta)
show(experiment)
```

```
## class: SummarizedExperiment
## dim: 140 149
## metadata(0):
## assays(1): rawValues
## rownames(140): 1_Q_QC 2_S_GC ... 139_S_HE 140_Q_QC
## rowData names(2): SampleType Class
## colnames(149): M1 M2 ... M148 M149
## colData names(5): Idx Name Label Perc_missing QC_RSD
```

Diferència entre *ExpresionSet* i *SummarizedExperiment*

La classe *ExpressionSe*t és una estructura més antiga, pensada especialment per a dades de microarrays. Aquesta classe permet emmagatzemar una sola matriu de dades (amb expressió gènica, intensitats, etc.) i relacionar-la amb informació sobre les mostres (*phenoData*) i informació sobre les característiques (gens o metabòlits, dins featureData). Tota aquesta informació es guarda en objectes de tipus *AnnotatedDataFrame*, que poden resultar una mica menys intuïtius o menys flexibles si els comparem amb estructures més modernes.

En canvi, la classe *SummarizedExperiment* és més recent i està dissenyada per gestionar de manera més eficient i modular dades d'altres tecnologies com *RNA-seq*, proteòmica o metabolòmica. Aquesta classe

permet incloure múltiples matrius de dades (anomenades *assays*) dins del mateix objecte, cosa que resulta especialment útil si es volen guardar, per exemple, tant els valors bruts com els normalitzats. La informació de les mostres es guarda dins l'objecte *colData*, i la dels metabòlits (o gens) dins *rowData*, ambdós com a *DataFrame* del paquet S4Vectors, més flexible i fàcil de gestionar.

Un altre punt fort de *SummarizedExperiment* és que s'integra millor amb molts paquets actuals del Bioconductor.

Per tant, mentre que *ExpressionSet* només permet una única matriu de dades i està pensat per a anàlisis més clàssics, *SummarizedExperiment* ofereix una estructura més robusta, modular i preparada per als fluxos de treball òmics més actuals.

Anàlisi exploratòria de les dades

Primer de tot, podem analitzar les metadades què seran una bona informació de com estan estrucutrades les dades, si hi ha grups clarament formats o d'altre manera, son totalment independents.

Per això, extreim les metadades del *SummarizedExperiment* i calculam les taules de freqüència dels valors que tenim.

```
meta = rowData(experiment)
table(meta$SampleType)
```

```
##
##     QC Sample
##     17    123
```

Com es pot veure en la primera taula, hi ha aproximàdament un **13,8%** de mostres artificial per fer el *quality control* de l'experiment, dada que entre dins lo raonable.

```
table(meta$Class)
```

```
##
## BN GC HE QC
## 40 43 40 17
```

Dins el tipus de mostra (*SampleType*), es veu que hi ha un repartiment bastant equitatiu que fa que els grups de mostres siguin bastant igualitaris i que doni fiabilitat als resultats de l'experiment.

```
table(meta$SampleType, meta$Class)
```

```
##
##           BN GC HE QC
##   QC       0  0  0 17
##   Sample  40 43 40  0
```

Aquí es veu que les dues variables son totalment independents perquè unes són mostres reals i les altres són de control de qualitat de l'experiment.

Una vegada es te una visió general de les metadades, es pot començar a analitzar les dades en cru que s'han obtingut de l'experiment. Les aïllam en una varible anomenada *valors_crus*.

```r
valors_crus = as.data.frame(t(assay(experiment, "rawValues")))
```

Podem mirar si hi ha algún valor faltant (NA) dins les dades de l'experiment.

```r
percent_na = colSums(is.na(valors_crus)) / nrow(valors_crus) * 100
head(percent_na)
```

```
##   1_Q_QC   2_S_GC   3_S_BN   4_S_HE   5_S_GC   6_S_BN
## 4.026846 4.697987 6.040268 6.040268 2.013423 4.026846
```

```r
summary(percent_na)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  0.6711  3.3557  4.3624  5.1246  6.7114 16.7785
```

Per lo que es pot veure, la mitja dels valors faltants és d'un **5,12%** aproximadament, cosa que és acceptable, però pot afectar als resultats de l'experiment, s'hauria de veure el comput general.