Primera prueba de evaluación contínua (PEC 1)

Carles Llongueras Aparicio

2024-10-24

Contents

Introducción

En este informe se detalla el proceso realizado para la descarga, análisis y posterior subida de un conjunto de datos de metabolómica. Para ello, se utilizará el contenedor *SummarizedExperiment* de la librería *BiocManager* para almacenar los datos necesarios del estudio, y tecnologías como *GitHub* para descargar y guardar en diferentes repositorios los análisis extraídos de estos.

Descarga de los datos

Primeramente, se han descargado los datos pertinentes sobre metabolómica del repositorio de GitHub *https://github.com/nutrimetabolomics/metaboData/*. Esto se ha logrado mediante el siguiente comando de git: *git clone https://github.com/nutrimetabolomics/metaboData.git*.

Seguidamente, dentro de la carpeta Datasets del repositorio que acabamos de descargar, hemos elegido el estudio que más nos ha interesado; en este caso, el referente al estudio de 2018 sobre el análisis de perfiles metabólicos en muestras biológicas, titulado 2018-MetabotypingPaper.

Creación del contenedor

Para comenzar con la creación del contenedor para almacenar nuestros datos, primero verificaremos si el paquete *BiocManager* está instalado, y lo instalaremos en caso de que no lo esté. Luego, tras asegurarnos de que tenemos el paquete necesario, cargaremos el paquete que necesitamos usar, en este caso, *SummarizedExperiment*.

```
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")
BiocManager::install("SummarizedExperiment")
library(SummarizedExperiment)
```

Siguiendo con este punto, almacenaremos en diferentes variables la información necesaria para crear posteriormente el contenedor. En el estudio con el que estamos trabajando, tenemos tres partes diferenciadas: el archivo .csv llamado *DataValues_S013*, que contiene los valores de la información que estamos analizando; el archivo *DataInfo_S013*, que proporciona información sobre las columnas del archivo previamente mencionado y ofrece detalles adicionales sobre las mismas; y, por último, un archivo llamado *description.md*, que contiene información sobre el estudio y explica el contenido de los otros archivos.

Después de haber guardado los tres archivos en tres variables diferentes, realizaremos una comprobación para asegurarnos de que los datos son coherentes entre sí. En este caso, verificaremos que el número de muestras en los datos coincida con el número de muestras en los metadatos. En caso contrario, detendríamos la ejecución, ya que esto indicaría que ha ocurrido algún tipo de error.

```
features <- read.csv("metaboData/Datasets/2018-MetabotypingPaper/DataValues_S013.csv",
                     row.names = 1)

metadata <- read.csv("metaboData/Datasets/2018-MetabotypingPaper/DataInfo_S013.csv",
                     row.names = 1)

metadataPath <- readLines("metaboData/Datasets/2018-MetabotypingPaper/description.md")

metadataDescription <- paste(metadataPath)
```

```r
if (ncol(features) != nrow(metadata)) {
  stop("El número de muestras en los datos y los metadatos no coinciden.")
}
```

Por último, con toda la información guardada y disponible, crearemos el objeto *SummarizedExperiment* invocando su función creadora y utilizando los diferentes archivos como parámetros. El primer parámetro será una lista de matrices con los distintos datos de nuestro estudio; en segundo lugar, pasaremos los metadatos en el mismo formato en que los hemos leído; y, por último, incluiremos la descripción del estudio.

```r
se <- SummarizedExperiment(assays = list(assays = as.matrix(features)),
                           colData = metadata, metadata = metadataDescription)
```

Exploración del Dataset

Ahora que ya hemos creado el objeto de tipo *SummarizedExperiment* podemos proceder a su exploración para conseguir una visión general del mismo. Primeramente haremos uso de la función *class* para determinar que, efectaivamente el objeto que hemos creado es del tipo que queremos.

```r
class(se)
```

```
## [1] "SummarizedExperiment"
## attr(,"package")
## [1] "SummarizedExperiment"
```

En segundo lugar, utilizamos la función *dim* para acceder a las dimensiones del objeto *SummarizedExperiment* que hemos creado. En este caso, observamos que el objeto contiene 39 filas y 695 columnas, lo que significa que disponemos de datos de 39 muestras (es decir, observaciones individuales) y 695 variables medidas en cada muestra. Esta configuración, con un número limitado de muestras, plantea desafíos únicos debido a la escasez de datos. Sin embargo, gracias al gran número de variables, es posible descubrir patrones complejos y asociaciones significativas entre las muestras

```r
dim(se)
```

```
## [1]  39 695
```

Siguiendo con el análisis, podemos utilizar la función *colData* para obtener los metadatos de las muestras del dataset. En este caso, observamos tres columnas que proporcionan información adicional sobre los datos:

- VarName: Se utiliza para vincular los datos con los metadatos (actuando como una clave primaria, o Primary Key si habláramos en términos de bases de datos).
- VarType: Indica el tipo de dato de cada variable (por ejemplo, integer o character), lo cual facilita la interpretación y el procesamiento posterior.
- Description: Asigna una descripción o categoría a cada variable, ayudando a comprender mejor su propósito o el tipo de información que contiene.

```r
colData(se)
```

```
## DataFrame with 695 rows and 3 columns
##                 VarName      varTpe Description
##             <character> <character> <character>
## SUBJECTS       SUBJECTS     integer    dataDesc
## SURGERY         SURGERY   character    dataDesc
## AGE                 AGE     integer    dataDesc
## GENDER           GENDER   character    dataDesc
## Group             Group     integer    dataDesc
## ...                 ...         ...         ...
## SM.C18.0_T5 SM.C18.0_T5     numeric    dataDesc
## SM.C18.1_T5 SM.C18.1_T5     numeric    dataDesc
## SM.C20.2_T5 SM.C20.2_T5     numeric    dataDesc
## SM.C24.0_T5 SM.C24.0_T5     numeric    dataDesc
## SM.C24.1_T5 SM.C24.1_T5     numeric    dataDesc
```

Después de haber revisado la estructura de los datos y los metadatos, lo pertinente sería observar los datos en sí. En este caso, podemos utilizar la función *dimnames* para mostrar los nombres de las filas y las columnas de las variables del estudio.

Al ejecutar esta función, podemos observar que las filas corresponden a los 39 sujetos de los cuales se ha obtenido la información, numerados del 1 al 39. Estos números no aportan información adicional, aparte de ser una forma de identificar las muestras.

Por otro lado, tenemos los nombres de las columnas, que contienen la mayoría de la información del dataset. En este caso, hemos limitado la salida a 20 nombres, dado que es difícil visualizar una tabla de 695 filas donde la mayoría de los nombres corresponden a información metabólica que, individualmente, no aporta mucho. Por esta razón, hemos acotado la salida a 20 para poder tener una vista más específica de la información que existe. En ella, podemos ver los nombres de diferentes columnas, como la edad, el sexo, el grupo al que pertenecen e incluso el tipo de cirugía que han recibido, junto con toda la información metabólica que se ha podido extraer de las muestras.

```
dimnames(se)[[1]]
```

```
##  [1] "1"  "2"  "3"  "4"  "5"  "6"  "7"  "8"  "9"  "10" "11" "12" "13" "14" "15"
## [16] "16" "17" "18" "19" "20" "21" "22" "23" "24" "25" "26" "27" "28" "29" "30"
## [31] "31" "32" "33" "34" "35" "36" "37" "38" "39"
```

```
head(dimnames(se)[[2]], 20)
```

```
##  [1] "SUBJECTS"          "SURGERY"           "AGE"
##  [4] "GENDER"            "Group"             "MEDDM_T0"
##  [7] "MEDCOL_T0"         "MEDINF_T0"         "MEDHTA_T0"
## [10] "GLU_T0"            "INS_T0"            "HOMA_T0"
## [13] "HBA1C_T0"          "HBA1C.mmol.mol_T0" "PESO_T0"
## [16] "bmi_T0"            "CC_T0"             "CINT_T0"
## [19] "CAD_T0"            "TAD_T0"
```

Otra manera de acceder a los datos sería mediante la función *assays*, que no solo te muestra cuántas matrices de datos están asociadas con el objeto (en nuestro caso, solo tenemos una), sino que, como se puede ver en la ejecución a continuación, permite acceder de manera individual y precisa a la parte que deseas observar. Esto la convierte en una herramienta valiosa para aislar y analizar secciones específicas del conjunto de datos.

```
assays(se)
```

```
## List of length 1
## names(1): assays
```

```
assays(se)[[1]][1:5,1:5]
```

```
##   SUBJECTS SURGERY   AGE  GENDER Group
## 1 " 1"     "by pass" "27" "F"    "1"
## 2 " 2"     "by pass" "19" "F"    "2"
## 3 " 3"     "by pass" "42" "F"    "1"
## 4 " 4"     "by pass" "37" "F"    "2"
## 5 " 5"     "tubular" "42" "F"    "1"
```

Por último, para ver la metadata asociada con el objeto en relación con la información sobre el experimento, utilizaremos la función *metadata*. Aquí podemos encontrar el texto que contiene información descriptiva del estudio, que sus creadores consideraron importante tener en cuenta.

```
metadata(se)
```

```
## [[1]]
## [1] "Data used in the paper \"Metabotypes of response to bariatric surgery independent of the magnitu
##
## [[2]]
## [1] ""
##
## [[3]]
## [1] "The data has been published in the paper web site and it also has an independent repository [ht
##
## [[4]]
## [1] ""
##
## [[5]]
## [1] "The dataset is formed by the following files :"
##
## [[6]]
## [1] ""
##
## [[7]]
## [1] "- DataInfo_S013.csv: Metadata. Information on each column in the \"DataValues_S013.csv\" file."
##
## [[8]]
## [1] "- DataValues_S013.csv: Clinical and metabolomic values for 39 patients at 5 time points."
##
## [[9]]
## [1] "- AAInformation_S006.csv: Additional information on metabolites in the \"DataValues_S013.csv\" f
##
## [[10]]
## [1] ""
```

Subida de los datos

Primeramente, accederemos a nuestra cuenta de GitHub (es necesario crear una si no tenemos una). Haremos clic en el botón *New* y le asignaremos un nombre, que en nuestro caso debe seguir la siguiente estructura: APELLIDO1-Apellido2-Nombre-PEC1. Posteriormente, podemos añadir una breve descripción del contenido y la finalidad del repositorio, así como indicar si queremos que se cree un README automáticamente. Después de estos pasos, dado que la visibilidad de nuestro repositorio será pública por defecto, estaremos listos para subir todos los archivos necesarios.

Después de haber creado un repositorio, el primer paso será descargarlo para tenerlo en nuestra máquina local. Como hemos hecho antes para el análisis, usaremos el comando *git clone https://github.com/CarlesLlongueras/Llongueras-Aparicio-Carles-PEC1* para descargar el repositorio vacío en cualquier parte de nuestro ordenador.

Una vez que tengamos el repositorio vacío, guardaremos toda la información que queremos subir al mismo: el informe, el objeto .RDA, el código de R, los datos en formato texto y los metadatos en formato RMD. Toda la información es trivial de conseguir, excepto el objeto .RDA, el cual podemos obtener con el siguiente comando: *save(se, file = "name.Rda").*

Una vez que tengamos todos los archivos dentro de la carpeta, ejecutaremos por consola los siguientes tres comandos para subir los archivos a nuestro repositorio de GitHub: primero, añadiremos los archivos a nuestro siguiente commit con *git add .*, en segundo lugar, haremos el commit con *git commit -m "Descripción"*, y, por último, haremos push de este commit a la rama principal con *git push origin main.*

URL del repositorio: https://github.com/CarlesLlongueras/Llongueras-Aparicio-Carles-PEC1