methylation

Introduction

Install the following packages:

```
## Loading required package: Biobase

## Loading required package: BiocGenerics

## Loading required package: parallel

##
## Attaching package: 'BiocGenerics'

## The following objects are masked from 'package:parallel':
##
##     clusterApply, clusterApplyLB, clusterCall, clusterEvalQ,
##     clusterExport, clusterMap, parApply, parCapply, parLapply,
##     parLapplyLB, parRapply, parSapply, parSapplyLB

## The following objects are masked from 'package:stats':
##
##     IQR, mad, sd, var, xtabs

## The following objects are masked from 'package:base':
##
##     anyDuplicated, append, as.data.frame, basename, cbind, colnames,
##     dirname, do.call, duplicated, eval, evalq, Filter, Find, get, grep,
##     grepl, intersect, is.unsorted, lapply, Map, mapply, match, mget,
##     order, paste, pmax, pmax.int, pmin, pmin.int, Position, rank,
##     rbind, Reduce, rownames, sapply, setdiff, sort, table, tapply,
##     union, unique, unsplit, which.max, which.min

## Welcome to Bioconductor
##
##     Vignettes contain introductory material; view with
##     'browseVignettes()'. To cite Bioconductor, see
##     'citation("Biobase")', and for packages 'citation("pkgname")'.

## Setting options('download.file.method.GEOquery'='auto')

## Setting options('GEOquery.inmemory.gpl'=FALSE)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:Biobase':
##
##     combine

## The following objects are masked from 'package:BiocGenerics':
##
##     combine, intersect, setdiff, union

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

## Loading required package: usethis

##
## Attaching package: 'limma'

## The following object is masked from 'package:BiocGenerics':
##
##     plotMA

##
## Attaching package: 'kableExtra'

## The following object is masked from 'package:dplyr':
##
##     group_rows
```

Importing the data

```
## Warning: One or more parsing issues, see `problems()` for details
## One or more parsing issues, see `problems()` for details
```

Some datasets on GEO may be derived from different microarray platforms. Therefore the object gse is a list of different datasets. You can find out how many were used by checking the length of the gse object. Usually there will only be one platform and the dataset we want to analyse will be the first object in the list (gse[[1]]).

```
length(gse)
```

```
## [1] 1
```

Extract the data

```r
gse <- gse[[1]]
gse
```

```
## ExpressionSet (storageMode: lockedEnvironment)
## assayData: 27578 features, 55 samples
##   element names: exprs
## protocolData: none
## phenoData
##   sampleNames: GSM813533 GSM813534 ... GSM813587 (55 total)
##   varLabels: title geo_accession ... tissue:ch1 (58 total)
##   varMetadata: labelDescription
## featureData
##   featureNames: cg00000292 cg00002426 ... cg27665659 (27578 total)
##   fvarLabels: ID Name ... ORF (38 total)
##   fvarMetadata: Column Description labelDescription
## experimentData: use 'experimentData(object)'
##   pubMedIds: 22613842
## Annotation: GPL8490
```

Exploratory analysis

The `exprs` function can retrieve the expression values as a data frame; with one column per-sample and one row per-gene.

```r
pdata= pData(gse) #sample information
edata= exprs(gse) #expression data
fdata = fData(gse) #gene annotation
```

Inspect the clinical variables

Data submitted to GEO contain sample labels assigned by the experimenters, and some information about the processing protocol. All these data can be extracted by the pData function.