GEO Tutorial

Asperger, Cieslik

08/12/2021

Setup

Install required libraries.

```r
if (!requireNamespace("BiocManager", quietly = TRUE))
    install.packages("BiocManager")

BiocManager::install("GEOquery")
BiocManager::install("limma")
install.packages("umap")
install.packages("rmarkdown")
install.packages("knitr")
```

```r
library(GEOquery)
library(limma)
library(umap)
```

Define IDs and other variables

```r
# define GSE and GPL accession ID
GSE_id <- "GSE72205"
GPL_id <- "GPL8432"

# define column which holds gene symbol
gene_symbol <- "ILMN_Gene"
```

Get GEO datasets

GSE3893 - https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE3893 GSE72205 - https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE72205 GSE39567 - https://www.ncbi.nlm.nih.gov/geo/query/acc.cgi?acc=GSE39567

```r
# download GSE dataset; declare a destination directory (destdir) if you want
# to avoid downloading the same files again next time you analyze them
gset <- getGEO(GSE_id, GSEMatrix = TRUE, AnnotGPL = TRUE)  #, destdir = 'temp'

# select samples based on GPL platform (if there is only one platform, choose
# it automatically)
if (length(gset) > 1) {
    idx <- grep(GPL_id, attr(gset, "names"))
} else {
    idx <- 1
}
```

```r
gset <- gset[[idx]]

# download GPL data; declare a destination directory (destdir) if you want to
# avoid downloading the same files again next time you analyze them
gpl <- getGEO(GPL_id)  #, destdir = 'temp'

# delete local variables to keep workspace clean
rm("idx")
```

Extract data from gset

```r
# exprs(gset) - Retrieve expression data from eSets. pData(gset) - Retrieve
# information on experimental phenotypes

# filter out invalid probes and empty ones
geneProbes <- which(!is.na(Table(gpl)$ID))
probeids <- as.character(Table(gpl)$ID[geneProbes])
probes <- intersect(probeids, rownames(exprs(gset)))

# extract expression data (only for valid probes)
geneMatrix <- exprs(gset)[probes, ]

# get annotation data from gpl and append it to the expression table
inds <- which(Table(gpl)$ID %in% probes)
geneMatrix <- cbind(geneMatrix, Table(gpl)[inds, gene_symbol, drop = FALSE])

# create matrix for expression data with gene names as row names
exprData <- geneMatrix
rownames(exprData) <- make.unique(exprData[, gene_symbol])
exprData <- exprData[, 1:length(exprData) - 1]

# extract phenotype data
pData <- pData(gset)

# delete local variables to keep workspace clean
rm("geneProbes", "probeids", "probes", "inds", "geneMatrix")
```

Display the fetched data.

```r
exprData[1:5, 1:5]
```

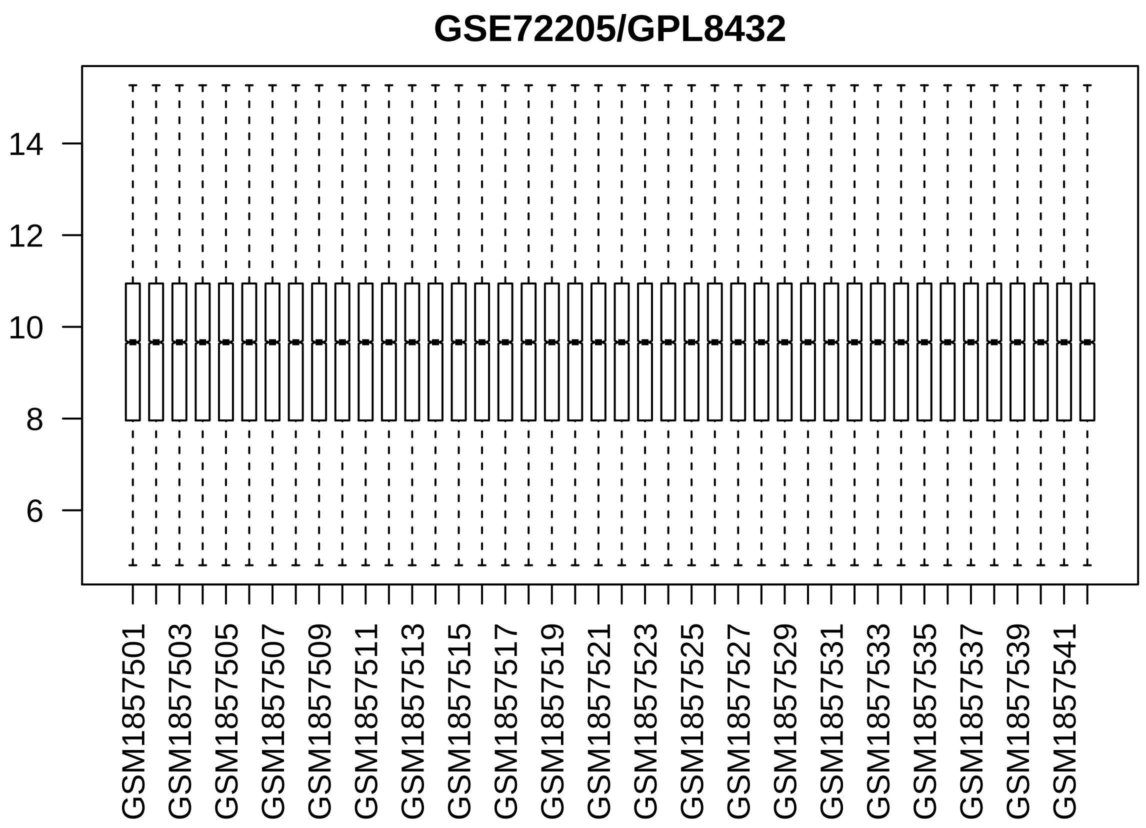
```
##          GSM1857501 GSM1857502 GSM1857503 GSM1857504 GSM1857505
## PHTF2     10.673118  10.512922  10.078586   8.627445  11.101501
## TRIM44    10.473174  10.121392   6.113813   7.312077  10.719397
## DGAT2L3    8.725645   6.839317   6.215991   9.833697   6.104573
## C15ORF39   8.577733   8.335006   7.398104   6.251762   9.576227
## PCDHGA9   10.404744   9.999729   8.634718   8.387246   9.073057
```

```r
pData[1:5, 1:5]
```

```
##             title geo_accession                status submission_date
## GSM1857501 DCIS 1    GSM1857501 Public on Aug 01 2016     Aug 19 2015
## GSM1857502  IDC 1    GSM1857502 Public on Aug 01 2016     Aug 19 2015
## GSM1857503 DCIS 2    GSM1857503 Public on Aug 01 2016     Aug 19 2015
## GSM1857504  IDC 2    GSM1857504 Public on Aug 01 2016     Aug 19 2015
## GSM1857505 DCIS 3    GSM1857505 Public on Aug 01 2016     Aug 19 2015
##            last_update_date
## GSM1857501      Aug 01 2016
## GSM1857502      Aug 01 2016
## GSM1857503      Aug 01 2016
## GSM1857504      Aug 01 2016
## GSM1857505      Aug 01 2016
```

Plot

```
palette(c("#dfeaf4", "#f4dfdf", "#AABBCC"))
par(mar = c(2 + round(max(nchar(sampleNames(gset)))/2), 4, 2, 1))
title <- paste0(GSE_id, "/", annotation(gset))
boxplot(exprs(gset), boxwex = 0.6, notch = T, main = title, outline = FALSE, las = 2)
```



```
# Clear unused variables
rm("title")
```

Create groups

```
# delete all 'pure DCIS' entries as they have no corresponding tissue
idx <- which(pData[, "disease state:ch1"] != "pure DCIS")
pData_sub <- pData[idx, ]

pData_sub$patient <- NA

for (i in 1:nrow(pData_sub)) {
    pData_sub[i, "patient"] <- strsplit(as.character(pData_sub[i, "title"]), split = " ")[[1]][2]
}
```

```r
pData_sub$patient <- as.numeric(pData_sub$patient)
pData_sub <- pData_sub[order(pData_sub[, "patient"]), ]

# save index of sample in the corresponding grouping list
group_A <- which(pData_sub[, "disease state:ch1"] == "DCIS")
group_B <- which(pData_sub[, "disease state:ch1"] == "IDC")

# subset exprData into two matrixes
exprData_A <- exprData[, group_A]
exprData_B <- exprData[, group_B]
```

T-Test

```r
# get all protein names
proteins <- rownames(exprData)

# create empty matrix to speed up the looping step
result_ttest <- matrix(nrow = length(proteins), ncol = 5)
colnames(result_ttest) <- c("p.value", "q.value", "median.A", "median.B", "median.ratio")
rownames(result_ttest) <- proteins
for (i in 1:length(proteins)) {
    protein <- proteins[i]
    expr_A <- unlist(exprData_A[i, ])
    expr_B <- unlist(exprData_B[i, ])
    result_ttest[i, "p.value"] <- t.test(expr_A, expr_B, alternative = "two.sided")$p.val
    result_ttest[i, "median.A"] <- median(expr_A)
    result_ttest[i, "median.B"] <- median(expr_B)
}

result_ttest[, "q.value"] <- p.adjust(result_ttest[, "p.value"], method = "hochberg")
result_ttest[, "median.ratio"] <- result_ttest[, "median.A"]/result_ttest[, "median.B"]
result_ttest <- result_ttest[order(result_ttest[, "q.value"], decreasing = F), ,
    drop = FALSE]

# clear unused variables
rm("i", "expr_A", "expr_B", "protein")
```