Selecting differentially expressed genes (DEG) with microarrays

Jacques van Helden and Denis Puthier

8 November 2014

Objectives

In this tutorial, we will load microarray data from DenBoer et al. (2009), and perform statistical tests to select differentially expressed genes (DEG).

Through this tutorial, we we learn to do the following operations in R:

- define file paths by concatenating different parts of them (base URL, base directory, sub-directories)
- create a directory on our computer to keep a local copy of the files
- download a files from a Web site
- load the content of these files into an R data structure called "data.frame", which enables to handle tables with named rows and columns
- inspect the content of the data tables

Configuration

Before running the analyses, we will load some configuration script, which will define the source directory, and create a default output directory for the results.

```
## Specify the base URL for the course
base.url <- "http://pedagogix-tagc.univ-mrs.fr/courses/statistics_bioinformatics"

## Load the general configuration file
source(file.path(base.url, 'R-files', 'config.R'))

## Set working directory to the results directory
setwd(dir.results)
```

Downoading the dataset from DenBoer, 2009

Define the location of the data from DenBoer, relative to the base URL.

```
url.denboer <- file.path(base.url, 'data', 'gene_expression','denboer_2009')
```

We define a path on our own computed, where the data will be downloaded.

```
## Define the path of the local copy of the data
dir.denboer <- file.path(dir.results, "denboer_2009_analysis")
dir.denboer.data <- file.path(dir.denboer,"data")
dir.create(dir.denboer.data,showWarnings=FALSE, recursive=TRUE)
```

```
## Print the location of the future local copy of the expression table
print(paste("Local copy of the data will be stored in directory",
            dir.denboer.data, sep="\n"))
```

```
## [1] "Local copy of the data will be stored in directory\n/Users/jvanheld/course_stats_bioinfo/results
```

We will now download the expression table, which contains one row per gene and one column per sample.

BEWARE: the whole file makes 20Mb. Depending on the bandwidth of your internet access, it can take a few seconds to several minutes to be downloaded from the Web site and loaded in R.

Note that it is not necessary to keep a local copy of the data tables. R allows you to directly read files from a Web site. However, since the file is big, we propose here to download it only once and keep a local copy, which can then be re-used if we need to reload it for the next sessions of the course. This costs us 22Mb of storage space, but minimize the transfer time.

```
## Download the expression table only f required
## (if the file is already in the result dir, avoid transferring multiple times)
for (file in c("GSE13425_Norm_Whole.txt", "phenoData_GSE13425.tab")) {
  local.file <- file.path(dir.denboer.data, file)
  if (file.exists(local.file)) {
    print (paste("Skipping download of file", file, " (Local copy already exists)."))
  } else {
    ## Define the URL of the data file on Web server of this course
    file.url <- file.path(url.denboer, file)
    ## Download the file
    download.file(file.url, local.file, method="wget", quiet = TRUE)
  }
}
```

```
## [1] "Skipping download of file GSE13425_Norm_Whole.txt  (Local copy already exists)."
## [1] "Skipping download of file phenoData_GSE13425.tab  (Local copy already exists)."
```

Loading the expression table

We now dispose of a local copy of the expression table, we can load it in R, in a variable called expr.matrix. We use the command read.table(), which will load the table in a data structure of the class data.frame().

```
## Load the expression table in a data.frame calles "expr.matrix"
file.expr.matrix <- file.path(dir.denboer.data, "GSE13425_Norm_Whole.txt")
expr.matrix <- read.table(file.expr.matrix, sep = "\t", head = T, row = 1)

## Let us theck the dimensions of the expression table.
dim(expr.matrix)
```

```
## [1] 22283   190
```

The dim() command prints the dimensions of a table. It displays "22283 190", indicating that the expression matrix contains 22,283 rows (genes) and 190 colums (samples).

We can display the first 5 columns of the 10 top rows, in order to check the content

```
print(expr.matrix[1:10,1:5])
```

```
##                  GSM338666 GSM338667 GSM338668 GSM338669 GSM338670
## DDR1|1007_s_at        4.38      4.38      4.23      3.98      5.12
## RFC2|1053_at          3.44      3.33      3.73      3.58      3.40
## HSPA6|117_at          2.51      2.62      2.37      2.43      2.49
## PAX8|121_at           6.24      6.21      6.10      6.09      6.53
## GUCA1A|1255_g_at      2.29      2.23      2.17      2.26      2.27
## UBA7|1294_at          5.57      5.83      5.46      5.20      6.83
## THRA|1316_at          3.09      3.34      3.10      3.20      3.16
## PTPN21|1320_at        2.35      2.46      2.30      2.34      2.21
## CCL5|1405_i_at        2.45      2.52      2.48      2.44      2.22
## CYP2E1|1431_at        2.33      2.42      2.18      2.15      2.27
```

Load the "phenotype" table, which provides a description of each sample (one row per sample, one column per description field).

```
pheno <- read.table(file.path(dir.denboer.data, "phenoData_GSE13425.tab"),
                    sep='\t', head=TRUE, row=1)
dim(pheno)
```

```
## [1] 190   4
```

We will use some convenient R commands (names(), rownames, head()) to inspect the content of the pheno table.

```
## Print column names of the phenotype table
names(pheno)
```

```
## [1] "Sample.title"               "Sample.source.name.ch1"
## [3] "Sample.characteristics.ch1" "Sample.description"
```

```
## Print the names of the first 20 rows
rownames(pheno)[0:20]
```

```
##  [1] "GSM338666" "GSM338667" "GSM338668" "GSM338669" "GSM338670"
##  [6] "GSM338671" "GSM338672" "GSM338673" "GSM338674" "GSM338675"
## [11] "GSM338676" "GSM338677" "GSM338678" "GSM338679" "GSM338680"
## [16] "GSM338681" "GSM338682" "GSM338683" "GSM338684" "GSM338685"
```

Print the values of the field Sample.title of pheno Since there are 190 values, we print a random sample of 20 of them, just to get an idea of what they look like.

```
print(sample(pheno$Sample.title,size=20))
```

```
##  [1] TEL-AML1               hyperdiploid           MLL
##  [4] MLL                    T-ALL                  hyperdiploid
##  [7] TEL-AML1               E2A-rearranged (E-sub) TEL-AML1
## [10] hyperdiploid           pre-B ALL              hyperdiploid
## [13] pre-B ALL              TEL-AML1               T-ALL
## [16] TEL-AML1               TEL-AML1               pre-B ALL
## [19] pre-B ALL              hyperdiploid
## 11 Levels: BCR-ABL BCR-ABL + hyperdiploidy ... TEL-AML1 + hyperdiploidy
```

We can also cound the number of samples per cancer type, sorted by decreasing order. For this, we use the command table()R allows to associate a label to each entry of the vector. In our case, each entry of the vector will contain the abbreviation (Bch, Bc, ...) and the vector labels will indicate the complete description of the corresponding cancer type.

```
## Defined the abbreviations for each ALL type
group.abbrev <- c(
  'BCR-ABL + hyperdiploidy'='Bch',
  'BCR-ABL'='Bc',
  'E2A-rearranged (E)'='BE',
  'E2A-rearranged (E-sub)'='BEs',
  'E2A-rearranged (EP)'='BEp',
  'MLL'='BM',
  'T-ALL'='T',
  'TEL-AML1 + hyperdiploidy'='Bth',
  'TEL-AML1'='Bt',
  'hyperdiploid'='Bh',
  'pre-B ALL'='pB'
)

## Check the content
print(group.abbrev)
```

```
##  BCR-ABL + hyperdiploidy                   BCR-ABL       E2A-rearranged (E)
##                    "Bch"                      "Bc"                      "BE"
##   E2A-rearranged (E-sub)      E2A-rearranged (EP)                       MLL
##                    "BEs"                     "BEp"                      "BM"
##                    T-ALL TEL-AML1 + hyperdiploidy                 TEL-AML1
##                      "T"                     "Bth"                      "Bt"
##             hyperdiploid                 pre-B ALL
##                     "Bh"                      "pB"
```

We will now define two other vectors, indicating: * the cancer type (values of the vector) associated to each sample (labels of the vector); * the abbreviation of the cancer type (values) for each sample (labels).

```
## Define a vector indicating the subtype of each sample
sample.subtypes <- as.vector(pheno$Sample.title)
names(sample.subtypes) <- names(expr.matrix)
head (sample.subtypes) ## Check first lines of the result
```

```
## GSM338666 GSM338667 GSM338668 GSM338669 GSM338670 GSM338671
##   "T-ALL"   "T-ALL"   "T-ALL"   "T-ALL"   "T-ALL"   "T-ALL"
```

```
tail (sample.subtypes) ## Check last lines of the result
```

```
##   GSM338850   GSM338851   GSM338852   GSM338853   GSM338854   GSM338855
## "pre-B ALL" "pre-B ALL" "pre-B ALL" "pre-B ALL" "pre-B ALL" "pre-B ALL"
```

```
## Define a vector with the abbreviated subtype for each sample
sample.labels <- group.abbrev[sample.subtypes]
names(sample.labels) <- names(expr.matrix)
head(sample.labels)
```

```
## GSM338666 GSM338667 GSM338668 GSM338669 GSM338670 GSM338671
##       "T"       "T"       "T"       "T"       "T"       "T"
```

```
tail(sample.labels)
```

```
## GSM338850 GSM338851 GSM338852 GSM338853 GSM338854 GSM338855
##      "pB"      "pB"      "pB"      "pB"      "pB"      "pB"
```

Summary

This part of the tutorial is now completed.

We loaded two data strucutres: * expr.matrix: an expression table, containing the expression values for 22.283 genes (rows) in 190 cancer samples (columns). * pheno: a phenotype table, providing a description of the samples.

Based on this initial data, we defined some additional vectors that will be used for display purpose: * group.abbrev: a vector with abbreviations (values) for each cancer type (labels) * sample.subtypes: a vector with the cancer subtype (values) for each sample (labels) * sample.labels: a vector with the abbreviated cancer subtype (values) for each sample (labels)