Selecting differentially expressed genes with R or TmeV

Denis Puthier and Jacques van Helden

2015-03-06

Contents

Retrieving the den Boer normalized dataset

Here we will use data from the microarray series GSE13425, which which was retrieved from the Gene Expression Omnibus (GEO) database. In this experiment, the authors applied a supervised classification method to define a transcriptomic signature, in order to classify samples from acute lymphoblastic leukemia (ALL). Lymphoblastic leukemia is characterized by the abnormal clonal proliferation, within the bone marrow, of lymphoid progenitors blocked at a precise stage of their differentiation.

Data were produced using Affymetrix geneChips (Affymetrix Human Genome U133A Array, HGU133A). Information related to this platform are available on GEO website under identifier GPL96.

Loading data into R

Protocol

- Start R.
- Have a look at the description of the **read.table()** function.
- We will now load three data tables into R using the read.table function. The function allows us to directly read the tables from the web server. We will successively load 3 files providing complementary information.
  - the expression matrix (GSE13425_Norm_Whole.txt)
    - Contains genes as rows and samples as columns.
    - Data were previously normalized using rma algorithm (they are thus transformed in logarithm base 2).
  - the A/P/M matrix (GSE13425_AMP_Whole.txt)
    - Indicates whether a gene was called **A**bsent, **P**resent or **M**arginal.
  - Phenotypic data (GSE13425_phenoData.txt)
    - The GSE13425_phenoData.txt file contains phenotypic information about samples.

```
## Get some help about the read.table fonction
#?read.table

## Define the URL of the example data sets
url.course <- "http://pedagogix-tagc.univ-mrs.fr/courses/ASG1"
url.base <- file.path(url.course, "data/marrays/")

## Load expression values
expr.file <- file.path(url.base, "GSE13425_Norm_Whole.txt")
expr.matrix <-  read.table(expr.file,sep="\t", head=T, row=1)

## Load phenotypic data
pheno <- read.table(file.path(url.base, 'phenoData_GSE13425.tab'),
                    sep='\t', head=TRUE, row=1)

## Load Absent/Marginal/Present (AMP) calls
amp <- read.table(file.path(url.base, "GSE13425_AMP_Whole.txt"),
                  sep="\t", head=T, row=1)
```

We will now define a directory to store the results on our computer.

```
## Define the output directory. You can adapt this to your local configuration.
dir.output <- "~/ASG1_practicals/GSE13425"

## Create the output directory (if it does not exist yet)
dir.create(dir.output, showWarnings=FALSE, recurs=TRUE)

## Change directory to dir.output
setwd(dir.output)
```

Exercise

- How many rows and columns does the object expr.matrix contain
- Does it correspond to the dimensions of the A/P/M matrix ?
- Which information is available about samples ?
- How many samples from each tumor subtype are present in the DenBoer dataset ?

View solution| Hide solution

```
## Check the dimension of the different tables
# an alternative is to use nrow and ncol
dim(expr.matrix)
```

Solution

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

```
dim(amp)
```

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

```
dim(pheno)
```

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

```
colnames(pheno)
```

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

The field “sample title” of the pheno table indicates the subtype of each ALL tumour. We can use the R function table() to count the number of samples assigned to each tumour class.

```
table(pheno$Sample.title)
```

```
##
##                  BCR-ABL  BCR-ABL + hyperdiploidy   E2A-rearranged (E-sub)
##                        4                        1                        4
##       E2A-rearranged (E)      E2A-rearranged (EP)             hyperdiploid
##                        1                        8                       44
##                      MLL                pre-B ALL                    T-ALL
##                        4                       44                       36
##                 TEL-AML1 TEL-AML1 + hyperdiploidy
##                       43                        1
```

We can convert the vector to a single-column data frame, to enhance its readability, and use this data frame to select the subtypes represented by at least 10 samples.

```r
print(as.data.frame(table(pheno$Sample.title)))
```

```
##                          Var1 Freq
## 1                     BCR-ABL    4
## 2   BCR-ABL + hyperdiploidy    1
## 3    E2A-rearranged (E-sub)    4
## 4        E2A-rearranged (E)    1
## 5       E2A-rearranged (EP)    8
## 6              hyperdiploid   44
## 7                       MLL    4
## 8                 pre-B ALL   44
## 9                     T-ALL   36
## 10                 TEL-AML1   43
## 11 TEL-AML1 + hyperdiploidy    1
```

```r
## Sort subtypes by decreasing number of samples
samples.per.subtype <- as.data.frame(sort(table(pheno$Sample.title),
                                          decreasing=TRUE))
print(samples.per.subtype)
```

```
##                          sort(table(pheno$Sample.title), decreasing = TRUE)
## hyperdiploid                                                             44
## pre-B ALL                                                                44
## TEL-AML1                                                                 43
## T-ALL                                                                    36
## E2A-rearranged (EP)                                                       8
## BCR-ABL                                                                   4
## E2A-rearranged (E-sub)                                                    4
## MLL                                                                       4
## BCR-ABL + hyperdiploidy                                                   1
## E2A-rearranged (E)                                                        1
## TEL-AML1 + hyperdiploidy                                                  1
```

```r
## Select subtypes represented by at least 10 samples
samples.per.subtype > 10
```

```
##                          sort(table(pheno$Sample.title), decreasing = TRUE)
## hyperdiploid                                                           TRUE
## pre-B ALL                                                              TRUE
## TEL-AML1                                                               TRUE
## T-ALL                                                                  TRUE
## E2A-rearranged (EP)                                                   FALSE
## BCR-ABL                                                               FALSE
## E2A-rearranged (E-sub)                                                FALSE
## MLL                                                                   FALSE
## BCR-ABL + hyperdiploidy                                               FALSE
## E2A-rearranged (E)                                                    FALSE
## TEL-AML1 + hyperdiploidy                                              FALSE
```

```r
rownames(samples.per.subtype)[samples.per.subtype > 10]
```

```
## [1] "hyperdiploid" "pre-B ALL"    "TEL-AML1"     "T-ALL"
```

Interpretation

The dataset from DenBoer contains **190 samples** belonging to **various tumour classes**. We can already notice that there is an **important imbalance** between the sizes of the tumour classes: T-ALL, pre-B ALL, TEL-AML1 and hyperdiploid are each represented by more than 40 samples, whereas the other classes (e.g. BCR-ABL, E2A-rearranged) are represented by a handful of samples.

The number of samples per group is a very important factor for selecting differentially expressed genes: in general, **the power of the tests** (i.e. the capacity to detect effectively differentially expressed genes) **increases with group sizes**.

Basics about Welch's t test

Welch's test is a variant of the classical Student test, whose goal is to test the equality between two means.

$$H_0 : m_{g,1} = m_{g,2}$$

where $m_{g,1}$ and $m_{g,2}$ represent the **respective mean expression values for a given gene g in two populations** (for example, all existing patients suffering from T-ALL versus all patients suffering from pre-B ALL). Of course, we do not dispose of measurements for all the patients suffering from these two types of ALL in the world (the population). We only dispose of two sets of samples, covering 36 (T-ALL) and 44 (pre-B ALL) patients, respectively. On the basis of these samples, we will estimate how likely it is that genes g is generally expressed at similar levels in the populations from which the samples were drawn.

The essential difference between **Student** and **Welch** is that the proper Student test relies on the assumption that the two sampled populations have the **same variance**, whereas Welch's test is designed to treat populations with **unequal variances**.

When detecting differentially expressed genes, **we cannot assume equal variance**. Indeed, a typical case would be that a gene of interest is expressed at very low level in the first group, and high level in the second group. The inter-individual fluctuations in expression values are expected to be larger when the gene is expressed at a high level than when it is poorly expressed. It is thus generally recommended to use Welch rather than Student test when analyzing microarray expression profiles.

BEWARE: Student and Welch tests **assume data normality**. Affymetrix microarray intensities are far from the normal distribution, even after log transformation. However, **t-test is robust to non-normality if there is a sufficient number of samples per group**. In the subsequent exercise, we will apply Welch test to detect genes differentially expressed between cancer types represented by ~40 samples each. We are thus in **reasonably good conditions** to run a Welch test. Nevertehless, in a next section we will also apply a non-parametric test (Wilcoxon), which does not rely on an assumption of normality.

Welch's t-test defines the t statistic by the following formula:

$$t = \frac{\bar{x_1} - \bar{x_2}}{\sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}}$$

Where:

- $\bar{x_i}$ is the sample mean,
- s_i^2 the sample variance,
- n_i the sample size.

The **t.test()** function can be used to calculate this score (and additional informations such as p.value). This function returns an **S3 object** whose slots can be listed using the **names()** function and accessed using the **$ operator** (such as with lists in R).

A first intuition

In order to get an intuition of the t statistics, let us create artificial datasets and compute the associated t value. In the following example x and y can be viewed as the expression values for gene g in two different classes of cancer.

Assuming that each group contains 4 patients, we will generate 4 random numbers following a normal distribution, to simulate the groups 1 and 2. We deliberately set the means to the same values (to fall under the null hypothesis), but we generate them with different standard deviations.

```
x <- rnorm(n=4, mean=6, s=1)
y <- rnorm(n=4, mean=6, s=2)
```

- Compute the associated t value using the **mean**, **sd** and **sqrt** functions.

View solution| Hide solution

```
# Compute the t statistics manually
nx <- length(x)
ny <- length(y)
diff <- mean(x) - mean(y)
t.obs <- diff/sqrt((sd(x)^2)/nx + (sd(y)^2)/ny)

# print the result
print(t.obs) # or t.obs or show(t.obs)
```

Solution

```
## [1] -0.4015133
```

- Now we can check that the same result is obtained using the **t.test** function implemented in R.

View solution| Hide solution

```
## Run the Welch test (this is specified by indicating that we don't expect equal variances)
simulated.welch <- t.test(x,y, var.equal=FALSE)
print(simulated.welch)
```

Solution

```
##
##  Welch Two Sample t-test
##
## data:  x and y
## t = -0.4015, df = 3.718, p-value = 0.71
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -5.673890  4.277562
## sample estimates:
## mean of x mean of y
##  5.513864  6.212029
```

```
## Note: during the practical, each student should obtain a different result, since values were generat

## Retrieve the t statistics
names(simulated.welch)
```

```
## [1] "statistic"   "parameter"   "p.value"     "conf.int"    "estimate"
## [6] "null.value"  "alternative" "method"      "data.name"
```

```
simulated.welch$statistic
```

```
##          t
## -0.4015133
```

```
## Compare the t statistics computed by the t.test() function and your manual computation
t.obs
```

```
## [1] -0.4015133
```

```
simulated.welch$statistic == t.obs
```

```
##    t
## TRUE
```

Applying Welch's t-test to the den Boer dataset

We would like to define genes that discriminate between "hyperdiploid" tumors and tumors of all the other subtypes represented by at least 10 samples in Den Boer dataset.

One possibility would be to iterate over all probesets, and to successively run the R method **t.test()** on each one. This would however be quite inefficient, and the results would not be very easy to handle, since it would be a list of objects of the class t.test.

Instead, we will use a custom function that runs Student or Welch test in parallel on all the elements of a data table.

Running t-tests on each row of a data matrix

Installing the qvalue library First we need to check if the *qvalue* library is installed (we will give more information about q-values in the next sessions).

```
### Running t-tests on each row of a data matrix
## We must first check if the q-value library from Bioconductor has
## been installed (if not, will be installed here)
if (!require("qvalue")) {
  source("http://bioconductor.org/biocLite.R")
  biocLite("qvalue")
}
```

```
## Loading required package: qvalue
```

Loading the function *t.test.multi()* The we will load a custom script written by J. van Helden (**Note:** the utilities for this course will soon be converted to an R package, in order to facilitate their installation and use).

```
## Load a custom the library for multiple t tests
url.stats4bioinfo <- "http://pedagogix-tagc.univ-mrs.fr/courses/statistics_bioinformatics"
source(file.path(url.stats4bioinfo, 'R-files/config.R'))
```

```
## [1] "Data repository http://pedagogix-tagc.univ-mrs.fr/courses/statistics_bioinformatics/data"
## [1] "R scripts source http://pedagogix-tagc.univ-mrs.fr/courses/statistics_bioinformatics/R-files
## [1] "Results will be saved to /Users/jvanheld/course_stats_bioinfo/results"
## [1] "Figures will be saved to /Users/jvanheld/course_stats_bioinfo/figures"
```

```
source(file.path(url.stats4bioinfo, 'R-files/util/util_student_test_multi.R'))
```

For the sake of curiosity, you can also have a look at the R code.

Defining sample groups We will select genes differentially expressed between one subtype of interest (for example ***hyperdiploid***) and all the other types of ALL represented by at least 10 samples. For the rest of the tutorial, we will refer to these subtypes as ***"Other"***.

```
## Classes to keep
print("Selecting cancer subtypes with >= 10 samples")
```

```
## [1] "Selecting cancer subtypes with >= 10 samples"
```

```
class.freq <- table(pheno$Sample.title)
classes.to.keep <- names(class.freq[class.freq>10])
subtype.of.interest <- "hyperdiploid"
classes.other <- setdiff(classes.to.keep, subtype.of.interest)
print(classes.to.keep)
```

```
## [1] "hyperdiploid" "pre-B ALL"    "T-ALL"        "TEL-AML1"
```

```r
## Define a Boolean vector indicating which samples belong
## to the two selected subtypes.
samples.to.keep <- pheno$Sample.title %in% classes.to.keep
sum(samples.to.keep)
```

```
## [1] 167
```

```r
## Extact a subset of expression matrix with only the two selected sets
expr.matrix.kept <- expr.matrix[,samples.to.keep]

## Export the table with the selected samples, in order to open it with TMEV
setwd(dir.output)
file <- paste(sep="", "GSE13425_Norm_",subtype.of.interest,"_vs_Other_ge10samples.txt")
write.table(expr.matrix.kept,
            file,
            col.names=NA,quote=F,sep="\t")


## Define a vector with the sample types for the two selected cancer subtype
sample.group <- as.vector(pheno[samples.to.keep, "Sample.title"])
names(sample.group) <- names(expr.matrix[samples.to.keep])
sample.group[sample.group != subtype.of.interest] = "Other"
print(sample.group)
```

```
##     GSM338666     GSM338667     GSM338668     GSM338669     GSM338670
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338671     GSM338672     GSM338673     GSM338674     GSM338675
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338676     GSM338677     GSM338678     GSM338679     GSM338680
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338681     GSM338682     GSM338683     GSM338684     GSM338685
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338686     GSM338687     GSM338688     GSM338689     GSM338690
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338691     GSM338692     GSM338693     GSM338694     GSM338695
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338696     GSM338697     GSM338698     GSM338699     GSM338700
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338701     GSM338702     GSM338703     GSM338704     GSM338705
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338706     GSM338707     GSM338708     GSM338709     GSM338710
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338711     GSM338712     GSM338713     GSM338714     GSM338715
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338716     GSM338717     GSM338718     GSM338719     GSM338720
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338721     GSM338722     GSM338723     GSM338724     GSM338725
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338726     GSM338727     GSM338728     GSM338729     GSM338730
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338731     GSM338732     GSM338733     GSM338734     GSM338735
##       "Other"       "Other"       "Other"       "Other"       "Other"
##     GSM338736     GSM338737     GSM338738     GSM338739     GSM338740
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338741      GSM338742      GSM338743      GSM338744      GSM338746
##         "Other"        "Other"        "Other"        "Other" "hyperdiploid"
##       GSM338747      GSM338748      GSM338749      GSM338750      GSM338751
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338752      GSM338753      GSM338754      GSM338755      GSM338756
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338757      GSM338758      GSM338759      GSM338760      GSM338761
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338762      GSM338763      GSM338764      GSM338765      GSM338766
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338767      GSM338768      GSM338769      GSM338770      GSM338771
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338772      GSM338773      GSM338774      GSM338775      GSM338776
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338777      GSM338778      GSM338779      GSM338780      GSM338781
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338782      GSM338783      GSM338784      GSM338785      GSM338786
## "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid" "hyperdiploid"
##       GSM338787      GSM338788      GSM338789      GSM338812      GSM338813
## "hyperdiploid" "hyperdiploid" "hyperdiploid"        "Other"        "Other"
##       GSM338814      GSM338815      GSM338816      GSM338817      GSM338818
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338819      GSM338820      GSM338821      GSM338822      GSM338823
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338824      GSM338825      GSM338826      GSM338827      GSM338828
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338829      GSM338830      GSM338831      GSM338832      GSM338833
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338834      GSM338835      GSM338836      GSM338837      GSM338838
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338839      GSM338840      GSM338841      GSM338842      GSM338843
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338844      GSM338845      GSM338846      GSM338847      GSM338848
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338849      GSM338850      GSM338851      GSM338852      GSM338853
##         "Other"        "Other"        "Other"        "Other"        "Other"
##       GSM338854      GSM338855
##         "Other"        "Other"
```

```
table(sample.group)
```

```
## sample.group
## hyperdiploid        Other
##           44          123
```

```
## Export sample groups, which will be used in other practicals
## (e.g. supervised classification)
setwd(dir.output)
file <- paste(sep="",
              "GSE13425_Norm_",subtype.of.interest,"_vs_Other_sample_groups.txt")
write.table(as.data.frame(sample.group),
            file,
            col.names=FALSE,
            row.names=TRUE,
            quote=F,sep="\t")
```

Compute Welch t-test for each gene We will now apply the **Welch test** on **each gene** of the Den Boer dataset, in order to select genes differentially expressed between the subtype of interest (***"hyperdiploid"***) and the other subtypes represented by at least 10 genes.

```
## Run the Welch test on each probesets of the DenBoer expression matrix.
## We will store the result in a table called "DEG" for "Differentially expressed genes", which will la
denboer.deg <- t.test.multi(expr.matrix.kept, sample.group, volcano.plot=FALSE)
```

```
## [1] "Fri Mar  6 15:37:06 2015 - Multiple t-test started"
## [1] "Fri Mar  6 15:37:08 2015 - Multiple t-test done"
```

```
## Inspect the result table
dim(denboer.deg)
```

```
## [1] 22283    17
```

```
names(denboer.deg)
```

```
##  [1] "mean.Other"            "mean.hyperdiploid"    "means.diff"
##  [4] "var.est.Other"         "var.est.hyperdiploid" "sd.est.Other"
##  [7] "sd.est.hyperdiploid"  "st.err.diff"           "t.obs"
## [10] "df.welch"              "P.value"              "E.value"
## [13] "sig"                   "fwer"                 "q.value"
## [16] "fdr"                   "rank"
```

```
## Select genes with a stringent threshold on E-value
eval.threshold <- 0.05
significant.probesets <- denboer.deg$E.value <= eval.threshold
table(significant.probesets) ## Count the number of significant probesets
```

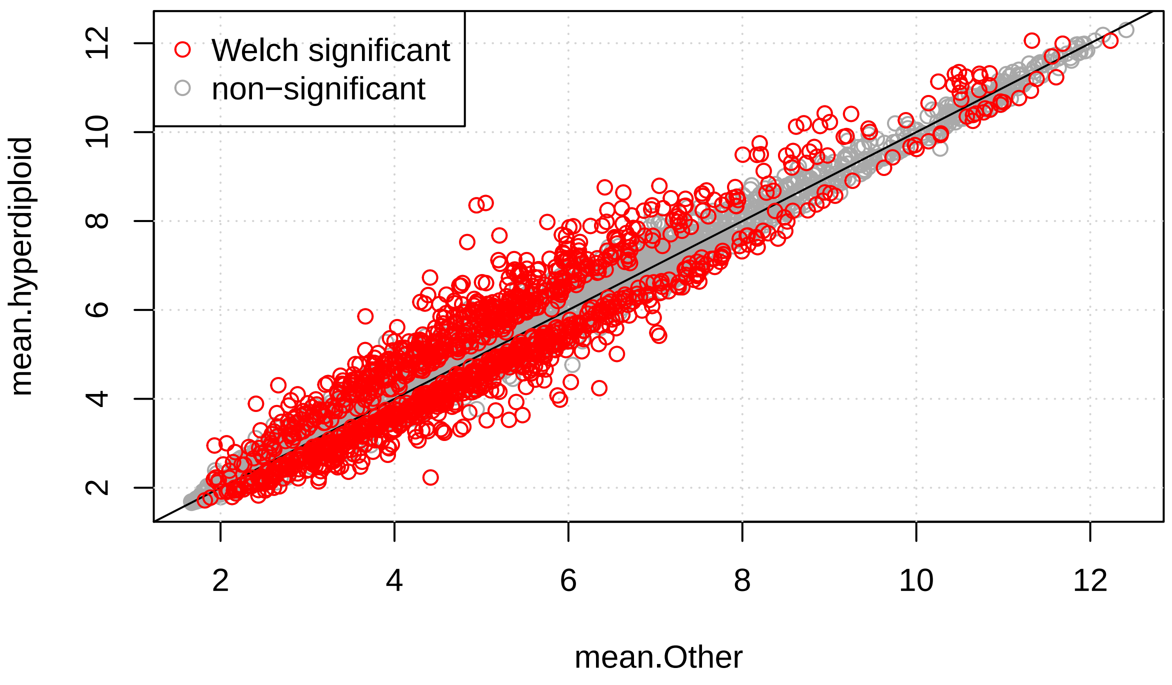
```
## significant.probesets
## FALSE  TRUE
## 20921  1362
```

Comparing sample means We will compare the mean expression value between hyperdiploids and the other selected subtypes, and highlight the significant genes.

```
## Plot the gene-wise means
plot(denboer.deg[, c("mean.Other", "mean.hyperdiploid")], col="darkgray")
grid()

abline(a=0,b=1, col="black") # Draw the diagonal line

## Highlight significant genes
lines(denboer.deg[significant.probesets,
                 c("mean.Other", "mean.hyperdiploid")],
      type='p', col="red")
legend("topleft",col=c("red", "darkgray"),
       legend=c("Welch significant","non-significant"),
       pch=1)
```



Exercise

- How do you explain that the regions covered by gray (non-significant) and red (significant) probesets overlap on the mean-mean plot ?

View solution| Hide solution

Solution The significance of a Welch (or a Student) test depends not only on the differences between the means, but also on the estimation of the standard deviation of this difference. In other terms, a same difference (or a same ratio) between two means could be either significant or not, depending on whether the two groups to be compared have a high or low variance.

Comparing the p-values of Welch and Wilcoxon tests

The apply function

The **apply** function can be used to apply a given function to a matrix or data.frame. This function has tree required arguments:

```
args(apply)
```

```
## function (X, MARGIN, FUN, ...)
## NULL
```

- X the matrix/data.frame
- MARGIN: 1 or 2 depending on wether the function has to be applied on rows or columns, respectively.

Defining a new function: return.t()

In the line below, we define a function called **return.t()**, to run the Welch test on a single probeset of the microarray table.

```
## Define a function to return the p-value of a Welch test
return.t <- function(x,y){  t.test(x[y==subtype.of.interest], x[y=="Other"], alternative="two.sided", va
```

- Use this function to compute the p-value of the Welch's t test for all probesets of expr.matrix.
- Define a similar function to compute the p-value of Wilcoxon's test to each probeset.
- Draw a plot to compare the p-values returned by the respective tests.

View solution| Hide solution

```


## Define a function to return the p-value of a Wilcoxon test
return.wilcox <- function(x,y){ wilcox.test(x[y=="Other"], x[y=="hyperdiploid"], alternative="two.sided

## Compute the pvalues and create a data frame with the results of the Welch and Wilcoxon tests
denboer.deg$welch.pval <- apply(expr.matrix.kept,1,return.t,sample.group)
denboer.deg$wilcox.pval <- apply(expr.matrix.kept,1,return.wilcox,sample.group)

## Check that all P-values are equal when computed with my
## custom Welch function, or with the return.t function
all(denboer.deg$welch.pval == denboer.deg$Pvalue)
```

Solution

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

```
## Select genes passing the p-value threshold, corrected by bonferoni's rule
pval.threshold <- eval.threshold/nrow(expr.matrix)
denboer.deg$welch.selected <- denboer.deg$welch.pval < pval.threshold
denboer.deg$wilcox.selected <- denboer.deg$wilcox.pval < pval.threshold

## Count selected genes for Welch and Wilcoxon tests, resp
sum(denboer.deg$welch.selected)
```

```
## [1] 1362
```

```
sum(denboer.deg$wilcox.selected)
```

```
## [1] 1169
```

```
## Compute a contigency table counting the number of
## consistent / different results between Welch and Wilcoxon tests
table(denboer.deg$welch.pval < pval.threshold,
      denboer.deg$wilcox.pval < pval.threshold)
```

```
##
##         FALSE  TRUE
##   FALSE 20767   154
##   TRUE    347  1015
```

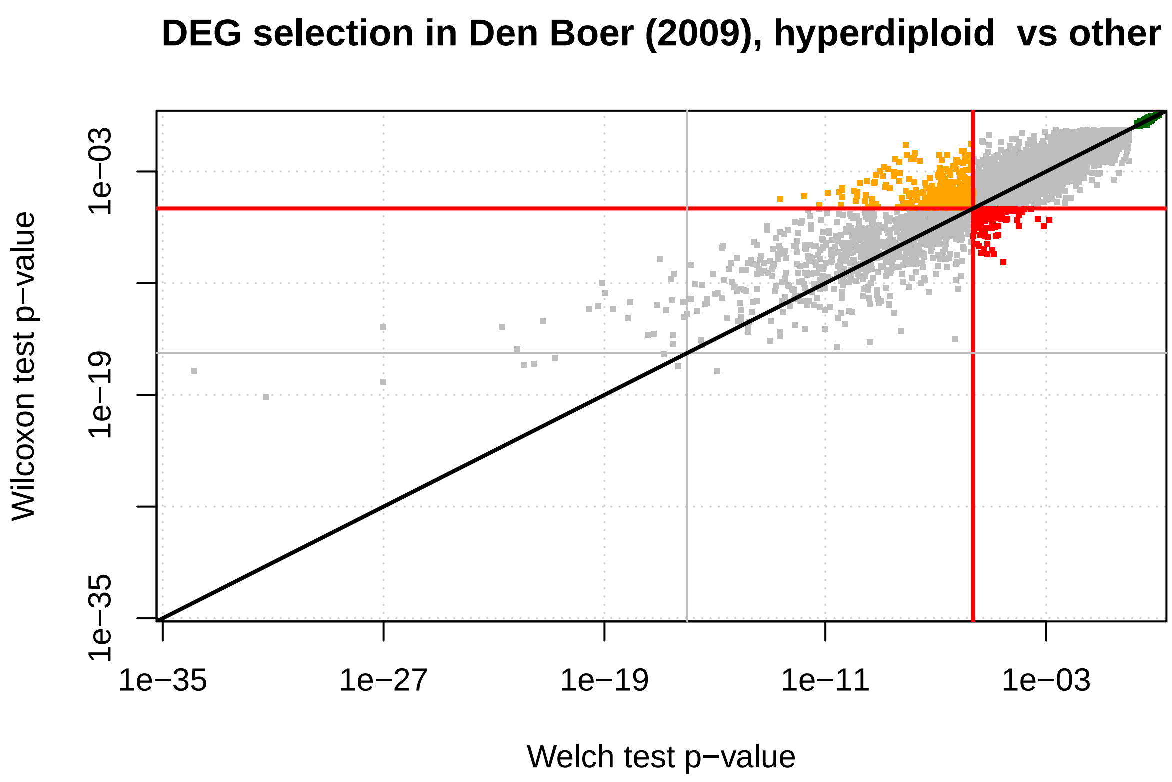
```
################################################################
## Plot the respective p-values returned by the two tests
pch <- "."
cex <- 3
min.pval <- min(denboer.deg$welch.pval, denboer.deg$wilcox.pval)
plot(denboer.deg$welch.pval,
     denboer.deg$wilcox.pval,
     log="xy", panel.first=grid(),
     xlim=c(min.pval, 1),  ylim=c(min.pval, 1),
     col="gray",
     xlab="Welch test p-value",
     ylab="Wilcoxon test p-value",
     main=paste("DEG selection in Den Boer (2009),",
                subtype.of.interest, " vs other"),
     pch=pch, cex=cex)

## Highlight in green the genes selected by both methods
welch.and.wilcox <- (denboer.deg$welch.pval < pval.threshold) &
                    (denboer.deg$wilcox.pval < pval.threshold)
points(denboer.deg[welch.and.wilcox, ], col="darkgreen", pch=pch, cex=cex)

## Highlight probesets whose selection is affected by the choice of the test
wilcox.not.welch <- denboer.deg$welch.pval >= pval.threshold & denboer.deg$wilcox.pval < pval.threshold
points(denboer.deg[wilcox.not.welch,c("welch.pval", "wilcox.pval")], col="red", pch=pch, cex=cex)

welch.not.wilcox <- denboer.deg$welch.pval < pval.threshold & denboer.deg$wilcox.pval >= pval.threshold
points(denboer.deg[welch.not.wilcox,c("welch.pval", "wilcox.pval")], col="orange", pch=pch, cex=cex)


## Draw lines to display the thresholds on the respective tests
abline(v=pval.threshold,col="red", lwd=2)
abline(h=pval.threshold,col="red", lwd=2)
abline(h=1e-16,col="gray") ## Draw the limit of floating point calculation, which is the limit for p.val
abline(v=1e-16,col="gray")
abline(a=0,b=1, col="black", lwd=2)
```



```
## Export the table with the results (Welch + Wilcoxon tests)
setwd(dir.output)
file <- paste(sep="",
              "GSE13425_Norm_",subtype.of.interest,"_vs_Other_sample_Welch.tab")
write.table(format(denboer.deg, digits=4),
            file,
            col.names=NA,
            row.names=TRUE,
            quote=F,sep="\t")
```

Drawing a volcano plot

The volcano plot is a classical representation of differential expression analysis. In this diagram, the **x axis represents the log** ratio and the **y axis the result of a statistic** expressed as $-log10(p - value)$.

Computing the log ratio

Exercise

- Calculate for each gene its average expression level in the "hyperdiploid" and "other" classes.
- Calculate the difference of the mean for each gene (log ratio).

View solution| Hide solution

```r
rowMeans.other <- apply(expr.matrix.kept[,sample.group== "Other"], 1, mean)

rowMeans.hyperdiploid <- apply(expr.matrix.kept[,sample.group== "hyperdiploid"], 1, mean)

diff <- rowMeans.other - rowMeans.hyperdiploid
range(diff)
```

Solution

```
## [1] -3.410530  2.184311
```

Volcano plot

Exercise

- Draw a volcano plot.
- Use the identify function to find the names of some interesting genes.

View solution| Hide solution

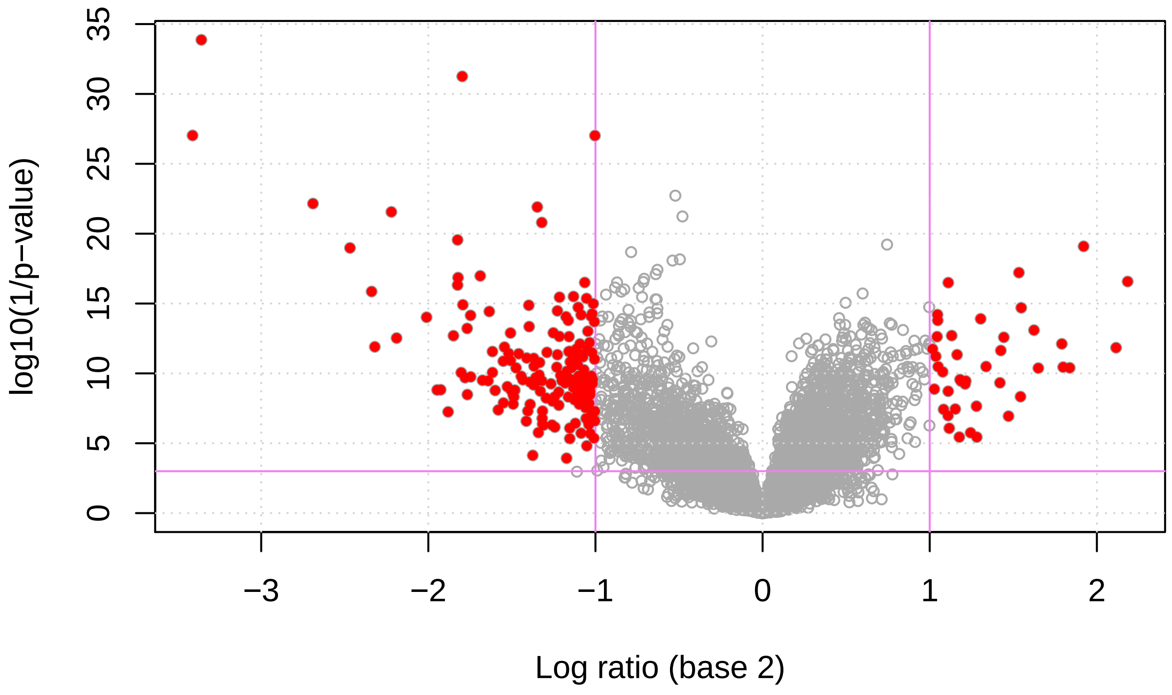
```r
## Compute the significance, i.e. -log10 of the p-value
t.res <- denboer.deg$welch.pval
mlt <- -log10(t.res)

## Draw the Volcano plot
plot(diff,mlt,pch=1,cex=0.7,
     xlab="Log ratio (base 2)",
     ylab="log10(1/p-value)",
     col="darkgray")
grid()

## Draw the selection thresholds
abline(v=c(-1,1), col="violet")
abline(h=3, col="violet")

## Select probesets based on two criteria (fold change + p-value)
retained <- (abs(diff) > 1) & (t.res < 1e-3)

## Color the selected probesets
points(diff[retained],mlt[retained],col="red",cex=0.7,pch=16)
```



Solution

Significance Analysis of Microarrays (SAM)

What is SAM ?

Probably the most popular method for differential expression analysis of microarray data is "Significance Analysis of Microarrays" (SAM). SAM will compute for each gene a score d which is close to the t statistics of the welch's test. However, it won't require any assumption about the data distribution. In order to compute the expected distribution of d under the null hypothesis SAM will performe a set of permutations on the class labels, and compute each time a simulated sets of results for the d statistics. The observed and simulated results will be used to compute FDR values. Sam is implemented in several R libraries (e.g: *siggenes*). Here we will use a more interactive program called "MultiExperiment Viewer (MeV).

Installing Mev on Linux system

If you are working on a Linux system, the commands below can be used to download and compule MeV on the Linux console.

Filtering genes on the basis of the absent/marginal/present (A/M/P) filter

The classical processing pipeline defined by Affymetrix associates a qualitative tag to each probeset, with three possible values:

- absent (A)
- marginal (M)
- present (P)

An **absent** call means that no significant signal was detected with the associated probeset. However, this “absence” might either indicate that the gene is not expressed in this particular sample, or that the gene is undetectable (irrespective of its expression level) due to some technical problem with this specific probeset.

It became a classical practice to filter out the genes called “absent” on an important fraction of the samples in a given series, by implicitly assuming that their recurrent absence reveals a technical problem rather than a biologically relevant effect (repression of the gene).

In the following exercise, we will apply an A/M/P filter to discard the genes declared absent in at least 30% of the genes.

Exercise

- Select genes giving a signal (“present” call) in at least 30% of the selected samples.

View solution| Hide solution

```
## Select a subset of the A/M/P matrix
amp.sub <- amp[,samples.to.keep]
dim(amp)
```

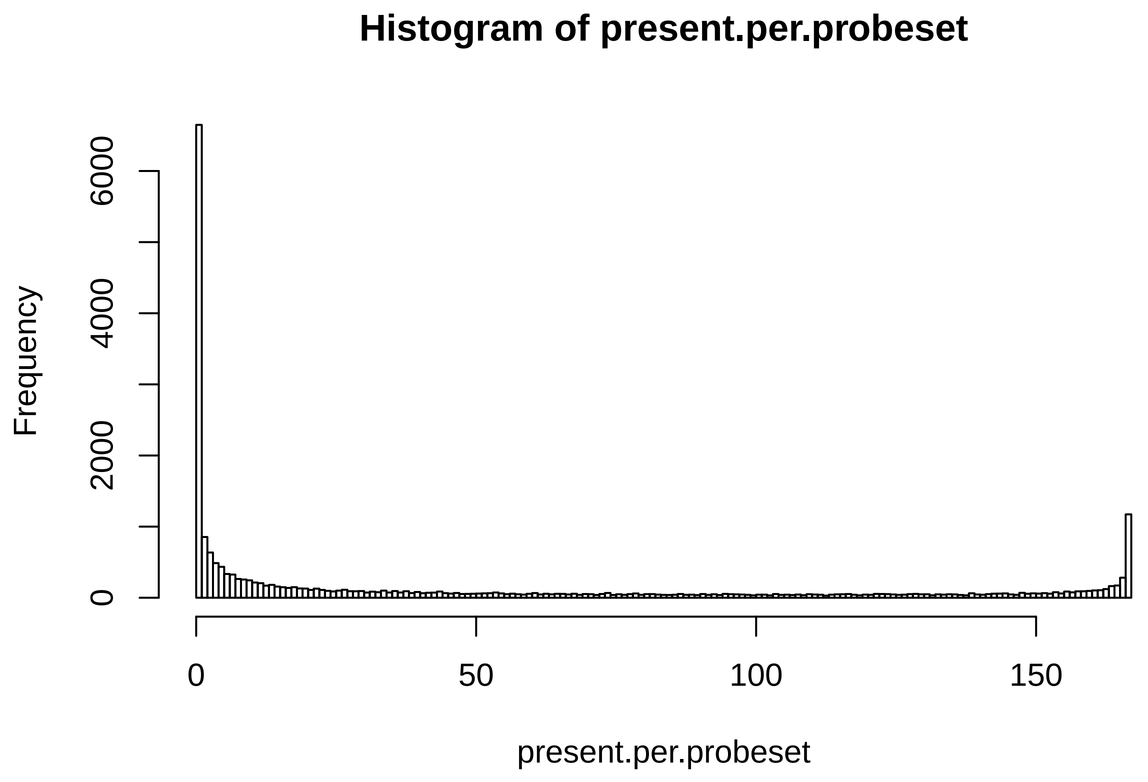
Solution

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

```
dim(amp.sub)
```

```
## [1] 22283   167
```

```
## Count the number of "Present" calls per probeset
isPresent <- amp.sub == "P"
present.per.probeset <- rowSums(isPresent)
hist(present.per.probeset, breaks=0:ncol(amp.sub))
```



```
## "Present filter":  Select probeset declared present in at least 25% of the samples
retained <- rowSums(isPresent) >=  0.25*ncol(expr.matrix.kept)
table(retained) ## Count number of retained and rejected probesets
```

```
## retained
## FALSE  TRUE
## 14013  8270
```

```
## Select a subset of the matrix with the retained probesets only
expr.matrix.kept <- expr.matrix.kept[retained, ]

## Check the number of probes (rows) and samples (columns) of the
## selected expression table
print(dim(expr.matrix.kept))
```

```
## [1] 8270  167
```

```
## Export the table with the selected samples, in order to open it with TMEV
setwd(dir.output)
file <- paste(sep="", "GSE13425_Norm_",subtype.of.interest,"_vs_Other_present30pc.txt")
write.table(expr.matrix.kept,
            file,
            col.names=NA,quote=F,sep="\t")
```

Applying SAM algorithm with MeV

Protocol

- Load the file using “File > Load data > Select file loader Tab delimited”.
- **Browse** to file “GSE13425_Norm_hyperdiploid_vs_Other.txt”, click on the **upper-leftmost expression value** and click on the **load** button.
- Select **Adjust data > Gene/Rows adjustment > median center Genes/Rows**
- Select **Analysis > Statistics > SAM**
- Set all samples from GSM338746.CEL.gz to GSM338789.CEL.gz to class B.
- Set the number of permutations to 500, select **Construct hierachical clustering** and click " OK“.
- Accept default parameters for hierarchical clustering.
- Set the **delta value to 2** and click OK.
- Select Analysis **results > SAM > Hierarchical trees > All Significant genes**.
- Select **Display > Set color scale limits** and **set lower limit to -4**, **midpoint value to 0** and **upper limit to 4**.
- Set **Display > Set Element Size** to **5x2**.
- To store the resulting file right click to select the whole gene tree and select **Save cluster**.
