Statistical Inference for High-throughput Experiments

The objective is to show in a high-throughput experiment we can not rely just on p-value to check significance.

In this study, I'll check if a specific drug it's influencing different genes. To do this, I'd used three data sets available on GitHub with a bunch of measures of various genes in case and control groups.

High-throughput experiment: have measurements from many features that are taken simultaneously. The central principle is parallelization; this means that rather than carrying out single experiments one after another, you can run several tests simultaneously and consequently decrease costs.

Dataset: GSE5859Subset (https://github.com/genomicsclass/GSE5859Subset)

Approach: check significance using t-test, compare results with Monte Carlo simulation, and apply Bonferroni correction

Package: genefilter (to run t test faster)

Key words: high-throughput experiment, t-test, Bonferroni correction, Monte Carlo simulation

Steps:

1) Load packages and data

2) Check normality

3) Apply t test

4) Generate a Monte Carlo simulation and apply t test

5) Bonferroni correction

1) Load packages and data

The data set is available on GitHub and I used two packages.

```r
#install_github("genomicsclass/GSE5859Subset")
library(GSE5859Subset)
data(GSE5859Subset) ##this loads the three tables

library(rafalib)
library(genefilter)

head(geneAnnotation, n=2)
```

```
##      PROBEID  CHR    CHRLOC SYMBOL
## 1  1007_s_at chr6  30852327   DDR1
## 30   1053_at chr7 -73645832   RFC2
```

```r
str(geneAnnotation)
```

```
## 'data.frame':    8793 obs. of  4 variables:
##  $ PROBEID: chr  "1007_s_at" "1053_at" "117_at" "121_at" ...
##  $ CHR    : chr  "chr6" "chr7" "chr1" "chr2" ...
##  $ CHRLOC : int  30852327 -73645832 161494036 -113973574 42123144 -49842638 38219068 -88932122 13534
##  $ SYMBOL : chr  "DDR1" "RFC2" "HSPA6" "PAX8" ...
```

```r
head(geneExpression, n=2)
```

```
##           GSM136508.CEL.gz GSM136530.CEL.gz GSM136517.CEL.gz
## 1007_s_at         6.543954         6.401470         6.298943
## 1053_at           7.546708         7.263547         7.201699
##           GSM136576.CEL.gz GSM136566.CEL.gz GSM136574.CEL.gz
## 1007_s_at         6.837899         6.470689         6.450220
## 1053_at           7.052761         6.980207         7.096195
##           GSM136575.CEL.gz GSM136569.CEL.gz GSM136568.CEL.gz
## 1007_s_at         6.052854         6.387026         6.640583
## 1053_at           6.983827         7.060558         7.010453
##           GSM136559.CEL.gz GSM136565.CEL.gz GSM136573.CEL.gz
## 1007_s_at         6.948474         6.778464         6.595414
## 1053_at           6.775048         7.063689         6.864693
##           GSM136523.CEL.gz GSM136509.CEL.gz GSM136727.CEL.gz
## 1007_s_at         6.255549         6.379983         6.133068
## 1053_at           7.174769         7.702533         7.280781
##           GSM136510.CEL.gz GSM136515.CEL.gz GSM136522.CEL.gz
## 1007_s_at         6.502051         6.331567         6.354293
## 1053_at           7.302209         7.456509         7.282859
##           GSM136507.CEL.gz GSM136524.CEL.gz GSM136514.CEL.gz
## 1007_s_at         6.517539         6.156754         6.037871
## 1053_at           7.689282         7.491967         7.413133
##           GSM136563.CEL.gz GSM136564.CEL.gz GSM136572.CEL.gz
## 1007_s_at         6.639091         6.393338         6.469794
## 1053_at           7.028731         6.697240         7.092346
```

```r
str(geneExpression)
```

```
##  num [1:8793, 1:24] 6.54 7.55 5.4 7.89 3.24 ...
##  - attr(*, "dimnames")=List of 2
##   ..$ : chr [1:8793] "1007_s_at" "1053_at" "117_at" "121_at" ...
##   ..$ : chr [1:24] "GSM136508.CEL.gz" "GSM136530.CEL.gz" "GSM136517.CEL.gz" "GSM136576.CEL.gz" ...
```

```r
head(sampleInfo, n=2)
```

```
##     ethnicity       date         filename group
## 107       ASN 2005-06-23 GSM136508.CEL.gz     1
## 122       ASN 2005-06-27 GSM136530.CEL.gz     1
```

```r
str(sampleInfo)
```

```
## 'data.frame':    24 obs. of  4 variables:
##  $ ethnicity: Factor w/ 3 levels "ASN","CEU","HAN": 1 1 1 1 1 1 1 1 1 1 ...
##  $ date     : Date, format: "2005-06-23" "2005-06-27" ...
##  $ filename : chr  "GSM136508.CEL.gz" "GSM136530.CEL.gz" "GSM136517.CEL.gz" "GSM136576.CEL.gz" ...
##  $ group    : num  1 1 1 1 1 1 1 1 1 1 ...
```

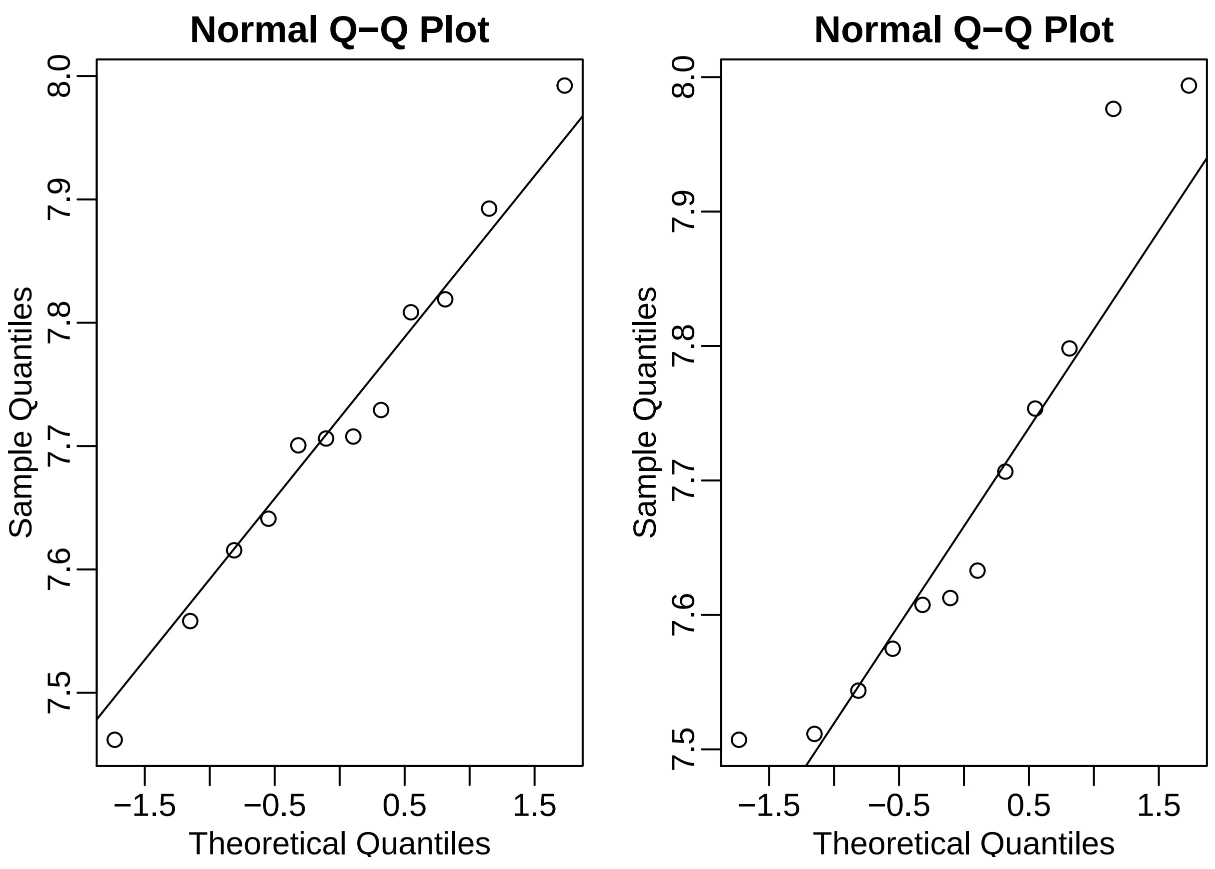
2) Check normality

```
g <- sampleInfo$group
e <- geneExpression[4,] # random choose gene 4

mypar(1,2)

# cases
qqnorm(e[g==1])
qqline(e[g==1])

# controls
qqnorm(e[g==0])
qqline(e[g==0])
```



The data has a normal distribution

3) Apply t test

```
# Apply t test for the entire data set
mytest <- function(x){
        t.test(x[g==1],x[g==0],var.equal = TRUE)$p.value
}

# apply for genes - rows (1- row, 2-column)
pvals <- apply(geneExpression,1,mytest)
```

```r
# Check number of genes (rows) are significant
sum( pvals <= 0.05)
```

```
## [1] 1383
```

We 13834 genes that are significant, but this result is not correct because in high-throughput experiments, the standard approach of using p-values is not useful. This happens because the P-value is only statistically valid when a single score is computed. And we can prove this creating a Monte Carlo simulation and apply a t test too.

4) Generate a Monte Carlo simulation and apply t test

Monte Carlo simulation relies on repeated random sampling to obtain numerical results. Their essential idea is using randomness to solve problems that might be deterministic in principle.

```r
# create a matrix using the MONTE CARLO SIMULATION with the same size of geneExpression data set
m <- nrow(geneExpression)
n <- ncol(geneExpression)

randomData <- matrix(rnorm(n*m),m,n)

# apply the test (knowing the null hypothesis is true for every single feature)
nullpvals <- apply(randomData,1,mytest)
sum( nullpvals<=0.05)
```

```
## [1] 456
```

We have 428 significant genes, but actually we have 428 false positive because we created a bunch of random normal numbers, independent, no relationship to the cases and controls using Monte Carlo simulation.

5) Apply Bonferroni correction

```r
g <- factor(sampleInfo$group)  #factor 1 and 0
results <- rowttests(geneExpression,g) # run test
pvals <- results$p.value
mean( pvals <0.05)
```

```
## [1] 0.1572842
```

```r
# How many genes have p-values smaller than 0.05 (are significant)?
sum( pvals <0.05)
```

```
## [1] 1383
```

```r
# [1] 1383

# or
table(pvals<0.05)
```

```
##
## FALSE  TRUE
##  7410  1383
```

```
# Apply the Bonferroni correction to the p-values
# sum(pvals <= alpha/m)
sum(pvals <= 0.05/8793)
```

```
## [1] 10
```

So 10 genes are called significant after Bonferroni correction.

Bonferroni is more conservative because divides the alpha per number of hypothesis being tested, but this correction is used to reduce the chances of obtaining false-positive results (type I errors).