Statistical analysis of RNAseq data

D.-L. Couturier and O. Rueda

Last modified: 22 Apr 2021

1 Section 1: Contrast matrices

1.1 Import sample information

import the information file related to each sample of the Toxoplasma Gondii Oocysts' dataset and check the class of each column

```
inf <- read.csv(file="~/courses/cruk/RNAseqWithR/202104/git/Course_Materials/data/samplesheet_corrected
                sep="\t",row.names=1)
sapply(inf,class)
```

```
##   Replicate      Status   TimePoint
##   "integer" "character" "character"
```

1.2 One 2-level factor:

define the design matrices corresponding to a model in which the gene expression is a function of *Status*

- without intercept
- with intercept

```
# model without intercept and default levels:
X1 = model.matrix(~ Status - 1, data = inf)
X1
```

```
##            StatusInfected StatusUninfected
## SRR7657878              1                0
## SRR7657881              1                0
## SRR7657880              1                0
## SRR7657874              1                0
## SRR7657882              0                1
## SRR7657872              1                0
## SRR7657877              0                1
## SRR7657876              0                1
## SRR7657879              0                1
## SRR7657883              0                1
## SRR7657873              1                0
## SRR7657875              0                1
## attr(,"assign")
## [1] 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
```

```r
# model with intercept and default reference level:
X2 = model.matrix(~ Status, data = inf)
X2
```

```
##             (Intercept) StatusUninfected
## SRR7657878            1                 0
## SRR7657881            1                 0
## SRR7657880            1                 0
## SRR7657874            1                 0
## SRR7657882            1                 1
## SRR7657872            1                 0
## SRR7657877            1                 1
## SRR7657876            1                 1
## SRR7657879            1                 1
## SRR7657883            1                 1
## SRR7657873            1                 0
## SRR7657875            1                 1
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
```

```r
# change the reference:
inf$Status = factor(inf$Status,levels=c("Uninfected","Infected"))
inf$Status
```

```
##  [1] Infected   Infected   Infected   Infected   Uninfected Infected   Uninfected
##  [8] Uninfected Uninfected Uninfected Infected   Uninfected
## Levels: Uninfected Infected
```

```r
X3 = model.matrix(~ Status, data = inf)
X3
```

```
##             (Intercept) StatusInfected
## SRR7657878            1              1
## SRR7657881            1              1
## SRR7657880            1              1
## SRR7657874            1              1
## SRR7657882            1              0
## SRR7657872            1              1
## SRR7657877            1              0
## SRR7657876            1              0
## SRR7657879            1              0
## SRR7657883            1              0
## SRR7657873            1              1
## SRR7657875            1              0
## attr(,"assign")
## [1] 0 1
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
```

```r
# matrix multiplication: let s assume a mean of 2 for infected mice and of 3 for uninfected ones
X1 %*% c(2, 3)
```

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

```
## SRR7657878    2
## SRR7657881    2
## SRR7657880    2
## SRR7657874    2
## SRR7657882    3
## SRR7657872    2
## SRR7657877    3
## SRR7657876    3
## SRR7657879    3
## SRR7657883    3
## SRR7657873    2
## SRR7657875    3
```

```
X2 %*% c(2, 1)
```

```
##            [,1]
## SRR7657878    2
## SRR7657881    2
## SRR7657880    2
## SRR7657874    2
## SRR7657882    3
## SRR7657872    2
## SRR7657877    3
## SRR7657876    3
## SRR7657879    3
## SRR7657883    3
## SRR7657873    2
## SRR7657875    3
```

```
X3 %*% c(3,-1)
```

```
##            [,1]
## SRR7657878    2
## SRR7657881    2
## SRR7657880    2
## SRR7657874    2
## SRR7657882    3
## SRR7657872    2
## SRR7657877    3
## SRR7657876    3
## SRR7657879    3
## SRR7657883    3
## SRR7657873    2
## SRR7657875    3
```

1.3 One 3-level factor:

create the *modified/fake* three-level status variable *status3* and define the design matrices corresponding to a model in which the gene expression is a function of this three-level status variable

- without intercept
- with intercept

```
#
inf$Status3 = as.character(inf$Status)
inf$Status3[c(1,2,10,12)] = "Half-infected"
inf
```

```r
# model without intercept and default levels:
X1 = model.matrix(~ Status3 - 1, data = inf)
X1
```

```
##            Status3Half-infected Status3Infected Status3Uninfected
## SRR7657878                    1                0                  0
## SRR7657881                    1                0                  0
## SRR7657880                    0                1                  0
## SRR7657874                    0                1                  0
## SRR7657882                    0                0                  1
## SRR7657872                    0                1                  0
## SRR7657877                    0                0                  1
## SRR7657876                    0                0                  1
## SRR7657879                    0                0                  1
## SRR7657883                    1                0                  0
## SRR7657873                    0                1                  0
## SRR7657875                    1                0                  0
## attr(,"assign")
## [1] 1 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Status3
## [1] "contr.treatment"
```

```r
# model with intercept and default levels
X2 = model.matrix(~ Status3, data = inf)
X2
```

```
##            (Intercept) Status3Infected Status3Uninfected
## SRR7657878           1                0                  0
## SRR7657881           1                0                  0
## SRR7657880           1                1                  0
## SRR7657874           1                1                  0
## SRR7657882           1                0                  1
## SRR7657872           1                1                  0
## SRR7657877           1                0                  1
## SRR7657876           1                0                  1
## SRR7657879           1                0                  1
## SRR7657883           1                0                  0
## SRR7657873           1                1                  0
## SRR7657875           1                0                  0
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Status3
## [1] "contr.treatment"
```

```r
# model with intercept and self-defined levels
inf$Status3 = factor(inf$Status3,levels=c("Uninfected","Half-infected","Infected"))
inf$Status3
```

```
##  [1] Half-infected Half-infected Infected      Infected      Uninfected    Infected
##  [7] Uninfected    Uninfected    Uninfected    Half-infected Infected      Half-infected
## Levels: Uninfected Half-infected Infected
```

```r
X3 = model.matrix(~ Status3, data = inf)
X3
```

```
##             (Intercept) Status3Half-infected Status3Infected
## SRR7657878            1                    1                0
## SRR7657881            1                    1                0
## SRR7657880            1                    0                1
## SRR7657874            1                    0                1
## SRR7657882            1                    0                0
## SRR7657872            1                    0                1
## SRR7657877            1                    0                0
## SRR7657876            1                    0                0
## SRR7657879            1                    0                0
## SRR7657883            1                    1                0
## SRR7657873            1                    0                1
## SRR7657875            1                    1                0
## attr(,"assign")
## [1] 0 1 1
## attr(,"contrasts")
## attr(,"contrasts")$Status3
## [1] "contr.treatment"
```

```
# matrix multiplication: let s assume a mean of
# > 2 for Uninfected
# > 3 for Half-infected
# > 4 for Infected
# ! CHALLENGE !
# ! FIND the values of the BETA vector corresponding to X1, X2 and X3 !
```

1.4 Two 2-level factors:

define the design matrices corresponding to a model in which the gene expression is a function of the two-level factors status and time

- without interaction
- with interaction

```
# design matrix without interaction
X1 = model.matrix(~ Status + TimePoint, data=inf)
X1
```

```
##             (Intercept) StatusInfected TimePointd33
## SRR7657878            1              1            0
## SRR7657881            1              1            0
## SRR7657880            1              1            0
## SRR7657874            1              1            1
## SRR7657882            1              0            1
## SRR7657872            1              1            1
## SRR7657877            1              0            0
## SRR7657876            1              0            0
## SRR7657879            1              0            0
## SRR7657883            1              0            1
## SRR7657873            1              1            1
## SRR7657875            1              0            1
## attr(,"assign")
## [1] 0 1 2
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
```

```
##
## attr(,"contrasts")$TimePoint
## [1] "contr.treatment"
```

```
# design matrix with interaction
X2 = model.matrix(~ Status * TimePoint, data=inf)
X2
```

```
##            (Intercept) StatusInfected TimePointd33 StatusInfected:TimePointd33
## SRR7657878           1              1            0                           0
## SRR7657881           1              1            0                           0
## SRR7657880           1              1            0                           0
## SRR7657874           1              1            1                           1
## SRR7657882           1              0            1                           0
## SRR7657872           1              1            1                           1
## SRR7657877           1              0            0                           0
## SRR7657876           1              0            0                           0
## SRR7657879           1              0            0                           0
## SRR7657883           1              0            1                           0
## SRR7657873           1              1            1                           1
## SRR7657875           1              0            1                           0
## attr(,"assign")
## [1] 0 1 2 3
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
##
## attr(,"contrasts")$TimePoint
## [1] "contr.treatment"
```

```
model.matrix(~ Status + TimePoint + Status:TimePoint, data=inf)
```

```
##            (Intercept) StatusInfected TimePointd33 StatusInfected:TimePointd33
## SRR7657878           1              1            0                           0
## SRR7657881           1              1            0                           0
## SRR7657880           1              1            0                           0
## SRR7657874           1              1            1                           1
## SRR7657882           1              0            1                           0
## SRR7657872           1              1            1                           1
## SRR7657877           1              0            0                           0
## SRR7657876           1              0            0                           0
## SRR7657879           1              0            0                           0
## SRR7657883           1              0            1                           0
## SRR7657873           1              1            1                           1
## SRR7657875           1              0            1                           0
## attr(,"assign")
## [1] 0 1 2 3
## attr(,"contrasts")
## attr(,"contrasts")$Status
## [1] "contr.treatment"
##
## attr(,"contrasts")$TimePoint
## [1] "contr.treatment"
```

```
# matrix multiplication: let s assume a mean of
# > 2 for Uninfected at 11 dpi
```

```
# > 3 for Infected at 11 dpi
# > 4 for Uninfected at 33 dpi
# > 6 for Infected at 33 dpi

# ! CHALLENGE !
# ! FIND the values of the BETA vector corresponding to X1 and X2!
```

2 Section 2: DESeq2

2.1 Introduction slide

Let's generate

- *cnts*, a toy matrix of counts of 1000 genes for 20 samples,
- *cond*, a vector indicating to which condition each sample belongs (1 for treatment 1, 2 for treatment 2),

```
set.seed(777)
cnts <- matrix(rnbinom(n=20000, mu=100, size=1/.25), ncol=20)
cond <- factor(rep(1:2, each=10))
```

Let's

- combine the count matrix, the sample information and the assumed model in an object of class *DESeqDataSet*,
- perform the DE analysis via the function *DESeq*
- print the results

```
library(DESeq2)
dds <- DESeqDataSetFromMatrix(cnts, DataFrame(cond), ~ cond)
dds <- DESeq(dds)
results(dds)
```

```
## log2 fold change (MLE): cond 2 vs 1
## Wald test p-value: cond 2 vs 1
## DataFrame with 1000 rows and 6 columns
##       baseMean log2FoldChange     lfcSE      stat    pvalue      padj
##      <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
## 1      97.3140      -0.682067  0.344525 -1.979730 0.0477339  0.745842
## 2     109.9860      -0.228819  0.450720 -0.507676 0.6116808  0.944354
## 3      98.8111       0.104291  0.462113  0.225683 0.8214483  0.978382
## 4     103.2615       0.306400  0.297682  1.029284 0.3033460  0.944354
## 5      97.9406       0.316338  0.357242  0.885501 0.3758864  0.944354
## ...        ...            ...       ...       ...       ...       ...
## 996    86.8057      0.0467703  0.287042  0.162939 0.8705668  0.980044
## 997   101.4437     -0.2070806  0.339886 -0.609264 0.5423495  0.944354
## 998    78.1356     -0.6372790  0.369515 -1.724637 0.0845930  0.824310
## 999    89.2920      0.7554725  0.306192  2.467314 0.0136131  0.614613
## 1000  103.5569     -0.0728875  0.348655 -0.209053 0.8344065  0.978382
```

```
results(dds,name="Intercept")
```

```
## log2 fold change (MLE): Intercept
## Wald test p-value: Intercept
## DataFrame with 1000 rows and 6 columns
##       baseMean log2FoldChange     lfcSE      stat       pvalue         padj
```

```
##      <numeric>      <numeric> <numeric> <numeric>    <numeric>    <numeric>
## 1      97.3140        6.90565  0.242562   28.4697 2.78073e-178 4.84448e-178
## 2     109.9860        6.89102  0.318468   21.6381 7.87448e-104 8.03519e-104
## 3      98.8111        6.57355  0.326862   20.1111  5.90379e-90  5.93346e-90
## 4     103.2615        6.52875  0.210965   30.9472 2.77420e-210 9.94335e-210
## 5      97.9406        6.44716  0.253037   25.4792 3.35581e-143 3.84399e-143
## ...        ...            ...       ...       ...          ...          ...
## 996    86.8057        6.41605  0.203037   31.6003 3.65281e-219 1.63803e-218
## 997   101.4437        6.76442  0.240023   28.1823 9.62980e-175 1.58125e-174
## 998    78.1356        6.57184  0.260146   25.2621 8.34043e-141 9.41358e-141
## 999    89.2920        6.05380  0.217898   27.7827 7.02445e-170 1.06593e-169
## 1000  103.5569        6.73029  0.246421   27.3122 3.03850e-164 4.29167e-164
```

```
results(dds,name="cond_2_vs_1")
```

```
## log2 fold change (MLE): cond 2 vs 1
## Wald test p-value: cond 2 vs 1
## DataFrame with 1000 rows and 6 columns
##       baseMean log2FoldChange     lfcSE      stat    pvalue      padj
##      <numeric>      <numeric> <numeric> <numeric> <numeric> <numeric>
## 1      97.3140      -0.682067  0.344525 -1.979730 0.0477339  0.745842
## 2     109.9860      -0.228819  0.450720 -0.507676 0.6116808  0.944354
## 3      98.8111       0.104291  0.462113  0.225683 0.8214483  0.978382
## 4     103.2615       0.306400  0.297682  1.029284 0.3033460  0.944354
## 5      97.9406       0.316338  0.357242  0.885501 0.3758864  0.944354
## ...        ...            ...       ...       ...       ...       ...
## 996    86.8057      0.0467703  0.287042  0.162939 0.8705668  0.980044
## 997   101.4437     -0.2070806  0.339886 -0.609264 0.5423495  0.944354
## 998    78.1356     -0.6372790  0.369515 -1.724637 0.0845930  0.824310
## 999    89.2920      0.7554725  0.306192  2.467314 0.0136131  0.614613
## 1000  103.5569     -0.0728875  0.348655 -0.209053 0.8344065  0.978382
```

2.2 Section 2 slides dedicated to dispersion

Let's print the relevant information to deduce the estimated NB distribution assumed for each gene and condition:

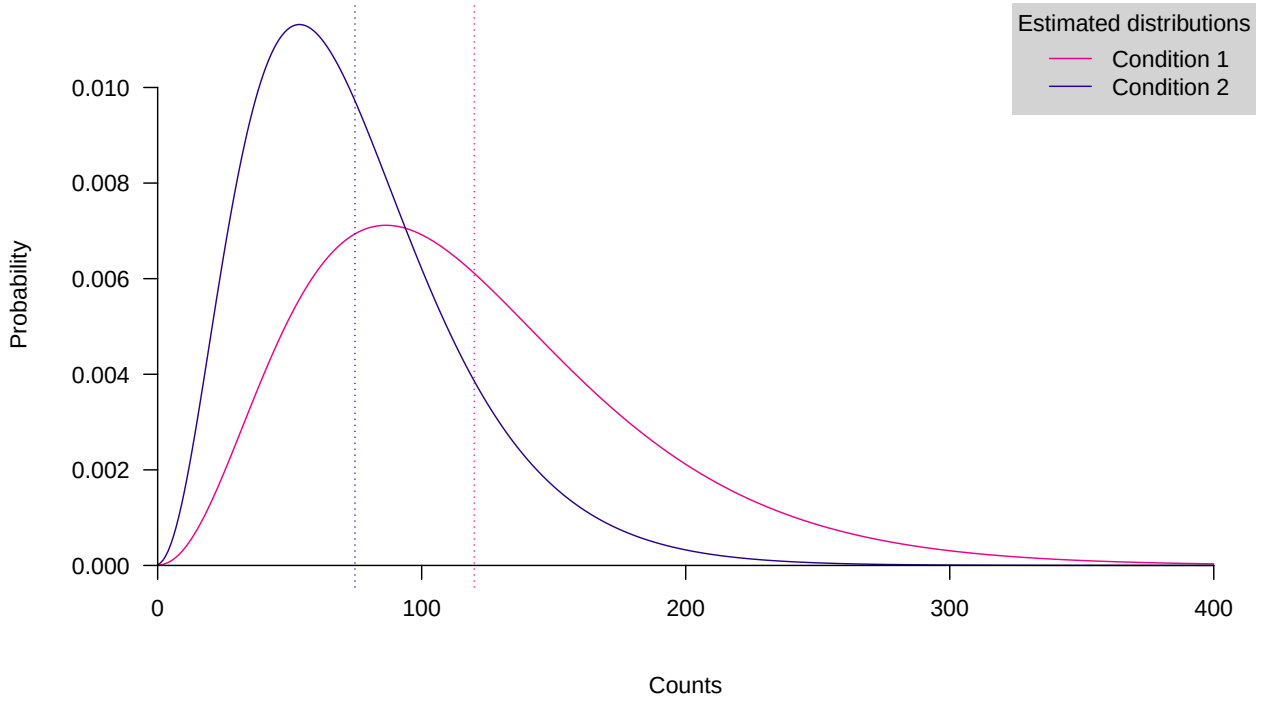
```
mcols(dds)[,c("Intercept","cond_2_vs_1","dispGeneEst","dispFit","dispersion")]
```

```
## DataFrame with 1000 rows and 5 columns
##      Intercept cond_2_vs_1 dispGeneEst   dispFit dispersion
##      <numeric>   <numeric>   <numeric> <numeric>  <numeric>
## 1      6.90565   -0.682067    0.294082  0.234624   0.274708
## 2      6.89102   -0.228819    0.479231  0.230525   0.479231
## 3      6.57355    0.104291    0.503276  0.235959   0.503276
## 4      6.52875    0.306400    0.189799  0.235749   0.203479
## 5      6.44716    0.316338    0.327393  0.235236   0.296668
## ...        ...         ...         ...       ...        ...
## 996    6.41605   0.0467703    0.167776  0.230673   0.186870
## 997    6.76442  -0.2070806    0.282533  0.236789   0.268003
## 998    6.57184  -0.6372790    0.363912  0.222687   0.315105
## 999    6.05380   0.7554725    0.206644  0.229562   0.213730
## 1000   6.73029  -0.0728875    0.304930  0.235483   0.282745
```

Let's reproduce the plot showing the fitted probability mass functions per condition for gene 1:

```
axe.x = seq(0,400)
f.x1  = dnbinom(axe.x, mu=2^6.90565, size=1/0.274708)
f.x2  = dnbinom(axe.x, mu=2^(6.90565-0.682067), size=1/0.274708)

par(mfrow=c(1,1),mar=c(4,4,0,0))
ylimw = max(c(f.x1,f.x2))
plot(1,1,ylim=c(0,ylimw),xlim=c(0,max(axe.x)),pch="",xlab="Counts",ylab="Probability",
     axes=FALSE)
lines(axe.x,f.x1,col=.cruk$col[1])
lines(axe.x,f.x2,col=.cruk$col[3])
axis(1,pos=0)
axis(2,las=2,pos=0)
legend("topright",bg="light gray",lty=1,col=.cruk$col[c(1,3)],
       legend=c("Condition 1","Condition 2"),title="Estimated distributions",box.lwd=NA)
abline(v=2^6.90565,col=.cruk$col[1],lty=3)
abline(v=2^(6.90565-0.682067),col=.cruk$col[3],lty=3)
```



3 Section 3: Large Scale Hypothesis testing: FDR

When we are doing thousands of tests for differential expression, the overall significance level of a test is very difficult to control. Let's see why: First, we simulate 40,000 genes not differentially expressed (with a mean of zero). We assume that we have 10 replicates of this experiment:

```
N <- 40000
R <- 10
X <- matrix(rnorm(N* R, 0, 1), nrow=N)
```

Now we assume that we run a t-test under the null hypothesis that the mean is zero for each of these genes, that is each row in the matrix:

```
t.test(X[1,])$p.value
```

```
## [1] 0.5723298
```

```
pvals <- apply(X, 1, function(y) t.test(y)$p.value)
```

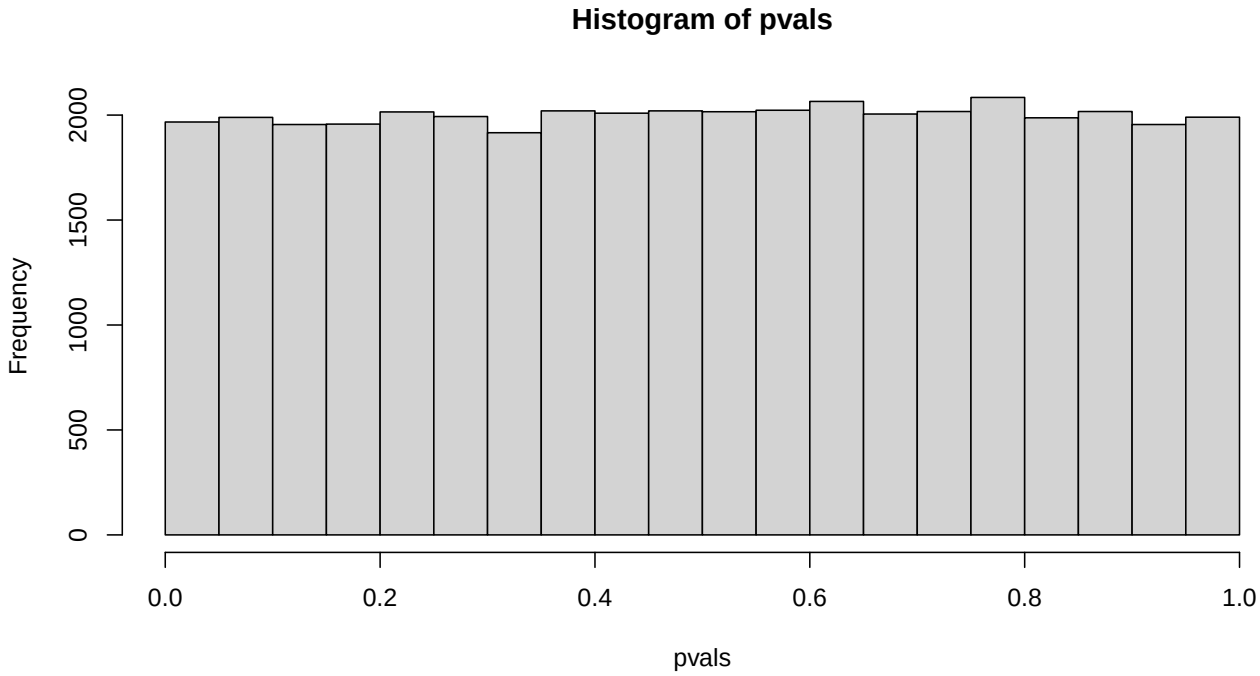
Because we have generated this data with mean zero, we know that none of these genes are differentially expressed, so we would like to be able to not reject any of the hypothesis. However, if you choose a significance level of 0.05 we get

```
sum(pvals<0.05)
```

```
## [1] 1967
```

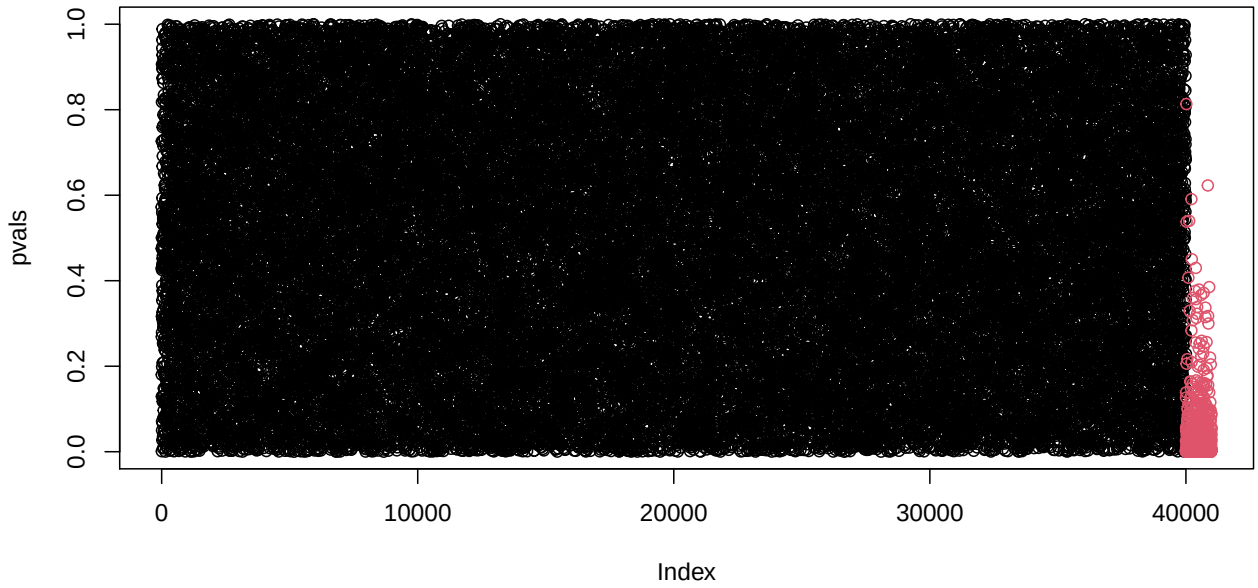
Too many rejections!!! In fact, if we look at the distributions of the p-values obtained we get:

```
hist(pvals)
```

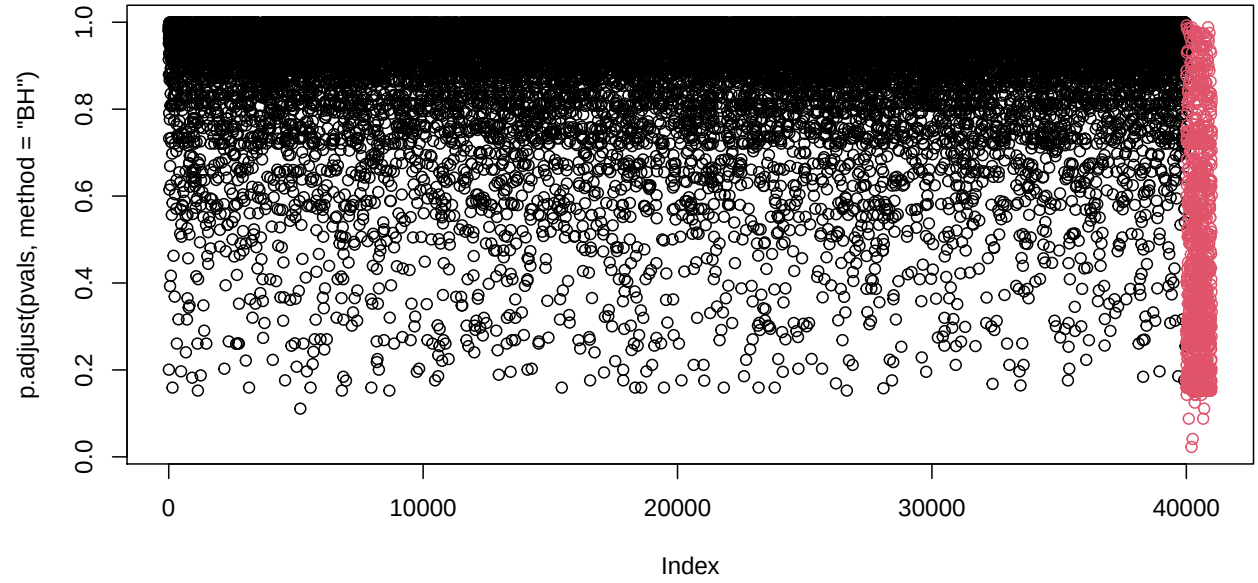


That is, if the null hypothesis is true, the p-values will follow a uniform distribution. This is the key to all methods that aim to control the proportion of false positives amongs the genes that we call differentially expressed. Let's add 1000 genes to our set that are really differentially expressed (mean of 1):

```
df <- 1000
Y <- matrix(rnorm(df* R, 1, 1), nrow=df)
Z <- rbind(X, Y)
pvals <- apply(Z, 1, function(y) t.test(y)$p.value)
#
plot(pvals,col=rep(1:2,c(40000,1000)))
```




```
plot(p.adjust(pvals, method="BH"),col=rep(1:2,c(40000,1000)))
```

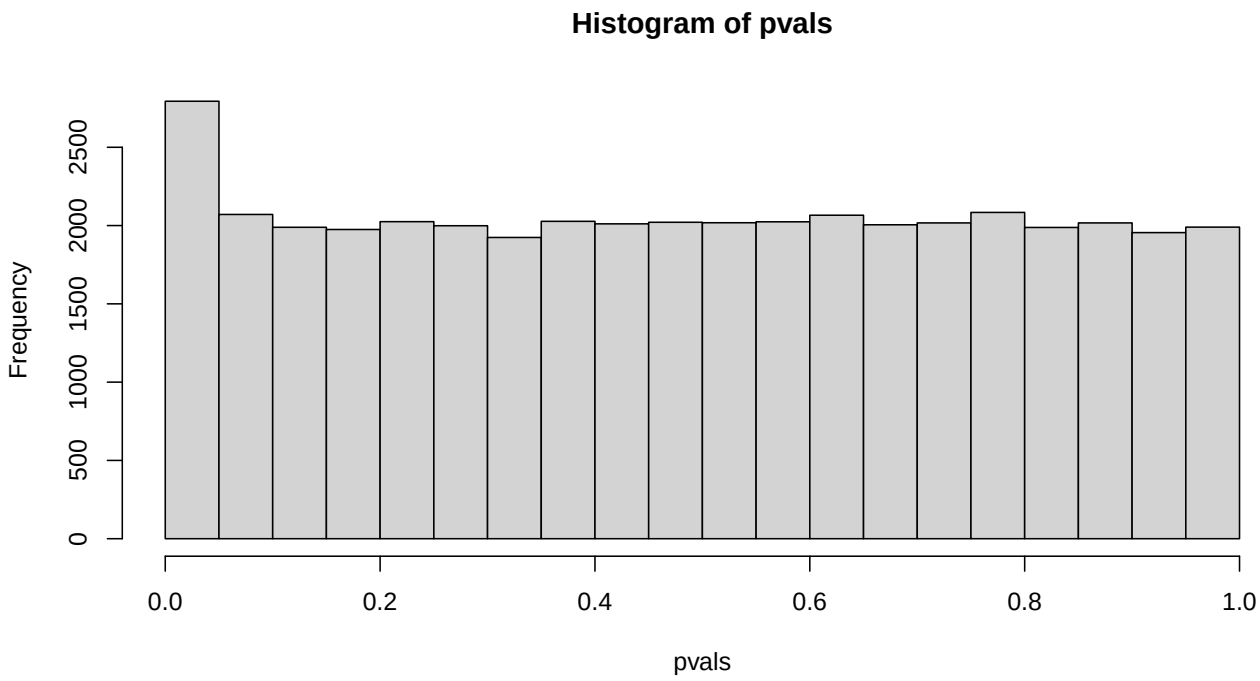



```
#
tapply(p.adjust(pvals, method="BH")<0.05,rep(1:2,c(40000,1000)),mean)
```

```
##     1     2
## 0.000 0.002
```

Let's look at the distribution of p-values now:

```
hist(pvals)
```



What would be the number of false positives now? How many would we expect if we reject p-values samller than our significance level, 0.05?

```
exp.sig<- (nrow(Z))*0.05
obs.sig <- sum(pvals<0.05)
FDR <- exp.sig / obs.sig
FDR
```

```
## [1] 0.7337151
```

We can compare this with the Benjamini-Hochberg method:

```
pvals.adj <- p.adjust(pvals, method="BH")
plot(pvals, pvals.adj)
abline(v=0.05, col=2)
```