FWER_FDR_pFDR

`!IMPORTANT DISCLAIMER!` -> the current exercise take inspiration from the Harvard course on EdX : `"Statistical Inference and Modeling for High-throughput Experiments"` and the book `"Data analysis for the Life Sciences", by Rafael A.Irizarry and Michael I. Love`

FWER, FDR, pFDR - GSE5859Subset

This is a mini-project which aim is to compute the `FWER`, the `FDR` and the `pFDR` for a data set of gene expressions obtained with a high-throughput analysis.

The second goal is to show the differences between the 3 different metrics and what are the pros and cons to use them.

INTRO

DESCRIPTION of the data set

This data set was originally created to compare the gene expressions across ethnic groups. 24 patients have been taken for this experiment.

This data set is divided in 3 different parts:

sampleInfo

"24 obs." x "4 columns". Info about the patients

- *ETHNICITY:* ethnicity of the patient from which the blood sample has been taken. -> (Factor with 3 levels "ASN", "CEU", "HAN")
- *DATE:* date when the sample has been taken. -> (Date format)

- FILENAME: is the id that connects this data set to the "geneExpression" one. -> (Character type)
- GROUP: indicates if the patient belongs to the "control" or to the "treatment" group, so it has 2 values: "1" or "0". -> (Numeric type)

	ethnicity	date	filename	group
107	ASN	2005-06-23	GSM136508.CEL.gz	1
122	ASN	2005-06-27	GSM136530.CEL.gz	1
113	ASN	2005-06-27	GSM136517.CEL.gz	1
163	ASN	2005-10-28	GSM136576.CEL.gz	1
153	ASN	2005-10-07	GSM136566.CEL.gz	1
161	ASN	2005-10-07	GSM136574.CEL.gz	1

geneExpression

"8793 obs." x "24 columns". Is the data set of the RNA expression levels of the different genes among different chromosomes. The sample is a blood sample.

- *COLUMNS:* here the columns represent the patients from which the sample was taken.
- ROWS: it indicates the **mRNA abundance** detected (expressed in the log_2 form) for each probe set (row) in each sample (column). For instance, the probe set "117_at" (for gene HSPA6, reference to the geneAnnotation data det) has a log_2 expression level of 5.4 in the sample "GSM136508.CEL.gz" (reference to the sampleInfo data set).

```
> geneExpression["117_at", "GSM136508.CEL.gz"]
[1] 5.402622
```

geneAnnotation

"8793 obs." x "4 columns". Is the data set about the connection between gene and chromosome.

- *PROBEID*: this is the probe identifier, it allow us to connect information from this table to the ones stored in the "geneExpression".
- *CHR:* indicates the chromosome on which the gene or probe is located.

- *CHRLOC:* This represents the chromosomal location of the gene or probe. The number indicates the base pair position along the chromosome. A negative value typically indicates that the gene is located on the **reverse (minus) strand**, while a positive value indicates the **forward (plus) strand**
- *SYMBOL:* the corresponding gene symbol

	PROBEID	CHR	CHRLOC	SYMBOL
1	1007_s_at	chr6	30852327	DDR1
30	1053_at	chr7	-73645832	RFC2
31	117_at	chr1	161494036	HSPA6
32	121_at	chr2	-113973574	PAX8
33	1255_g_at	chr6	42123144	GUCA1A
34	1294_at	chr3	-49842638	UBA7

DOWNLOAD of the data set

```
library(devtools)
library(rafalib)
#download if needed
## install_github("genomicsclass/GSE5859Subset")
## BiocManager::install(c("genefilter", "qvalue"))

library(GSE5859Subset)
data(GSE5859Subset)
```

```
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 ...
```

```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 -889321
 $ SYMBOL : chr  "DDR1" "RFC2" "HSPA6" "PAX8" ...
```

```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
```

T - TEST

We will perform multiple row-wise t-tests among the "geneExpression" rows but before that we need to say something about the theory behind the tests we are going to compute.

Assumptions

To ensure the validity of p-values computed by a t-test, certain assumptions must be met. These assumptions apply to each row-wise t-test when comparing gene expression levels across two experimental conditions (e.g., control vs treatment).

- **NORMALITY:** The data should be realizations of a random variable following a Normal (Gaussian) distribution. In other words, both the control and treatment groups should contain observations drawn from normally distributed populations. In our case, since the goal is to study FWER, FDR, and pFDR under multiple testing, we will assume **a priori** that the data in each group are normally distributed to simplify the analysis.

  In real-world applications, however, this assumption may not always hold—especially with small sample sizes. One approach to check normality is to use a **Shapiro-Wilk test**, but doing this for many small groups (e.g., 12 patients) can be time-consuming and unreliable due to limited statistical power. An alternative is to **increase the sample size**, allowing the **Central Limit Theorem (CLT)** to ensure approximate normality of the sampling distribution of the mean.

- **INDEPENDENCE:** We assume that observations within each group (e.g., control or treatment) per gene are independent, based on the study design. We will address the question of dependence between genes/tests when discussing multiple testing correction.
- **HOMOSCEDASTICTY:** To validly apply a standard two-sample t-test, we assume that the variances of the two groups (control and treatment) for each gene are equal. In our case, we will assume equal variances as a simplifying assumption. In real-world scenarios, we could apply Levene's test for each gene to assess the equality of variances. However, running a Levene test per gene and switching test types accordingly is often impractical and unstable, particularly with small sample sizes. A common alternative is to use Welch's t-test by default, as it does not assume equal variances. While this simplifies the problem, it may still not capture the full complexity of high-throughput data.

In high-throughput settings, these assumptions may be partially violated or hard to verify for each gene individually. Tools like limma, DESeq2, or edgeR are designed to handle these complexities more robustly.

Running the row-wise t-tests

The first thing we need to do is specify which individuals in the "geneExpression" data set belong to the control and the treatment group -> this is done with the factor "g".

We run the row-wise t-tests and stored the p-values compute in ascending order.

```
library(genefilter)
g <- factor(sampleInfo$group) # to indicate which observation belong to which group
pvals <- sort(rowttests(geneExpression, g)$p.value)
head(pvals)
```

```
[1] 8.453679e-23 5.429425e-20 1.044354e-19 3.691745e-19 1.224535e-16
[6] 5.053589e-12
```

FWER

We are now going to describe what is the FWER and how to compute it.

- **DEFINITION:** is the probability of obtaining at least 1 type I error in the multiple testing framework.

- **FORMULA:**

$$FWER = 1 - (1 - \alpha)^m,$$

where m is the number of tests

- **ALPHA:** is the significance threshold set a priori (commonly 0.05).

Assumption

This formula assumes that all "m" tests are `independent`. When tests are dependent, the actual FWER may differ, and more conservative corrections may be needed.

Bonferroni or Sidak to control the FWER?

Using a significance threshold of 0.05 in a multiple testing context—such as high-throughput analysis—can lead to many false positives. As the number of tests m increases, so does the risk of obtaining at least one false discovery, as indicated by the FWER formula. One way to control this is to use a corrected alpha to maintain the FWER at a desired level (e.g., 0.05). In our case, we will use two such methods: Bonferroni and Sidak

SIDAK

The Sidak method adjusts the per-test significance level so that the FWER remains at or below a desired level. When we use the SIDAK method we implicitly assume the independence between the tests.

```
alpha <- 0.05  # desired FWER
m <- nrow(geneExpression)  # total number of tests

# Compute Sidak-corrected per-test alpha
sidak_alpha <- 1 - (1 - alpha)^(1/m)

paste0("Sidak per-test alpha = ", signif(sidak_alpha, 4))
```

```
[1] "Sidak per-test alpha = 5.833e-06"
```

BONFERRONI

Unlike Sidak, Bonferroni does not require independence among the t-tests. As the Sidak method, Bonferroni adjusts the per-test significance level so that the FWER remains at or below a desired level

```
alpha <- 0.05  # desired FWER
m <- nrow(geneExpression)  # total number of tests

# Compute Bonferroni-corrected per-test alpha
bonf_alpha <- alpha / m

paste0("Bonferroni per-test alpha = ", signif(bonf_alpha, 4))
```

```
[1] "Bonferroni per-test alpha = 5.686e-06"
```

Bonferroni vs Sidak

Let's compare the 2 results

```
bonf_alpha < sidak_alpha
```

```
[1] TRUE
```

In general, the corrected alpha computed using the Bonferroni method is more conservative than the one computed with the Sidak method, in other words, using Bonferroni will result in a lower number of significant findings in a multiple testing context, especially when the number of tests is large.

Why we do not use FWER in practice?

If we use the Bonferroni-corrected alpha as the new threshold for our tests, we will often find a surprisingly small number of significant results. This is because lowering alpha (to control the Family-Wise Error Rate) drastically reduces the chance of false positives (Type I errors), but also increases the chance of false negatives (Type II errors).

In other words, lowering alpha this way reduces the power of the test — the ability to detect true effects — which can be problematic in high-dimensional settings like genomics or transcriptomics.

```r
sum(pvals < bonf_alpha)
```

```
[1] 10
```

FDR

As a more powerful solution to the FWER is the FDR, a metric defined as

$$\text{FDR} = \mathbb{E}\left[\frac{FP}{FP+TP}\right] = \mathbb{E}\left[\frac{V}{R}\right]$$

- **FP:** tests that appear significant but are actually false discoveries — i.e., the null hypothesis is true, but rejected.
- **TP:** tests that are significant and correspond to a real effect — i.e., the null hypothesis is false, and correctly rejected.
- **FDP:** the fraction "FP / FP + TP" is called the "observed False Discovery Proportion", namely "FDP"
- **FDR:** the expected value of the observed False Discovery Proportion (FDP).

BH vs STOREY

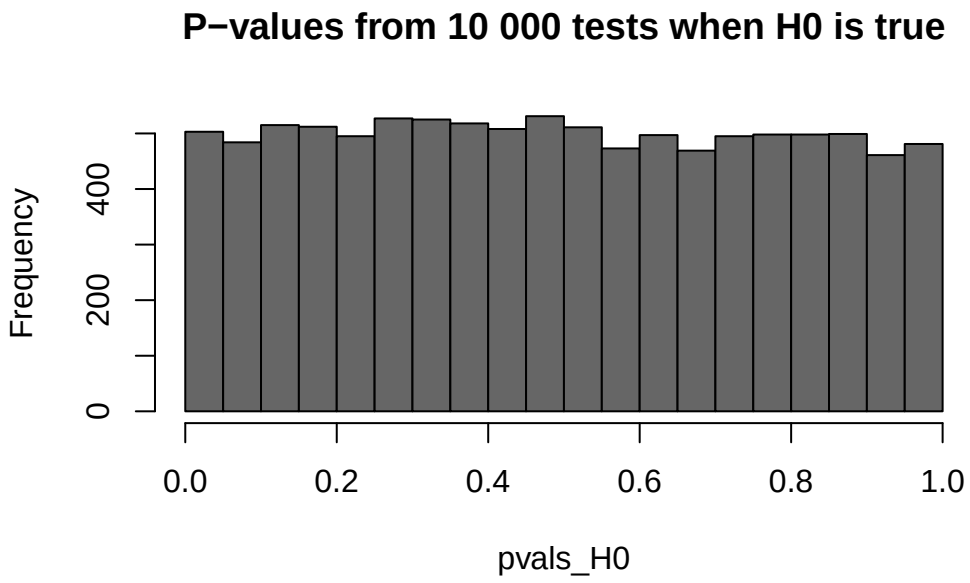
In our case we will go through 2 different methods that have been developed to control the FDR.

BH (Benjamini-Hochberg) - Assumptions

- **P-VALUES VALIDITY:** it requires that under the null hypothesis the p-values are uniformly distributed on [0;1].

```r
set.seed(1)
n <- 24
m <- 10000
mat <- matrix(rnorm(n*m), ncol = n) #data taken from the same normal distribution
g <- factor(c(rep(0, n/2), rep(1, n/2)))
pvals_H0 <- rowttests(mat, g)$p.value
```

```r
hist(pvals_H0,
     main = "P-values from 10 000 tests when H0 is true",
     col = "grey40")
```



- **INDEPENDENCY:** it requires that the test statistics (p-values) are independent. In other words, BH method controls the FDR at a level alpha under independent or positively dependent test-statistic.

For the purpose of the project we are going to assume that both of the assumptions are respected.

BH - Run the code

```r
bh_values <- p.adjust(pvals, method = "fdr") #p-values obtained with BH method

sum(bh_values < 0.05)
```

```
[1] 13
```

With BH method we have obtained a slightly higher number of positives then the Bonferroni method

```r
sum(bh_values < 0.05) > sum(pvals < bonf_alpha)
```

```
[1] TRUE
```

In our case, we are working with real gene expression data from the GSE5859Subset dataset, where we do not know a priori how many null hypotheses are true. It is likely that some genes are truly differentially expressed.

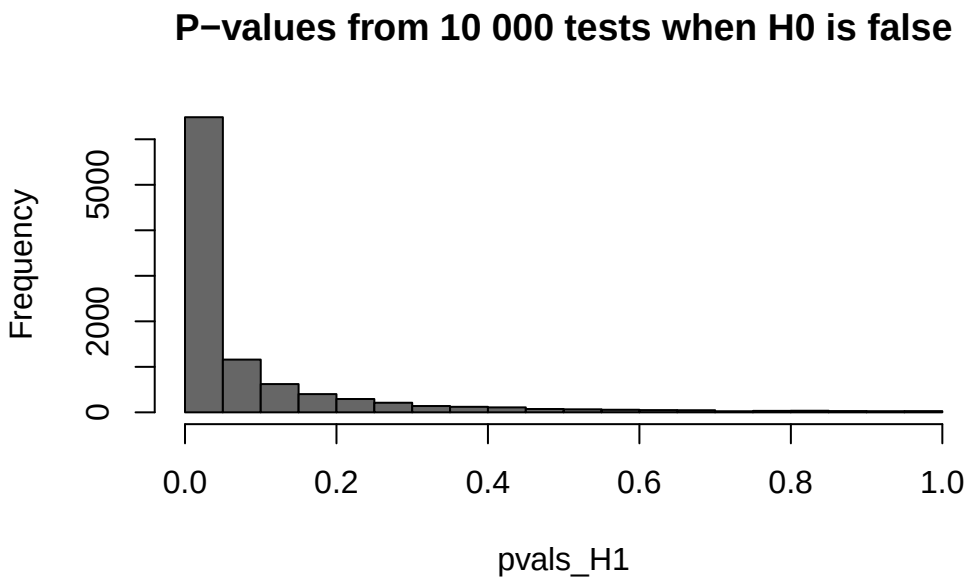
The BH method still controls the FDR under valid assumptions, but it can be conservative because it assumes all nulls are true (i.e., _0=1). In contrast, methods like Storey's adapt to the data by estimating the true proportion of nulls, _0<1, and can therefore yield more discoveries while still controlling the FDR.

STOREY - Assumptions

As we previously said, Storey method does not assume _0=1, instead it compute its estimate directly from the data. (here we will not show the procedure to get it).

- **DISTRIBUTION UNDER NULL and ALTERNATIVE:** just like BH, Storey assumes that under the null hypothesis, p-values are uniformly distributed on [0;1] and under the alternative hypothesis, p-values are concentrated near 0. The graph below show how under the alternative hypothesis the p-values are concentrated near 0.

```r
set.seed(1)
n <- 24
m <- 10000
mat <- matrix(rnorm(n*m), ncol = n) #data taken from the same normal distribution
mat[, 13:ncol(mat)] <- mat[, 13:ncol(mat)] + 1
g <- factor(c(rep(0, n/2), rep(1, n/2)))
pvals_H1 <- rowttests(mat, g)$p.value
hist(pvals_H1,
     main = "P-values from 10 000 tests when H0 is false",
     col = "grey40")
```



- **INDEPENDENCE:** it assumes p-values are independent.

STOREY - Run the code

```
library(qvalue)
sortey <- qvalue(pvals)
qvals <- sortey$qvalues
sum(qvals < 0.05) # discoveries at estimated pFDR < 5%
```

```
[1] 22
```

Using the Storey method, we identify 22 discoveries with an estimated FDR $< 5\%$. The `qvalue` function computes a q-value for each p-value, representing the expected false discovery rate if we reject that hypothesis and all others with smaller p-values.

```
qvals[length(pvals)]
```

```
[1] 0.6695489
```

For example, the q-value of the largest p-value in our set is 0.6696, meaning that if we were to reject **all** hypotheses, the expected FDR would be 66.96%.

The Storey method also estimates the proportion of true null hypotheses in the data

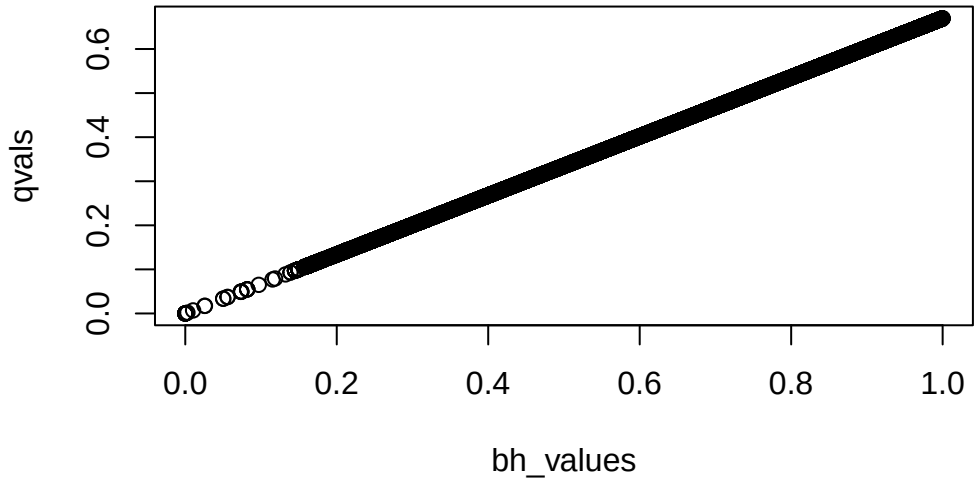
```
pi0 <- sortey$pi0
print(signif(pi0, 4))
```

```
[1] 0.6696
```

PLOT - BH vs STOREY

This means roughly 67% of the tested hypotheses are estimated to be null.

```
plot(bh_values, qvals)
```



In general the p-values computed using BH method are higher than the ones computed with Storey's method. This means that BH is more conservative while Storey allows for more null to be false (remember that with Storey we estimate the proportion of true null in our data set).

CHEATING SIMULATION

We have computed the FWER, FDR and pFDR to the "GSE5859Subset" and we have obtained some results that we cannot verify, since we do not know the true proportion of null hypotheses in the real dataset.

In order to give a better demonstration of the differences among these metrics we are going to define a customed data set where the first 500 t-tests are false under the null hypothesis and the other are true under the null.

"Cheating" refers to the fact that we have a prior knowledge of which tests are truly null or not.

```
set.seed(1)
positives <- 500
n <- 24
m <- 8793
m0 <- m - positives
g <- factor(c(rep(0,n/2), rep(1,n/2)))
bonf <- 0.05 / m

mat <- matrix(rnorm(n*m),m,n)
delta <- 2 #difference of "2" between positives and not
mat[1:positives, 1:(n/2)] <- mat[1:positives, 1:(n/2)] + delta #for the first 500 rows we hav
pvals <- rowttests(mat, g)$p.value #pvalues

bonf_p <- as.integer(sum(pvals < bonf))
bonf_fp <- length(which(which(pvals < bonf) > 500))
bonf_tp <- length(which(which(pvals < bonf) < 500))
bonf_fn <- positives - bonf_tp
bonf_vec <- c("FP" = bonf_fp , "FN" = bonf_fn)
```

We are going to perform a Monte Carlo simulation — that is, replicate the code above 1000 times — to properly assess the behavior of different multiple testing metrics. First, we will use the Bonferroni correction, then compute the FDR using the BH procedure, and finally evaluate the pFDR estimated using Storey's method.

Bonferroni

```r
set.seed(1)
positives <- 500
n <- 24
m <- 8793
m0 <- m - positives
g <- factor(c(rep(0,n/2), rep(1,n/2)))
bonf <- 0.05 / m

bonf_list <- replicate(1000, {
  mat <- matrix(rnorm(n*m),m,n)
  delta <- 2 #difference of "2" between positives and not
  mat[1:positives,1:(n/2)] <- mat[1:positives,1:(n/2)]+delta #for the first 500 rows we have
  pvals <- rowttests(mat, g)$p.value #pvalues

  bonf_p <- as.integer(sum(pvals < bonf))
  bonf_fp <- length(which(which(pvals < bonf) > 500))
  bonf_tp <- length(which(which(pvals < bonf) < 500))
  bonf_fn <- positives - bonf_tp
  bonf_vec <- c("FP" = bonf_fp , "FN" = bonf_fn)
})

bonf_fpr <- mean(bonf_list[1,]) / m0 #expected False Positive Rate if we use Bonferroni
bonf_fnr <- mean(bonf_list[2,]) / positives  #expected False Negative Rate if we use Bonferr

print(paste0("Expected False Positive Rate if we use Bonferroni: ",signif(bonf_fpr,4)*100, "%
```

```
[1] "Expected False Positive Rate if we use Bonferroni: 0.0005306%"
```

```r
print(paste0("Expected False Negative Rate if we use Bonferroni: ",signif(bonf_fnr,4)*100, "%
```

```
[1] "Expected False Negative Rate if we use Bonferroni: 76.38%"
```

As we described in detail above, the Bonferroni method dramatically reduces the Type I error rate. However, this comes at the cost of a significant increase in the Type II error rate, resulting in very low statistical power.

BH

```r
set.seed(1)
alpha <-0.05
bh_list <- replicate(1000, {
  mat <- matrix(rnorm(n*m),m,n)
  delta <- 2 #difference of "2" between positives and not
  mat[1:positives,1:(n/2)] <- mat[1:positives,1:(n/2)]+delta #for the first 500 rows we have
  pvals <- rowttests(mat, g)$p.value #pvalues

  bh_values <- p.adjust(pvals, method = "fdr")
  bh_p <- as.integer(sum(bh_values < 0.05))
  bh_fp <- length(which(which(bh_values < 0.05) > 500))
  bh_tp <- length(which(which(bh_values < 0.05) < 500))
  bh_fn <- positives - bh_tp
  bh_vec <- c("FP" = bh_fp , "FN" = bh_fn)
})

bh_fpr <- mean(bh_list[1,]) / m0 #expected False Positive Rate if we use FDR (BH)
bh_fnr <- mean(bh_list[2,]) / positives  #expected False Negative Rate if we use FDR (BH)

print(paste0("Expected False Positive Rate if we use BH: ",signif(bh_fpr,4)*100, "%"))
```

```
[1] "Expected False Positive Rate if we use BH: 0.2738%"
```

```r
print(paste0("Expected False Negative Rate if we use BH: ",signif(bh_fnr,4)*100, "%"))
```

```
[1] "Expected False Negative Rate if we use BH: 8.326%"
```

Since we designed the data set with 500 true positives and 8293 true null hypotheses, we know the ground truth.As expected, with a high proportion of true nulls, the BH method controls the FDR effectively and behaves conservatively, leading to a modest false negative rate (~8.3%).

Storey

```r
set.seed(1)
alpha <-0.05
qvals_list <- replicate(1000, {
  mat <- matrix(rnorm(n*m),m,n)
  delta <- 2 #difference of "2" between positives and not
```

```
  mat[1:positives,1:(n/2)] <- mat[1:positives,1:(n/2)]+delta #for the first 500 rows we have
  pvals <- rowttests(mat, g)$p.value #pvalues

  qvals_values <- qvalue(pvals)$qvalues
  qvals_p <- as.integer(sum(qvals_values < 0.05))
  qvals_fp <- length(which(which(qvals_values < 0.05) > 500))
  qvals_tp <- length(which(which(qvals_values < 0.05) < 500))
  qvals_fn <- positives - qvals_tp
  qvals_vec <- c("FP" = qvals_fp , "FN" = qvals_fn)
})

qvals_fpr <- mean(qvals_list[1,]) / m0 #expected False Positive Rate if we use pFDR (Sortey r
qvals_fnr <- mean(qvals_list[2,]) / positives  #expected False Negative Rate if we use pFDR

print(paste0("Expected False Positive Rate if we use BH: ",signif(qvals_fpr,4)*100, "%"))
```

```
[1] "Expected False Positive Rate if we use BH: 0.2933%"
```

```
print(paste0("Expected False Negative Rate if we use BH: ",signif(qvals_fnr,4)*100, "%"))
```

```
[1] "Expected False Negative Rate if we use BH: 7.915%"
```

As expected, the Storey method is less conservative than the BH method, since it uses an estimate of the true null proportion (0) rather than assuming all hypotheses are null. This results in a higher estimated pFDR but a lower false negative rate, reflecting improved power.

CONCLUSIONS

In this analysis, we compared three multiple testing correction methods: `FWER` control using the Bonferroni procedure, `FDR` control via the Benjamini-Hochberg (BH) method, and `pFDR` estimation using Storey's q-value approach. Each method offers a different balance between error control and statistical power.

- Bonferroni is the most conservative, aiming to minimize the probability of even a single false positive. While this provides strong control over Type I errors, it comes at the cost of low power and a high false negative rate, especially in high-dimensional settings.
- The BH procedure relaxes this strictness by controlling the expected proportion of false discoveries (FDR) rather than eliminating them entirely. This makes it more powerful and practical for exploratory analyses, particularly when some false positives can be tolerated.

- Storey's method builds upon BH by estimating the proportion of true null hypotheses directly from the data. This leads to a more adaptive thresholding strategy that further increases power while still maintaining control over the expected proportion of false discoveries (pFDR), especially in settings where many hypotheses are truly non-null.

Overall, Bonferroni is best suited for confirmatory analyses where any false positives are unacceptable, BH offers a balanced approach for general use, and Storey's method is ideal for large-scale studies where discovery is prioritized and some false positives are acceptable.