Multiple Testing Corrections

Alexander McLain

Simulated Data Examples

We will use a Monte Carlo simulation using our mice data to imitate a situation in which we perform tests for 10,000 different tests, none of them having an effect on the outcome. This implies that the null hypothesis is true for tests and thus $M = M_0 = 10,000$ and $M_1 = 0$. Let's run the tests with a sample size of $n = 12$ and compute R. Our procedure will declare any test achieving a p-value smaller than $\alpha = 0.05$ as significant.

```
set.seed(1)
population = rnorm(1e7)
alpha <- 0.05
N <- 12
m <- 10000
pvals <- replicate(m,{
  control = sample(population,N)
  treatment = sample(population,N)
  t.test(treatment,control)$p.value
})
```

Let's see what the value of R is for this experiment.

```
sum(pvals < 0.05) ##This is R
```

```
## [1] 472
```

Now we'll simulate some data were there is an affect of treatment. Let's set up the values:

```
alpha <- 0.05
N <- 12
m <- 10000
p0 <- 0.90 ##10% of hypotheses are signals, 90% are nulls
m0 <- m*p0
m1 <- m-m0
nullHypothesis <- c( rep(TRUE,m0), rep(FALSE,m1))
delta <- 1
```

Now, let's simulate the data.

```
set.seed(1)
calls <- sapply(1:m, function(i){
  control <- sample(population,N)
  treatment <- sample(population,N)
  if(!nullHypothesis[i]) treatment <- treatment + delta
  ifelse( t.test(treatment,control)$p.value < alpha,
          "Called Significant",
          "Not Called Significant")
})
```

Let's see how we did in our results:

```r
null_hypothesis <- factor( nullHypothesis, levels=c("TRUE","FALSE"))
table(null_hypothesis,calls)
```

null_hypothesis/calls	Called Significant	Not Called Significant
TRUE	409	8591
FALSE	644	356

Note that this indicates that (per test) the power $1 - \beta$ is ~ 0.644 (true power is 0.649). The first column of the table above shows us V and S. Note that V and S are random variables. If we run the simulation repeatedly, these values change. Here is a quick example:

```r
B <- 10 ##number of simulations
VandS <- replicate(B,{
  calls <- sapply(1:m, function(i){
    control <- sample(population,N)
    treatment <- sample(population,N)
    if(!nullHypothesis[i]) treatment <- treatment + delta
    t.test(treatment,control)$p.val < alpha
  })
  cat("V =",sum(nullHypothesis & calls), "S =",sum(!nullHypothesis & calls),
     "V/R=", sum(nullHypothesis & calls)/sum(calls),"\n")
  c(sum(nullHypothesis & calls),sum(!nullHypothesis & calls),sum(nullHypothesis & calls)/sum(calls))
})
```

```
## V = 444 S = 649 V/R= 0.4062214
## V = 422 S = 638 V/R= 0.3981132
## V = 440 S = 645 V/R= 0.40553
## V = 434 S = 638 V/R= 0.4048507
## V = 427 S = 638 V/R= 0.400939
## V = 428 S = 604 V/R= 0.4147287
## V = 445 S = 668 V/R= 0.3998203
## V = 438 S = 663 V/R= 0.3978202
## V = 471 S = 632 V/R= 0.4270172
## V = 477 S = 651 V/R= 0.4228723
```

Family-Wise Error rate

To do an FWER correction we can use Bonfeerroni or the Holm procedure. These (alohg with other methods) can be fitted using the *p.adjust* function in R.

```r
?p.adjust
```

```
Adjust P-values for Multiple Comparisons

Description:

     Given a set of p-values, returns p-values adjusted using one of
     several methods.

Usage:

     p.adjust(p, method = p.adjust.methods, n = length(p))
     p.adjust.methods
     # c("holm", "hochberg", "hommel", "bonferroni", "BH", "BY",
     #   "fdr", "none")

Arguments:

       p: numeric vector of p-values (possibly with 'NA's).  Any other
          R object is coerced by 'as.numeric'.

  method: correction method, a 'character' string.  Can be abbreviated.

       n: number of comparisons, must be at least 'length(p)'; only set
          this (to non-default) when you know what you are doing!

Details:

     The adjustment methods include the Bonferroni correction
     ('"bonferroni"') in which the p-values are multiplied by the
     number of comparisons.  Less conservative corrections are also
     included by Holm (1979) ('"holm"'), Hochberg (1988)
     ('"hochberg"'), Hommel (1988) ('"hommel"'), Benjamini & Hochberg
     (1995) ('"BH"' or its alias '"fdr"'), and Benjamini & Yekutieli
     (2001) ('"BY"'), respectively.  A pass-through option ('"none"')
     is also included.  The set of methods are contained in the
     'p.adjust.methods' vector for the benefit of methods that need to
     have the method as an option and pass it on to 'p.adjust'.

     The first four methods are designed to give strong control of the
     family-wise error rate.  There seems no reason to use the
     unmodified Bonferroni correction because it is dominated by Holm's
     method, which is also valid under arbitrary assumptions.

     Hochberg's and Hommel's methods are valid when the hypothesis
     tests are independent or when they are non-negatively associated
     (Sarkar, 1998; Sarkar and Chang, 1997).  Hommel's method is more
     powerful than Hochberg's, but the difference is usually small and
     the Hochberg p-values are faster to compute.

     The '"BH"' (aka '"fdr"') and '"BY"' methods of Benjamini,
     Hochberg, and Yekutieli control the false discovery rate, the
     expected proportion of false discoveries amongst the rejected
     hypotheses.  The false discovery rate is a less stringent
     condition than the family-wise error rate, so these methods are
     more powerful than the others.

     Note that you can set 'n' larger than 'length(p)' which means the
     unobserved p-values are assumed to be greater than all the
     observed p for '"bonferroni"' and '"holm"' methods and equal to 1
     for the other methods.

References:

     Benjamini, Y., and Hochberg, Y. (1995).  Controlling the false
```

```
discovery rate: a practical and powerful approach to multiple
testing.  _Journal of the Royal Statistical Society Series B_,
*57*, 289-300.  doi:10.1111/j.2517-6161.1995.tb02031.x
<https://doi.org/10.1111/j.2517-6161.1995.tb02031.x>.

Benjamini, Y., and Yekutieli, D. (2001).  The control of the false
discovery rate in multiple testing under dependency.  _Annals of
Statistics_, *29*, 1165-1188.  doi:10.1214/aos/1013699998
<https://doi.org/10.1214/aos/1013699998>.

Holm, S. (1979).  A simple sequentially rejective multiple test
procedure.  _Scandinavian Journal of Statistics_, *6*, 65-70.
<https://www.jstor.org/stable/4615733>.

Hommel, G. (1988).  A stagewise rejective multiple test procedure
based on a modified Bonferroni test.  _Biometrika_, *75*, 383-386.
doi:10.2307/2336190 <https://doi.org/10.2307/2336190>.

Hochberg, Y. (1988).  A sharper Bonferroni procedure for multiple
tests of significance.  _Biometrika_, *75*, 800-803.
doi:10.2307/2336325 <https://doi.org/10.2307/2336325>.

Shaffer, J. P. (1995).  Multiple hypothesis testing.  _Annual
Review of Psychology_, *46*, 561-584.
doi:10.1146/annurev.ps.46.020195.003021
<https://doi.org/10.1146/annurev.ps.46.020195.003021>.  (An
excellent review of the area.)

Sarkar, S. (1998).  Some probability inequalities for ordered MTP2
random variables: a proof of Simes conjecture.  _Annals of
Statistics_, *26*, 494-504.  doi:10.1214/aos/1028144846
<https://doi.org/10.1214/aos/1028144846>.

Sarkar, S., and Chang, C. K. (1997).  The Simes method for
multiple hypothesis testing with positively dependent test
statistics.  _Journal of the American Statistical Association_,
*92*, 1601-1608.  doi:10.2307/2965431
<https://doi.org/10.2307/2965431>.

Wright, S. P. (1992).  Adjusted P-values for simultaneous
inference.  _Biometrics_, *48*, 1005-1013.  doi:10.2307/2532694
<https://doi.org/10.2307/2532694>.  (Explains the adjusted P-value
approach.)
```

Controlling the FWER at 0.05 is a very conservative approach. Using the p-values computed in the previous section…

```r
set.seed(1)
pvals <- sapply(1:m, function(i){
  control <- sample(population,N)
  treatment <- sample(population,N)
  if(!nullHypothesis[i]) treatment <- treatment + delta
  t.test(treatment,control)$p.value
})
```

. . . we note that only:

```r
p_bonf <- p.adjust(pvals,method = "bonferroni")
p_holm <- p.adjust(pvals,method = "holm")
sum(p_bonf <= alpha)
```

```
## [1] 5
```

```r
sum(p_holm <= alpha)
```

```
## [1] 5
```

are called significant after applying the Bonferroni and Holm procedures. This is despite having 1000 tests that are actually significant (and a good power for each test).

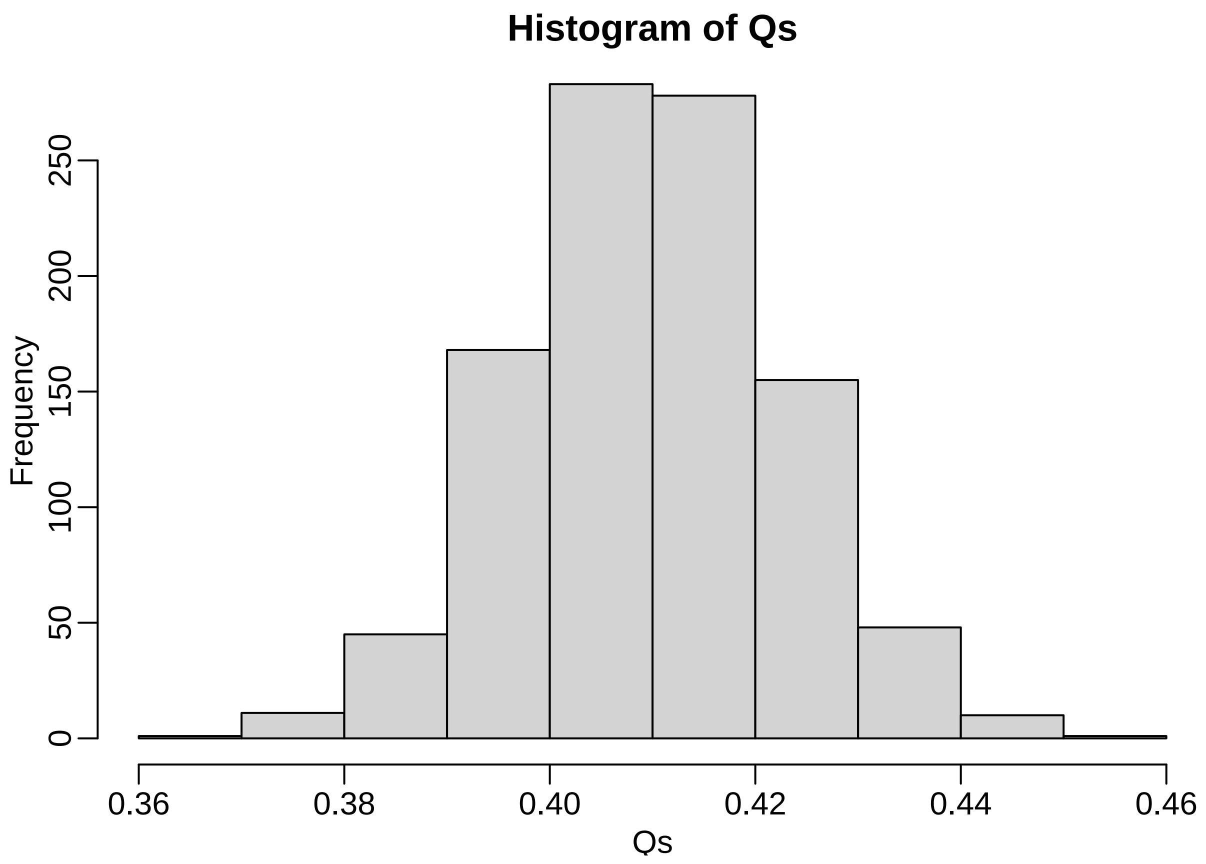
False Discovery Rate

Before running the simulation, we are going to vectortize the code. This means that instead of using *sapply* to run M tests, we will create a matrix with all data in one call to sample. This code runs several times faster than the code above, which is necessary here due to the fact that we will be generating several simulations. Understanding this chunk of code and how it is equivalent to the code above using sapply will take a you long way in helping you code efficiently in R.

```r
library(genefilter) ##rowttests is here (genefilter is available on Bioconductor)
set.seed(1)
##Define groups to be used with rowttests
g <- factor( c(rep(0,N),rep(1,N)) )
B <- 1000 ##number of simulations
Qs <- replicate(B,{
  ##matrix with control data (rows are tests, columns are mice)
  controls <- matrix(sample(population, N*m, replace=TRUE),nrow=m)

  ##matrix with control data (rows are tests, columns are mice)
  treatments <-  matrix(sample(population, N*m, replace=TRUE),nrow=m)

  ##add effect to 10% of them
  treatments[which(!nullHypothesis),]<-treatments[which(!nullHypothesis),]+delta

  ##combine to form one matrix
  dat <- cbind(controls,treatments)

  calls <- rowttests(dat,g)$p.value < alpha
  R=sum(calls)
  Q=ifelse(R>0,sum(nullHypothesis & calls)/R,0)
  return(Q)
})
```

The code above is a Monte Carlo simulation that generates $10,000$ experiments $1,000$ times, each time saving the observed Q. Here is a histogram of these values:

```r
library(rafalib)
mypar(1,1)
hist(Qs) ##Q is a random variable, this is its distribution
```

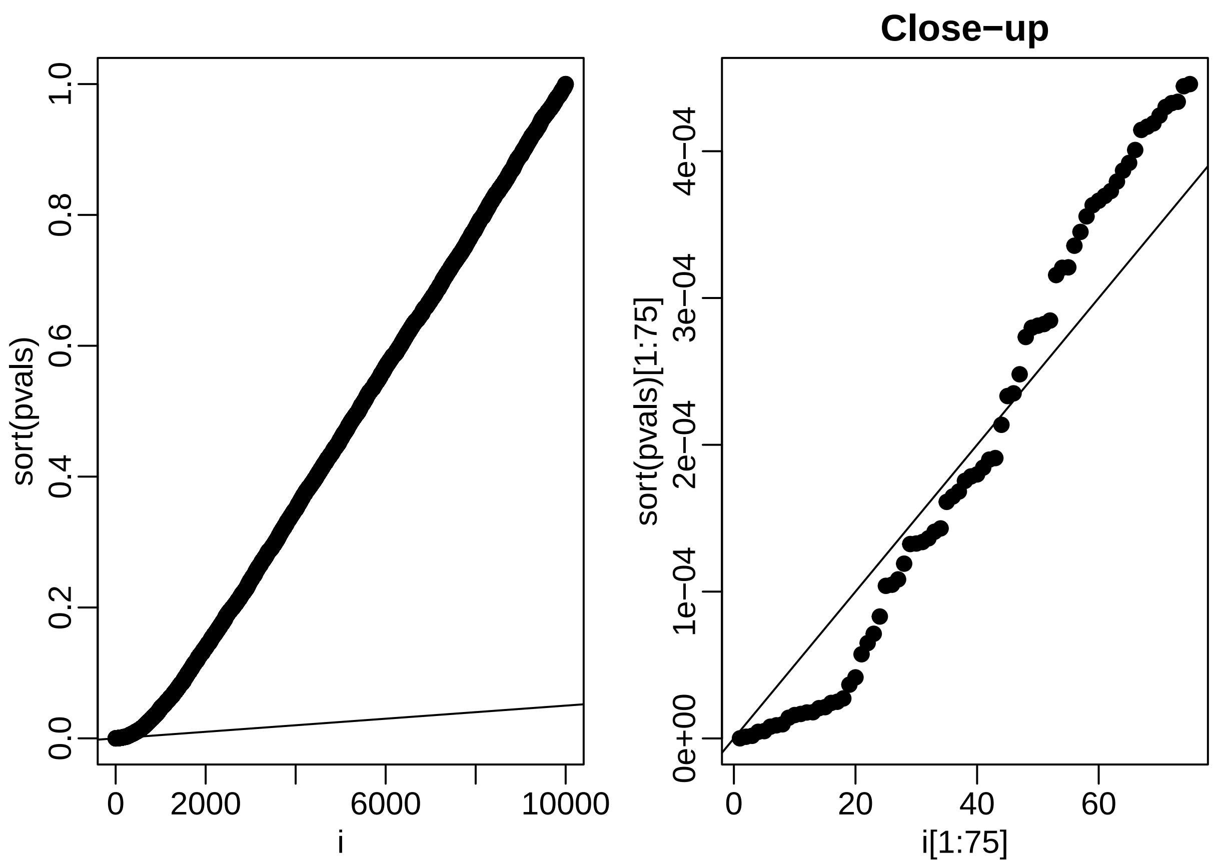


```r
FDR=mean(Qs)
FDR
```

```
## [1] 0.409716
```

Let's look at the BH procedure for the above data:

```r
alpha <- 0.05
i = seq(along=pvals)

mypar(1,2)
plot(i,sort(pvals), pch = 19)
abline(0,i/m*alpha)
##close-up
plot(i[1:75],sort(pvals)[1:75],main="Close-up", pch = 19)
abline(0,i/m*alpha)
```



```
k <- max( which( sort(pvals) < i/m*alpha) )
cutoff <- sort(pvals)[k]
cat("k =",k,"p-value cutoff=",cutoff)
```

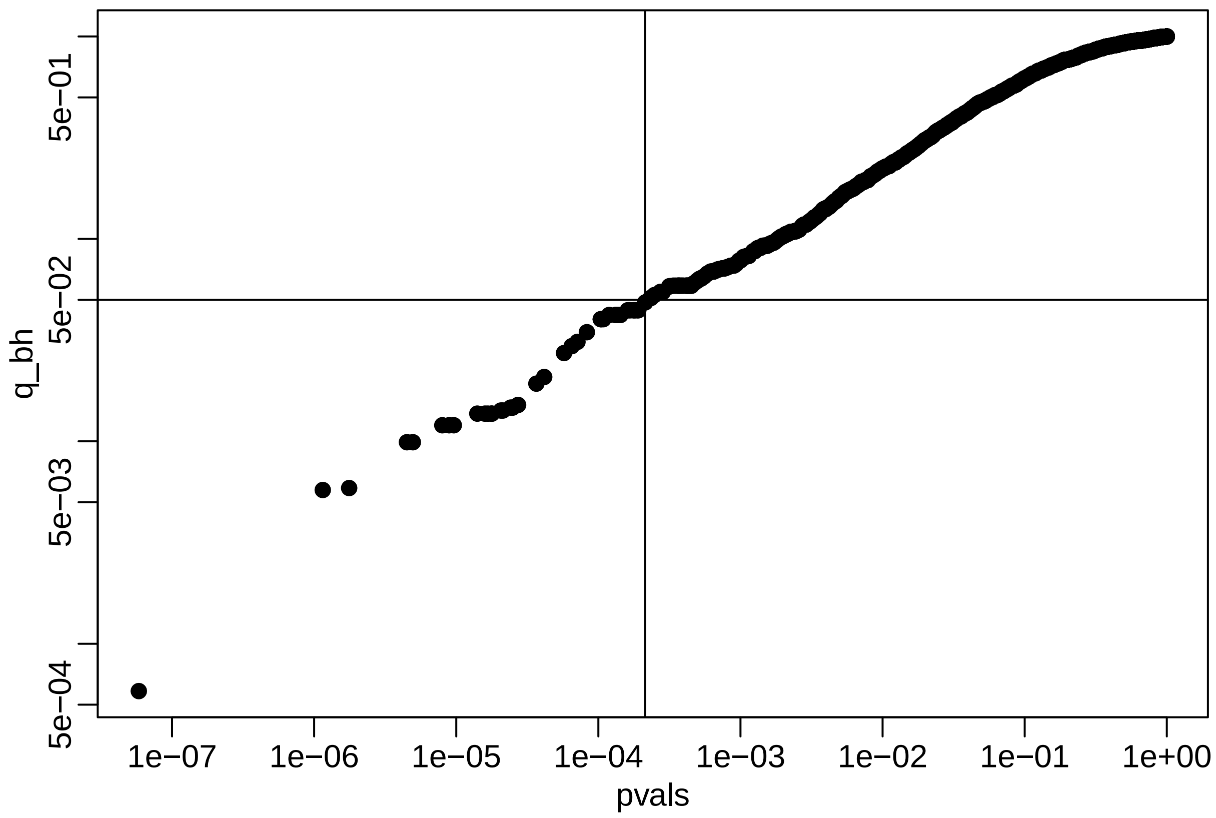
```
## k = 44 p-value cutoff= 0.0002136265
```

Or using *p.adjust* we have:

```
q_bh <- p.adjust(pvals, method="fdr")
table(null_hypothesis, q_bh <= alpha)
```

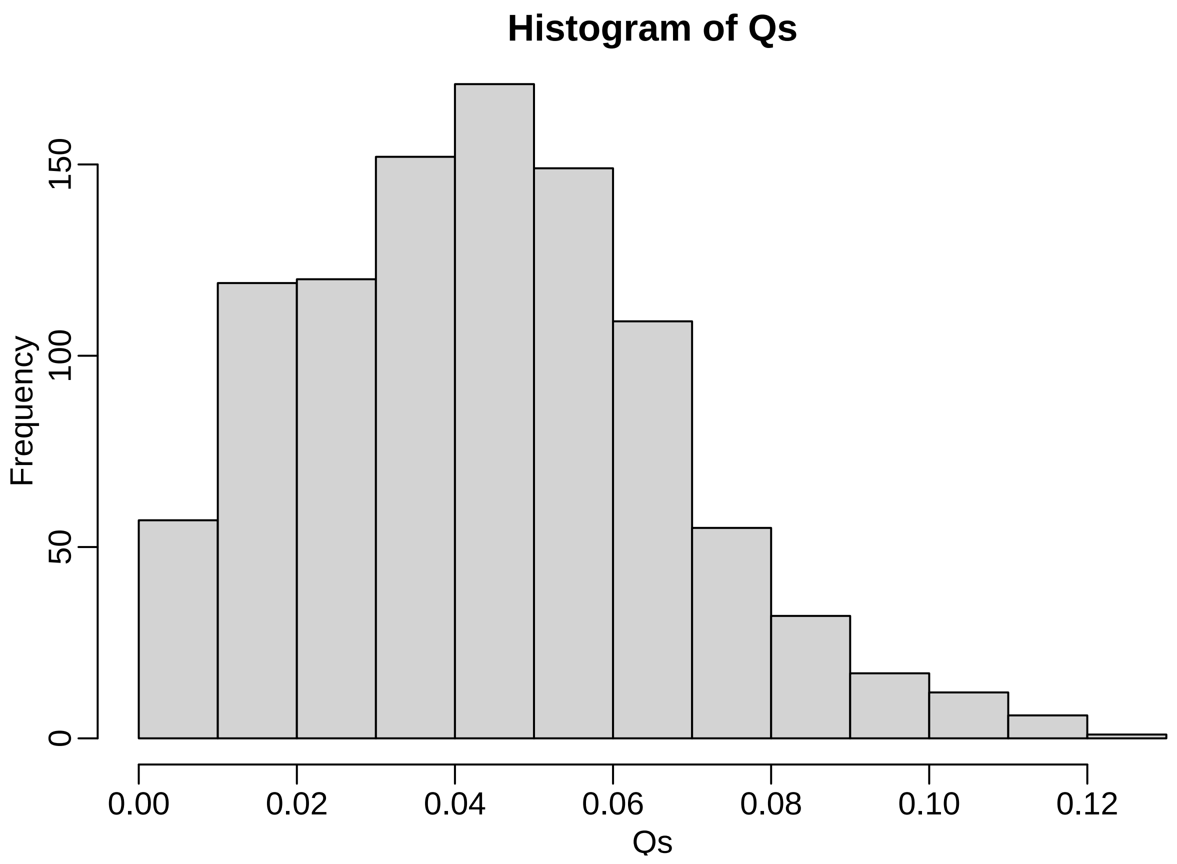
null_hypothesis/	FALSE	TRUE
TRUE	9000	0
FALSE	956	44

```
mypar(1,1)
plot(pvals,q_bh,log="xy", pch = 19)
abline(h=alpha,v=cutoff) ##cutoff was computed above
```



We can run a Monte-Carlo simulation to confirm that the FDR is in fact lower than .05. We compute all p-values first, and then use these to decide which get called.

```
alpha <- 0.05
B <- 1000 ##number of simulations. We should increase for more precision
res <- replicate(B,{
  controls <- matrix(sample(population, N*m, replace=TRUE),nrow=m)
  treatments <-  matrix(sample(population, N*m, replace=TRUE),nrow=m)
  treatments[which(!nullHypothesis),]<-treatments[which(!nullHypothesis),]+delta
  dat <- cbind(controls,treatments)
  pvals <- rowttests(dat,g)$p.value
  ##then the FDR
  calls <- p.adjust(pvals,method="fdr") < alpha
  R=sum(calls)
  Q=ifelse(R>0,sum(nullHypothesis & calls)/R,0)
  return(c(R,Q))
})
Qs <- res[2,]
mypar(1,1)
hist(Qs) ##Q is a random variable, this is its distribution
```



```
summary(Qs)
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
0	0.0266667	0.0434783	0.043753	0.0588235	0.125

```
Rs <- res[1,]
mean(Rs==0)*100
```

```
## [1] 0
```

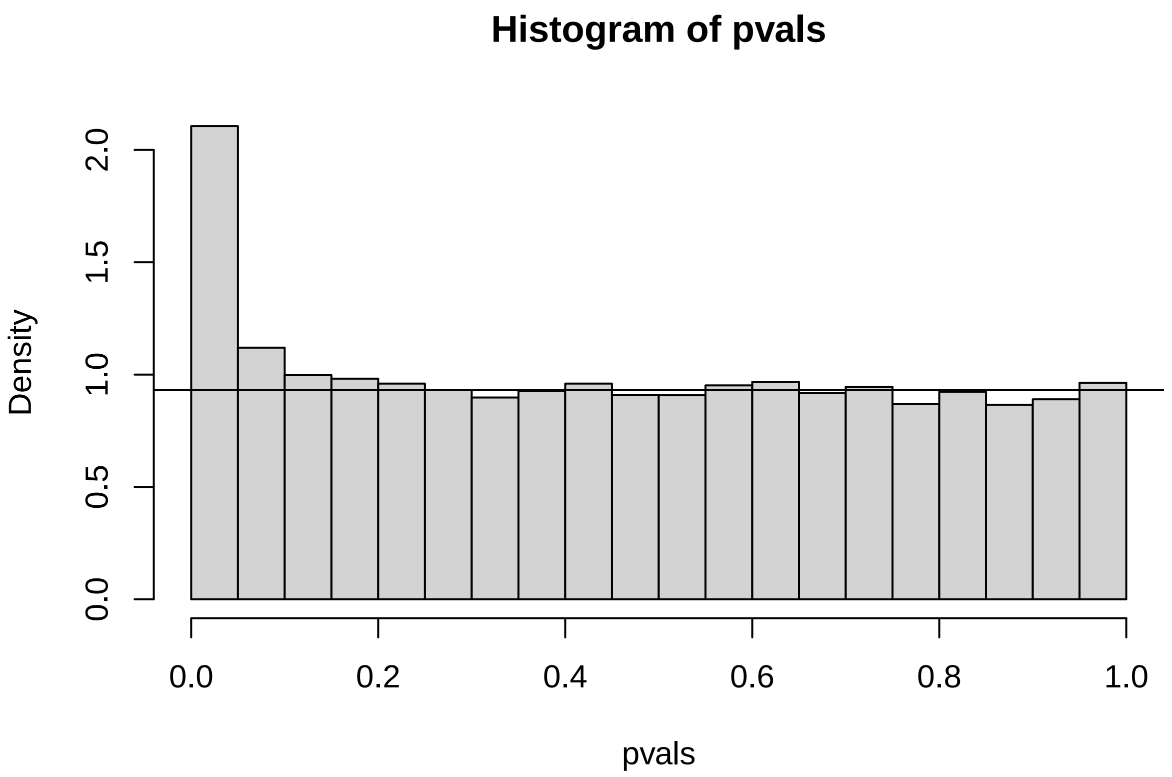
```
summary(Rs*(1-Qs)) ## Number of correct rejections
```

Min.	1st Qu.	Median	Mean	3rd Qu.	Max.
7	66	79	79.724	93	145

Q-values

In this Section we'll use the q-value approach from Storey and Tibshraini (2003). To do this we'll first estimate π_0. When doing this we'll use $\lambda = 0.1$, which is the value I usually use in practice as it has been shown to be robust to dependence in the p-values.

```
hist(pvals,breaks=seq(0,1,0.05),freq=FALSE)
lambda = 0.1
pi0=sum(pvals> lambda) /((1-lambda)*m)
abline(h= pi0)
```



```r
print(pi0)
```

```
## [1] 0.9318889
```

Then, using the relationship between the BH and Storey procedures that we discussed, the estimated q_values are:

```r
q_st <- p.adjust(pi0*pvals, method="fdr")
table(null_hypothesis, q_st <= alpha)
```

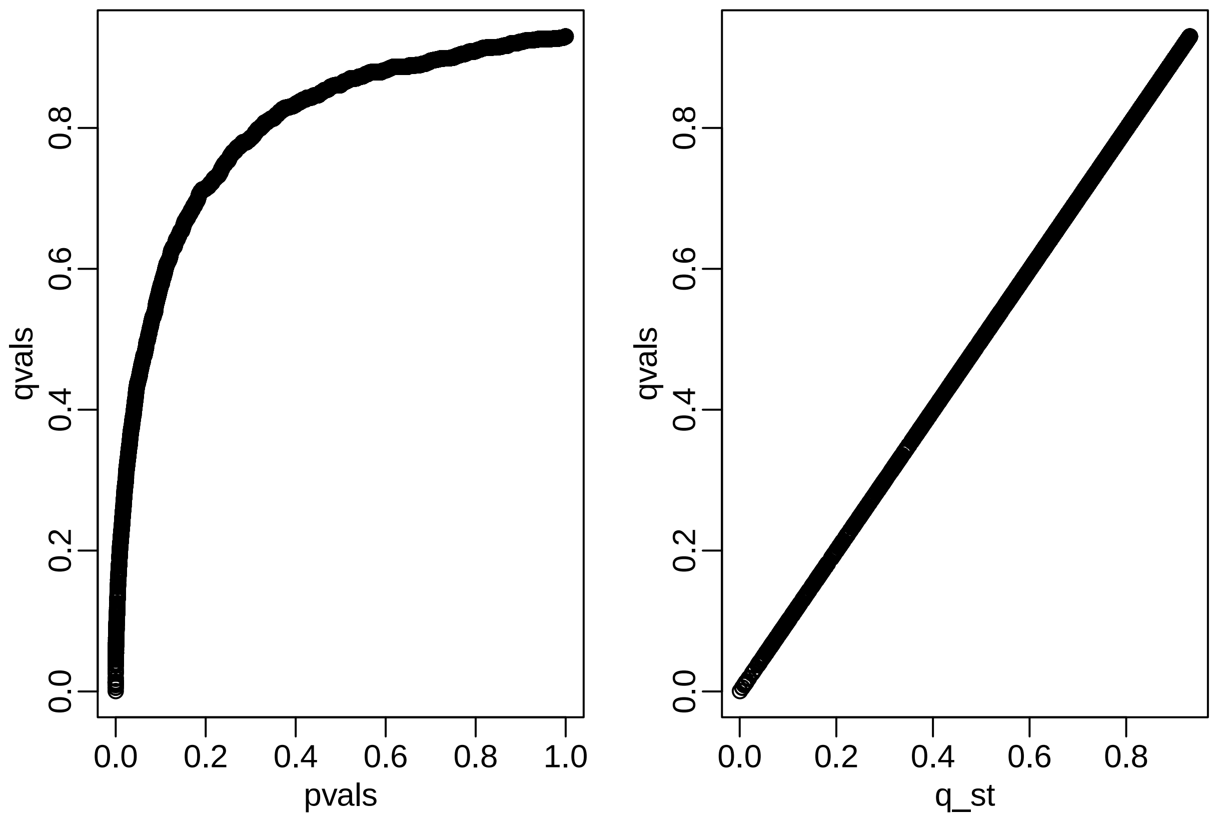
null_hypothesis/	FALSE	TRUE
TRUE	9000	0
FALSE	953	47

This can all be done using the *qvalue* package (available on bioconductor not CRAN):

```r
library(qvalue)
?qvalue
```

```
Estimate the q-values for a given set of p-values

Description:

     Estimate the q-values for a given set of p-values.  The q-value of
     a test measures the proportion of false positives incurred (called
     the false discovery rate) when that particular test is called
     significant.

Usage:
     qvalue(p, fdr.level = NULL, pfdr = FALSE, lfdr.out = TRUE, pi0 = NULL,
       ...)

Arguments:

       p: A vector of p-values (only necessary input).

fdr.level: A level at which to control the FDR. Must be in (0,1].
          Optional; if this is selected, a vector of TRUE and FALSE is
          returned that specifies whether each q-value is less than
          fdr.level or not.

    pfdr: An indicator of whether it is desired to make the estimate
          more robust for small p-values and a direct finite sample
          estimate of pFDR - optional.

lfdr.out: If TRUE then local false discovery rates are returned.
          Default is TRUE.

     pi0: It is recommended to not input an estimate of pi0.
          Experienced users can use their own methodology to estimate
          the proportion of true nulls or set it equal to 1 for the BH
          procedure.

     ...: Additional arguments passed to 'pi0est' and 'lfdr'.

Details:

     The function 'pi0est' is called internally and calculates the
     estimate of pi_0, the proportion of true null hypotheses. The
     function 'lfdr' is also called internally and calculates the
     estimated local FDR values.  Arguments for these functions can be
     included via '...' and will be utilized in the internal calls made
     in 'qvalue'. See <http://genomine.org/papers/Storey_FDR_2011.pdf>
     for a brief introduction to FDRs and q-values.

Value:

     A list of object type "qvalue" containing:

    call: Function call.

     pi0: An estimate of the proportion of null p-values.

 qvalues: A vector of the estimated q-values (the main quantity of
          interest).

 pvalues: A vector of the original p-values.

    lfdr: A vector of the estimated local FDR values.

significant: If fdr.level is specified, and indicator of whether the
          q-value fell below fdr.level (taking all such q-values to be
          significant controls FDR at level fdr.level).

pi0.lambda: An estimate of the proportion of null p-values at each
          lambda value (see vignette).

  lambda: A vector of the lambda values utilized to obtain
          'pi0.lambda'.

References:

     Storey JD. (2002) A direct approach to false discovery rates.
     Journal of the Royal Statistical Society, Series B, 64: 479-498.
     <http://onlinelibrary.wiley.com/doi/10.1111/1467-9868.00346/abstract>
     Storey JD and Tibshirani R. (2003) Statistical significance for
     genome-wide experiments. Proceedings of the National Academy of
     Sciences, 100: 9440-9445.
     <http://www.pnas.org/content/100/16/9440.full>

     Storey JD. (2003) The positive false discovery rate: A Bayesian
     interpretation and the q-value. Annals of Statistics, 31:
     2013-2035.
     <http://projecteuclid.org/DPubS/Repository/1.0/Disseminate?view=body&id=pdf_1&handle=euclid.aos/10

     Storey JD, Taylor JE, and Siegmund D. (2004) Strong control,
     conservative point estimation, and simultaneous conservative
     consistency of false discovery rates: A unified approach. Journal
     of the Royal Statistical Society, Series B, 66: 187-205.
     <http://onlinelibrary.wiley.com/doi/10.1111/j.1467-9868.2004.00439.x/abstract>

     Storey JD. (2011) False discovery rates. In _International
     Encyclopedia of Statistical Science_.
     <http://genomine.org/papers/Storey_FDR_2011.pdf>
     <http://www.springer.com/statistics/book/978-3-642-04897-5>
```

```r
res <- qvalue(pvals)
qvals <- res$qvalues

mypar(1,2)
plot(pvals,qvals)
plot(q_st,qvals)
```



The *qvalue* package contains multiple methods to estimate π_0 see *?pi0est* for details.

Example of leukemia data

We'll illustrate the multiple testing methods with gene expression data from the leukemia ALL/AML study of Golub et al. (1999). Load the leukemia dataset:

```r
library(multtest) #Useful package for multiple testing and data
data(golub)
dim(golub)
```

```
## [1] 3051   38
```

Note that each column is a sample (subject), and $golub_{j,i}$ is the expression level for gene j in tumor mRNA sample i. All of the genes have identifiers and tumor class labelss (0 for ALL, 1 for AML).

```r
dim(golub.gnames)
```

```
## [1] 3051    3
```

```r
golub.gnames[1:4, ]
```

36	AFFX-HUMISGF3A/M97935_MA_at (endogenous control)	AFFX-HUMISGF3A/M97935_MA_at
37	AFFX-HUMISGF3A/M97935_MB_at (endogenous control)	AFFX-HUMISGF3A/M97935_MB_at
38	AFFX-HUMISGF3A/M97935_3_at (endogenous control)	AFFX-HUMISGF3A/M97935_3_at
39	AFFX-HUMRGE/M10098_5_at (endogenous control)	AFFX-HUMRGE/M10098_5_at

```
golub.cl
```

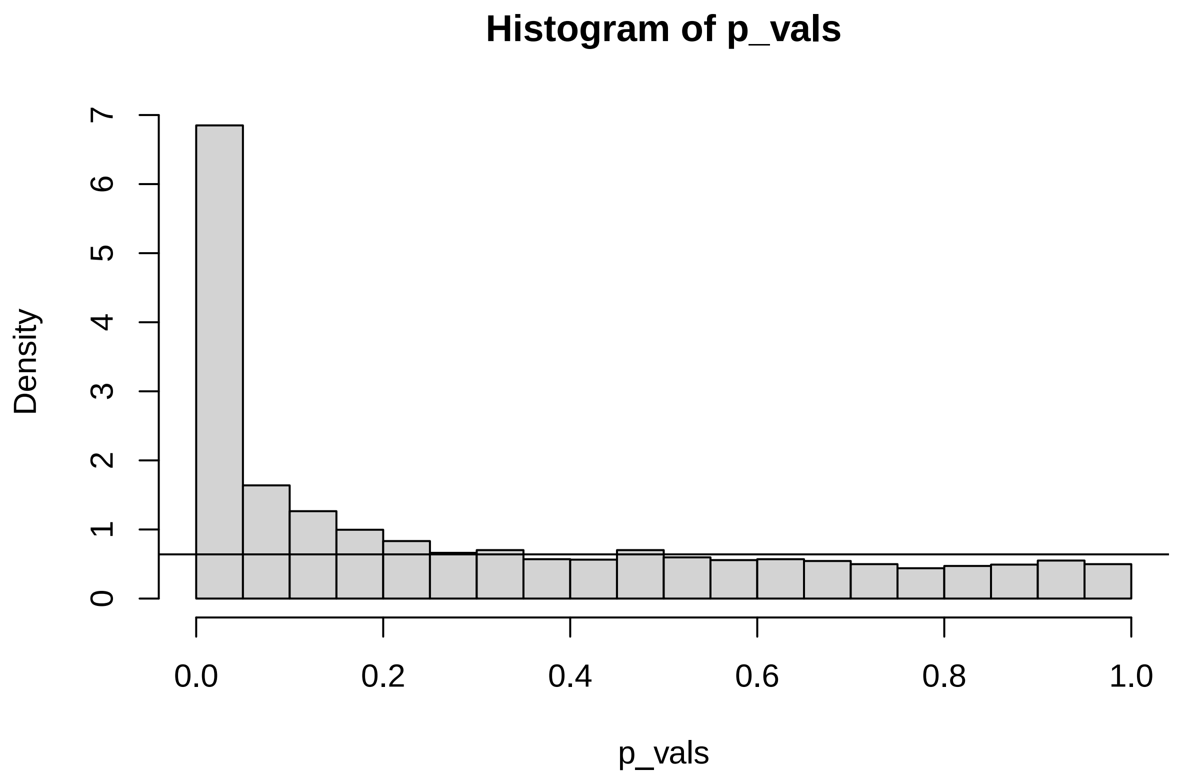
```
##  [1] 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1
```

We'll use the *rowttests* to compute the p-values (t-tests on each row):

```
p_vals <- rowttests( golub, factor(golub.cl) )$p.value
```

Let's take a look at a histogram of the p-values, and estimate the proportion of null hypotheses:

```
m <- nrow(golub)
hist(p_vals,breaks=seq(0,1,0.05),freq=FALSE)
lambda = 0.1
pi0=sum(p_vals> lambda) /((1-lambda)*m)
abline(h= pi0)
```



```
print(pi0)
```

```
## [1] 0.6394989
```

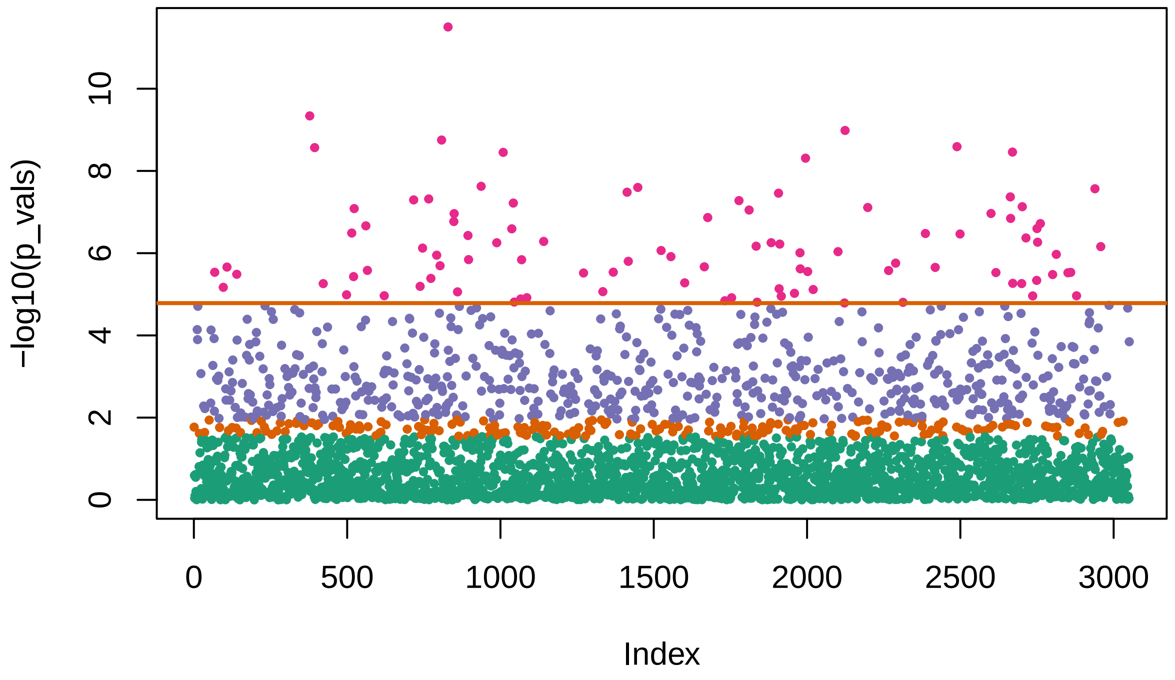
Now let's implement all of the methods to the p-values.

```
p_bonf <- p.adjust(p_vals, method = "bonf")
q_bh <- p.adjust(p_vals, method = "fdr")
q_st <- qvalue(p_vals)
data.frame("Bonf" = sum(p_bonf < 0.05), "BH" = sum(q_bh < 0.05), "Storey" = sum(q_st$qvalues < 0.05))
```

Bonf	BH	Storey
98	681	876

To display the results we can use a volcano plot:

```r
plot(-log10(p_vals), col = 1 + 1*I(p_bonf < 0.05) + 1*I(q_bh < 0.05)
     + 1*I(q_st$qvalues < 0.05), pch = 19, cex = 0.5)
abline(h = log10(m/alpha), lwd = 2, col = 2)
```


−log10(p_vals)
Index