

HDS Exercise set 2_Post

Shabbeer Hassan

Problem 2 – Solution

```
library(qvalue)

# Generate 5000 p-values
m = 1000
p = 5000
pval <- c(rbeta(m, 1, 100), runif(p-m, 0, 1))

# BH
BH <- p.adjust(pval, method = "BH", n = length(pval))

# Qvalue
Qval <- qvalue(pval)
pi0 <- Qval$pi0 # overall proportion of true null hypotheses

# Linear Reg
lm.fit <- lm(Qval$qvalues ~ BH)
lm.fit$coefficients

##      (Intercept)              BH
## -8.641789e-15   8.624009e-01
```

On comparison of slope to the π_0 estimate from `qval` we see that they are similar to each other. To answer this why this might be so, we need to look at the definitions of the terms itself.

p-value = most extreme probability for a test statistic under the null hypothesis, not accounting for multiple comparisons.. BH p-value or p_{BH} = most extreme probability for the above, after accounting for multiple comparisons such that there is an upper-bound to the overall false positive rate at $\leq p$. And finally, q-value is a direct estimate of the FDR associated with p_{BH} .

Now, as in the examples before and other course notes we see that q-values depend also on the estimated fraction of test p-values in the chance or uniform component of the distribution at some p_{FDR} p . Here,

π_0 = estimated probability (overall) of a given result being truly null (i.e., false positive) at p_{FDR} . And then, q-value = BH p-value * π_0 (probability that test t incorrectly rejects the null at p_{BH}).

So $q = p_{BH} * \pi_0$ as seen above in the example, and directly estimates the p_{FDR} for any given test t . In fact, going by the above regression equation and the explanation above, if π_0 is estimated as at or very near 1.0, then p_{BH} and q will be the same for any given test t .

Problem 3 – Solutions

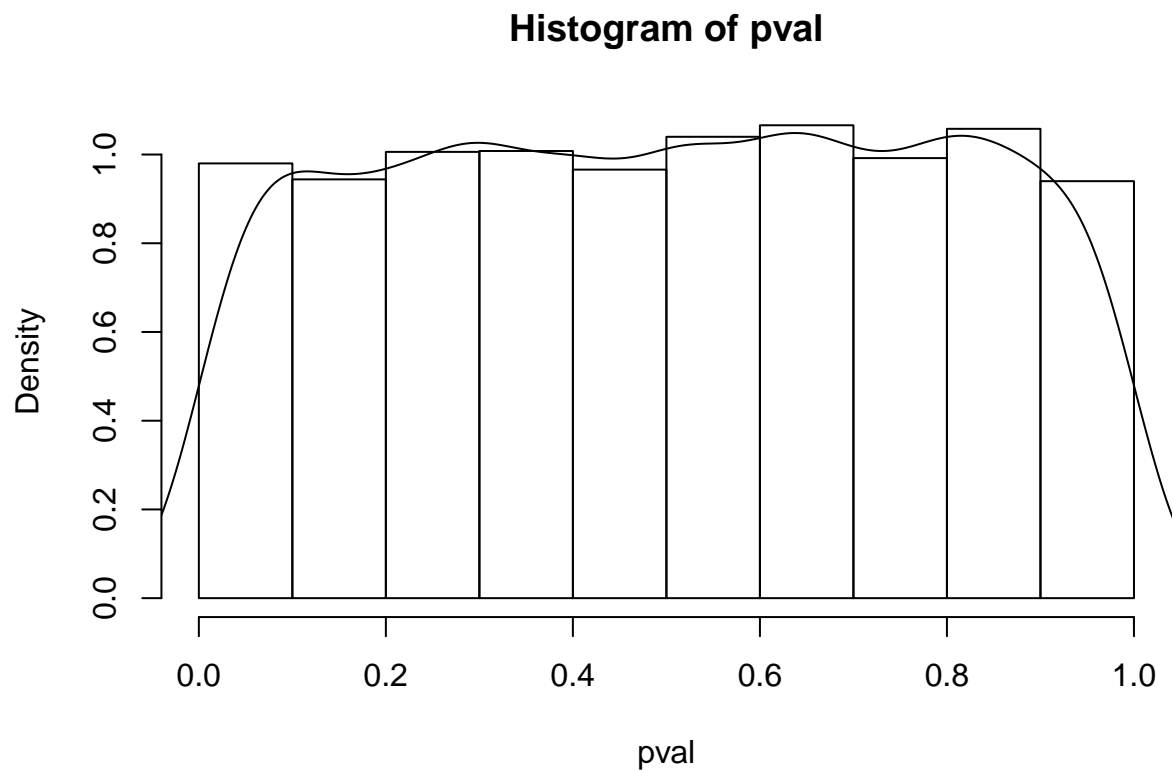
(a)

```

# Generate 5000 p-values
m = 500
p = 5000
#pval <- c(rbeta(m, b.1, b.0), runif(p-m, 0, 1))

b.1 = 1
b.0 = 1
pval <- c(rbeta(m, b.1, b.0), runif(p-m, 0, 1))
hist(pval, xlab = "pval", prob = TRUE)
lines(density(pval))

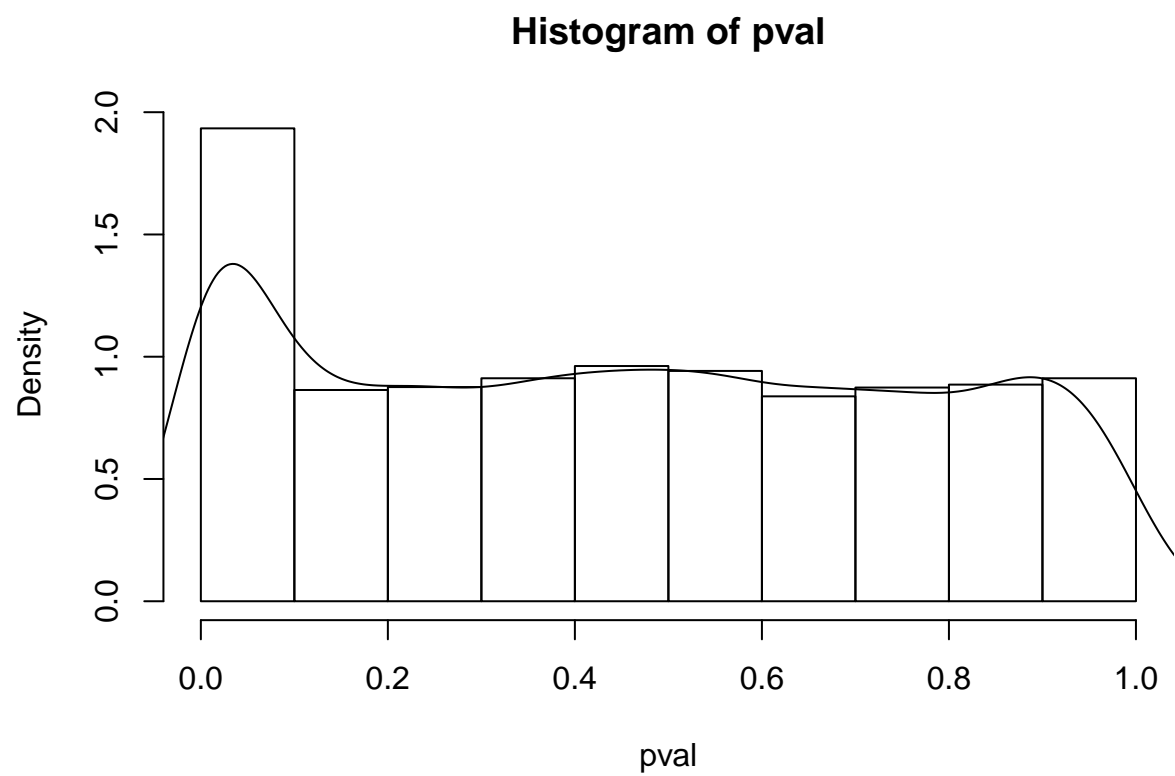
```



```

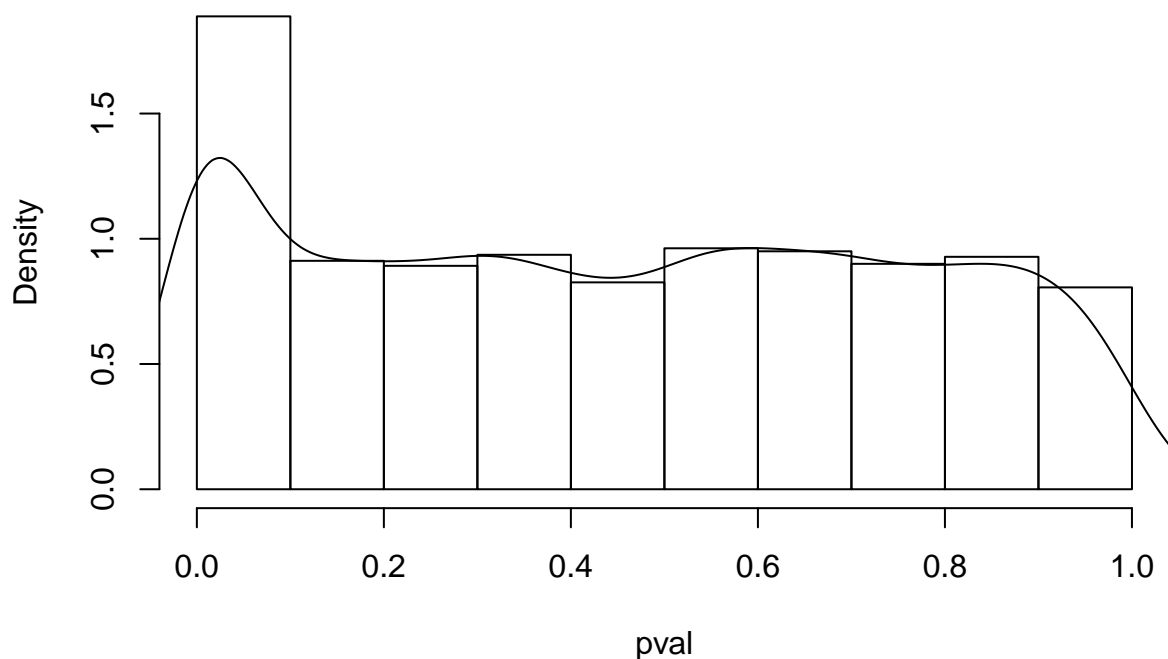
b.1 = 1
b.0 = 100
pval <- c(rbeta(m, b.1, b.0), runif(p-m, 0, 1))
hist(pval, xlab = "pval", prob = TRUE)
lines(density(pval))

```



```
b.1 = 1
b.0 = 500
pval <- c(rbeta(m, b.1, b.0), runif(p-m, 0, 1))
hist(pval, xlab = "pval", prob = TRUE)
lines(density(pval))
```

Histogram of pval



(b)

```
# Generate R = 500 replicates
R = 500
qval_fun <- function(x) {qvalue(p = x)}

#####

b.1 = 1
b.0 = 1

# Generate replicated pvalues and apply qval function over the columns
pval_new <- replicate(R, c(rbeta(m, b.1, b.0), runif(p-m, 0, 1)))
eff <- c(rep(T, m), rep(F, p - m))
qval <- lapply(1:ncol(pval_new), function(x) qvalue(pval_new[,x]))

# Extract variables/elements from the above list
qval_rep <- lapply(qval, `[`, c('qvalues'))
qval_rep_df <- data.frame(matrix(unlist(qval_rep), nrow=length(qval_rep), byrow=T))

pi0_rep <- lapply(qval, `[`, c('pi0'))
pi0_rep_df <- data.frame(matrix(unlist(pi0_rep), nrow=length(pi0_rep), byrow=T))
names(pi0_rep_df) <- names(pi0_rep[[which(lengths(pi0_rep)>0)[1]])

alpha=0.1
```

```

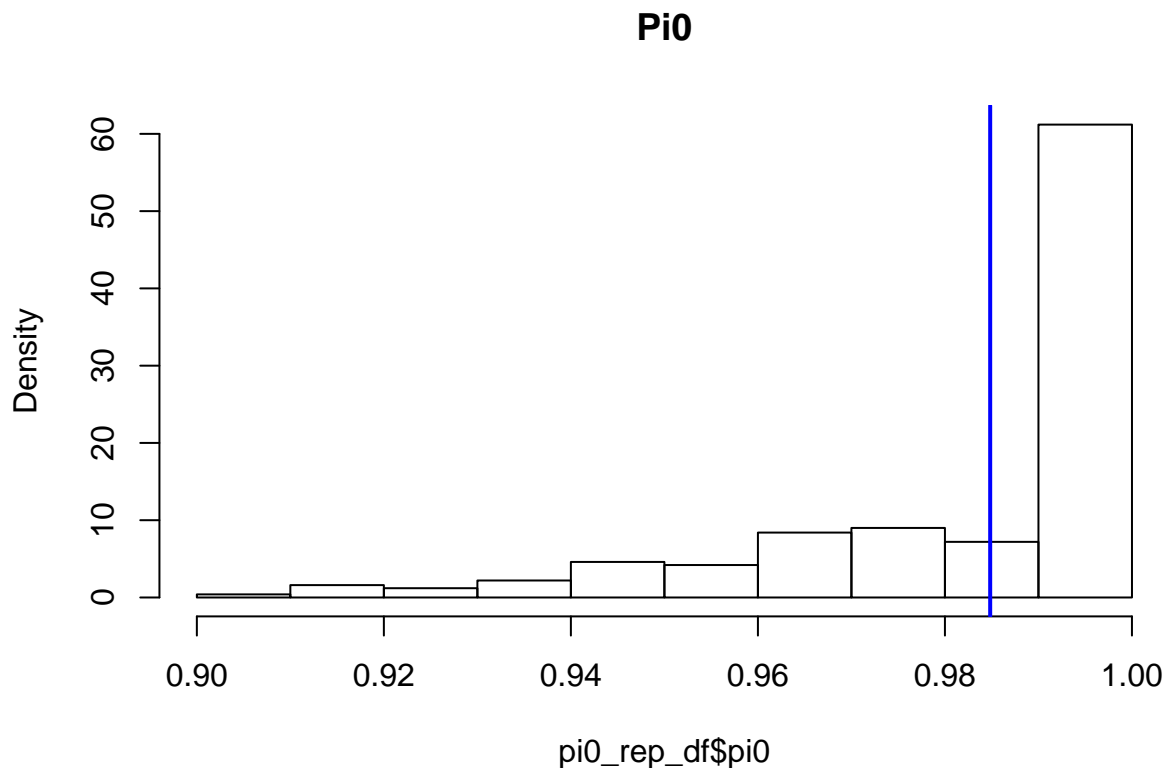
# Discoveries
D_func <- function(x) { (sum(qval_rep_df[x] < alpha)) }
Disc <- lapply(1:ncol(pval_new), D_func)
Disc_rep_df <- data.frame(matrix(unlist(Disc), nrow=length(Disc), byrow=T)) # Extract and generate df f
colnames(Disc_rep_df) <- "Discoveries"

# FDP
FD_func <- function(x) {sum(qval_rep_df[(m + 1):p] < alpha)}
FD <- lapply(1:ncol(pval_new), FD_func)
FD_rep_df <- data.frame(matrix(unlist(FD), nrow=length(FD), byrow=T))
colnames(FD_rep_df) <- "False Discoveries"
FDP <- as.data.frame(ifelse(Disc_rep_df$Discoveries == 0, 0, FD_rep_df$`False Discoveries`/Disc_rep_df$
colnames(FDP) <- "FD_Proportion"

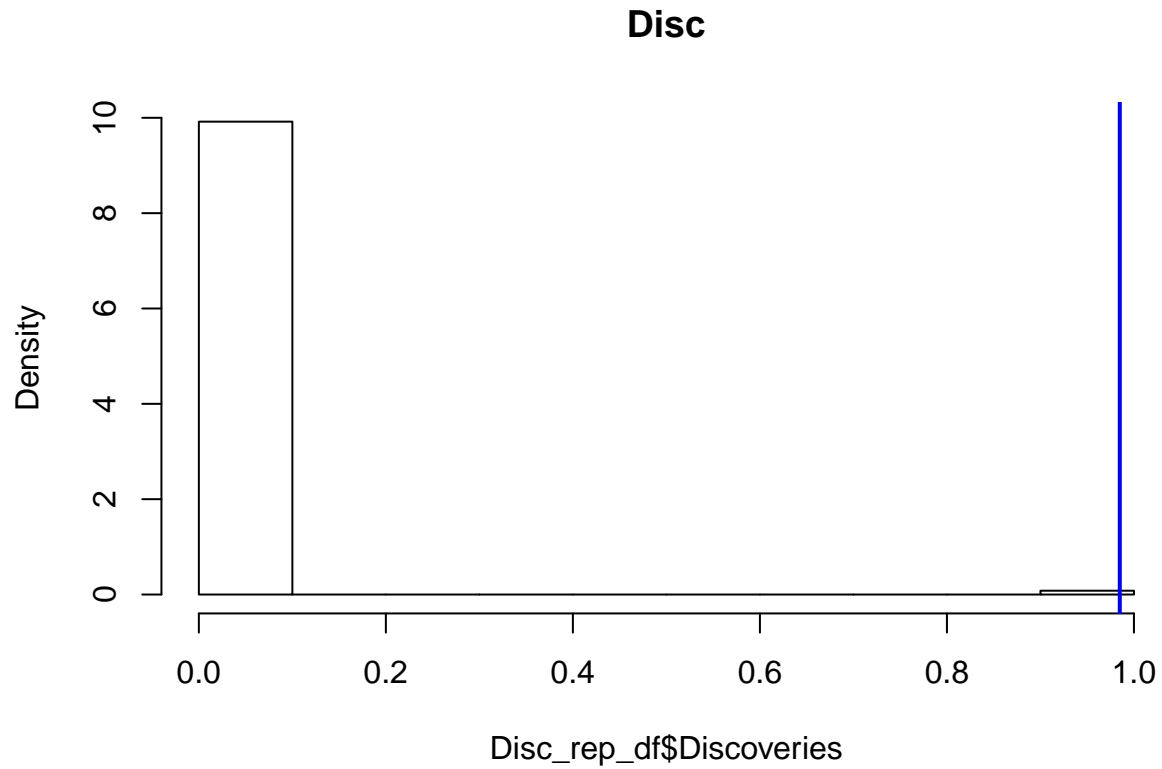
# Power
val <- which(Disc_rep_df$Discoveries < alpha)
Disc_corr <- sum(eff[val] == "TRUE")
pow <- Disc_corr/m

# Histograms
pi_mean <- mean(pi0_rep_df$pi0)
hist(pi0_rep_df$pi0, main = "Pi0", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)

```

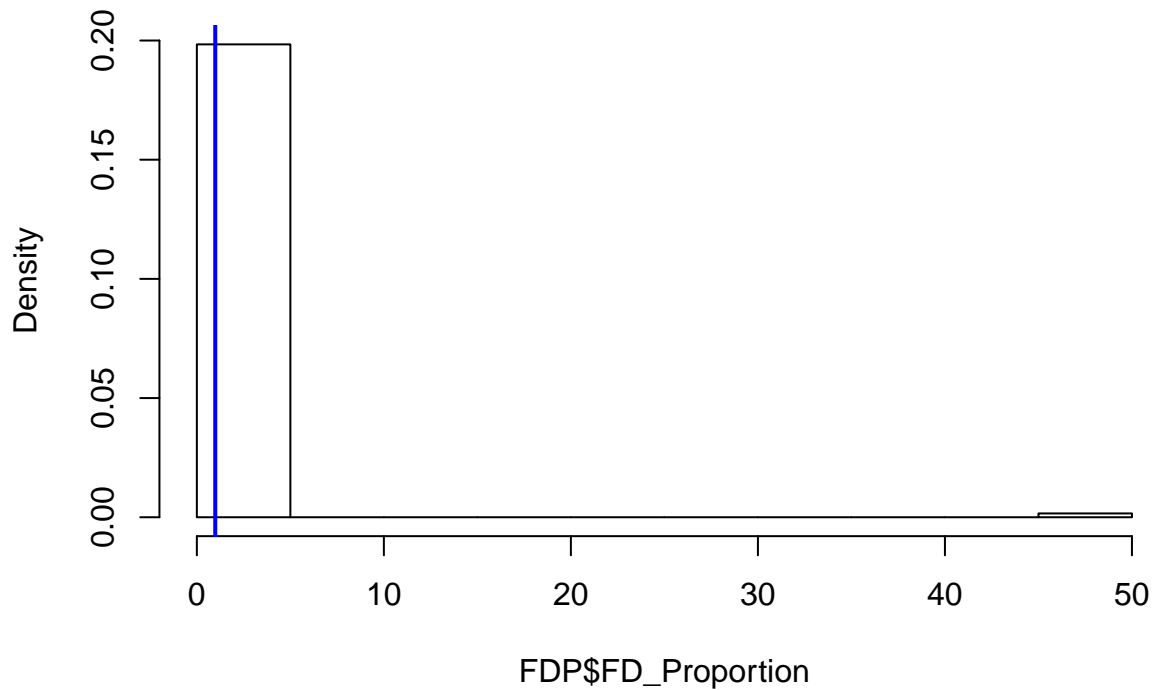


```
Disc_mean <- mean(Disc_rep_df$Discoveries)
hist(Disc_rep_df$Discoveries, main = "Disc", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)
```



```
FDP_mean <- mean(FDP$FD_Proportion)
hist(FDP$FD_Proportion, main = "FDP", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)
```

FDP



```
#####
b.1 = 1
b.0 = 100

# Generate replicated pvalues and apply qual function over the columns
pval_new <- replicate(R, c(rbeta(m, b.1, b.0), runif(p-m, 0, 1)))
eff <- c(rep(T, m), rep(F, p - m))
qval <- lapply(1:ncol(pval_new), function(x) qvalue(pval_new[,x]))

# Extract variables/elements from the above list
qval_rep <- lapply(qval, `[`, c('qvalues'))
qval_rep_df <- data.frame(matrix(unlist(qval_rep), nrow=length(qval_rep), byrow=T))

pi0_rep <- lapply(qval, `[`, c('pi0'))
pi0_rep_df <- data.frame(matrix(unlist(pi0_rep), nrow=length(pi0_rep), byrow=T))
names(pi0_rep_df) <- names(pi0_rep)[[which(lengths(pi0_rep)>0)[1]]]

alpha=0.1

# Discoveries
D_func <- function(x) { (sum(qval_rep_df[x] < alpha)) }
Disc <- lapply(1:ncol(pval_new), D_func)
Disc_rep_df <- data.frame(matrix(unlist(Disc), nrow=length(Disc), byrow=T))
colnames(Disc_rep_df) <- "Discoveries"
```

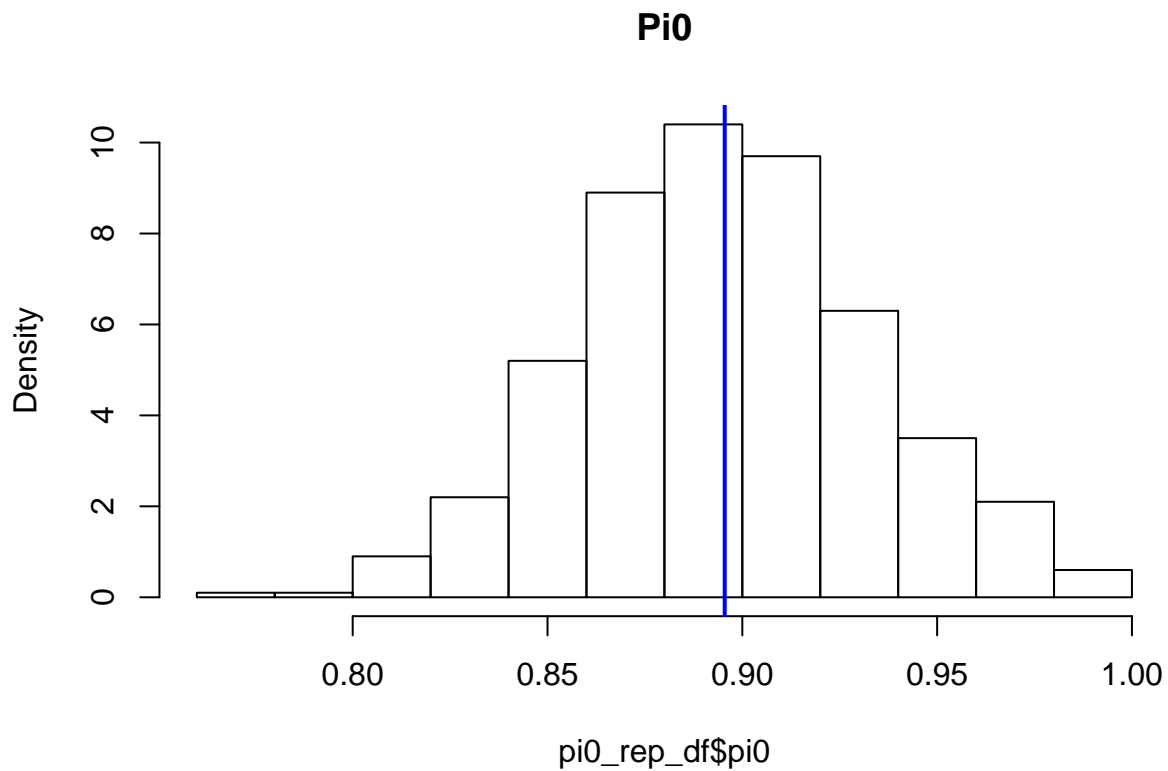
```

# FDP
FD_func <- function(x) {sum(qval_rep_df[(m + 1):p] < alpha)}
FD <- lapply(1:ncol(pval_new), FD_func)
FD_rep_df <- data.frame(matrix(unlist(FD), nrow=length(FD), byrow=T))
colnames(FD_rep_df) <- "False Discoveries"
FDP <- as.data.frame(ifelse(Disc_rep_df$Discoveries == 0, 0, FD_rep_df$`False Discoveries`/Disc_rep_df$Discoveries))
colnames(FDP) <- "FD_Proportion"

# Power
val <- which(Disc_rep_df$Discoveries < alpha)
Disc_corr <- sum(eff[val] == "TRUE")
pow <- Disc_corr/m

# Histograms
pi_mean <- mean(pi0_rep_df$pi0)
hist(pi0_rep_df$pi0, main = "Pi0", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)

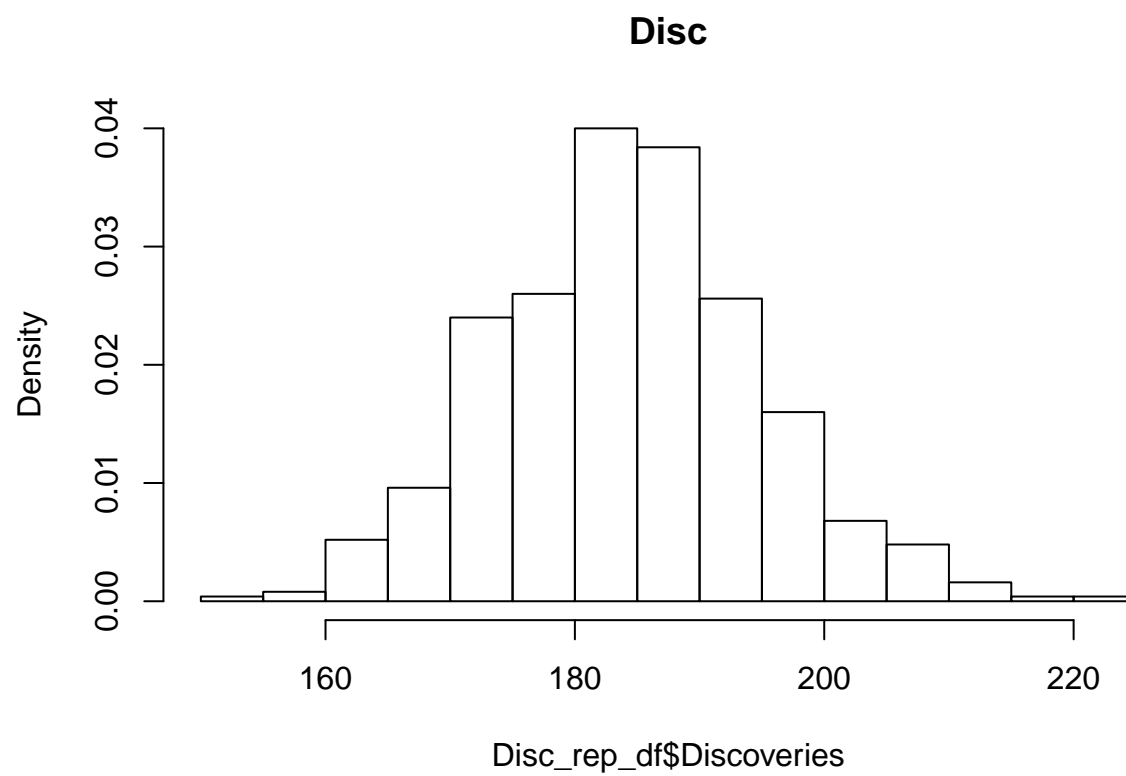
```



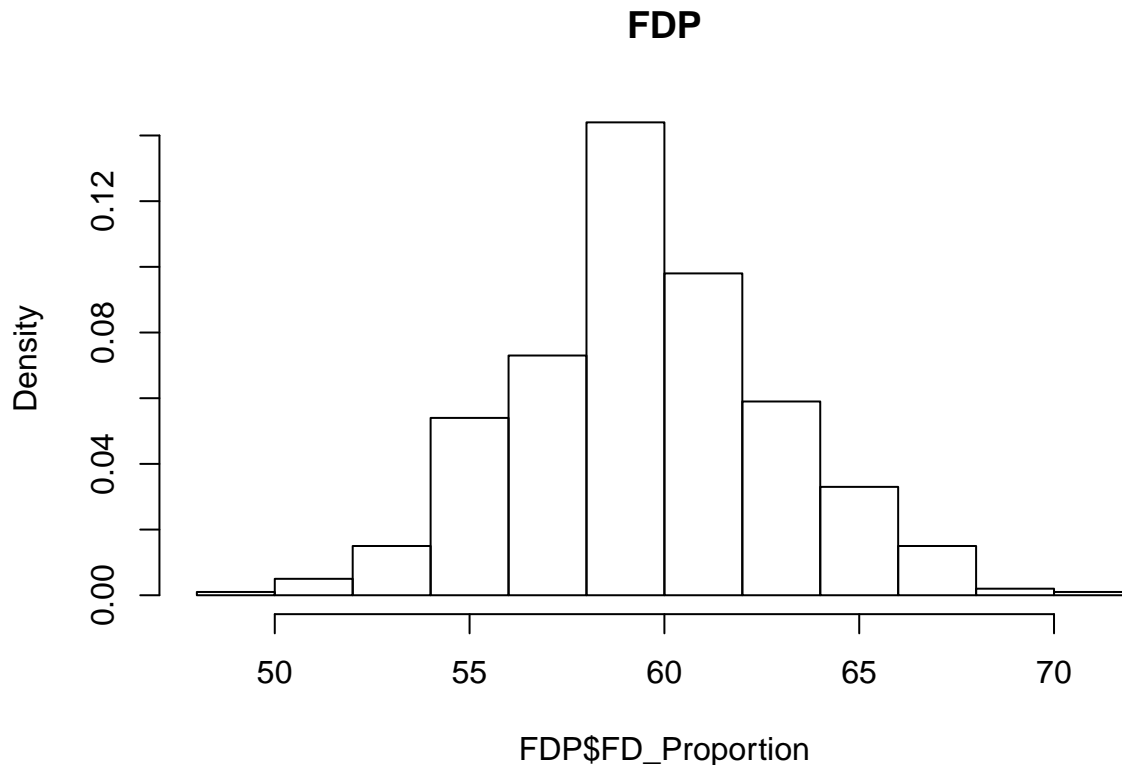
```

Disc_mean <- mean(Disc_rep_df$Discoveries)
hist(Disc_rep_df$Discoveries, main = "Disc", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)

```

```
FDP_mean <- mean(FDP$FD_Proportion)
hist(FDP$FD_Proportion, main = "FDP", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)
```



```
#####

b.1 = 1
b.0 = 500

# Generate replicated pvalues and apply qual function over the columns
pval_new <- replicate(R, c(rbeta(m, b.1, b.0), runif(p-m, 0, 1)))
eff <- c(rep(T, m), rep(F, p - m))
qval <- lapply(1:ncol(pval_new), function(x) qvalue(pval_new[,x]))

# Extract variables/elements from the above list
qval_rep <- lapply(qval, `[`, c('qvalues'))
qval_rep_df <- data.frame(matrix(unlist(qval_rep), nrow=length(qval_rep), byrow=T))

pi0_rep <- lapply(qval, `[`, c('pi0'))
pi0_rep_df <- data.frame(matrix(unlist(pi0_rep), nrow=length(pi0_rep), byrow=T))
names(pi0_rep_df) <- names(pi0_rep)[[which(lengths(pi0_rep)>0)[1]]]

alpha=0.1

# Discoveries
D_func <- function(x) { (sum(qval_rep_df[x] < alpha)) }
Disc <- lapply(1:ncol(pval_new), D_func)
Disc_rep_df <- data.frame(matrix(unlist(Disc), nrow=length(Disc), byrow=T))
colnames(Disc_rep_df) <- "Discoveries"
```

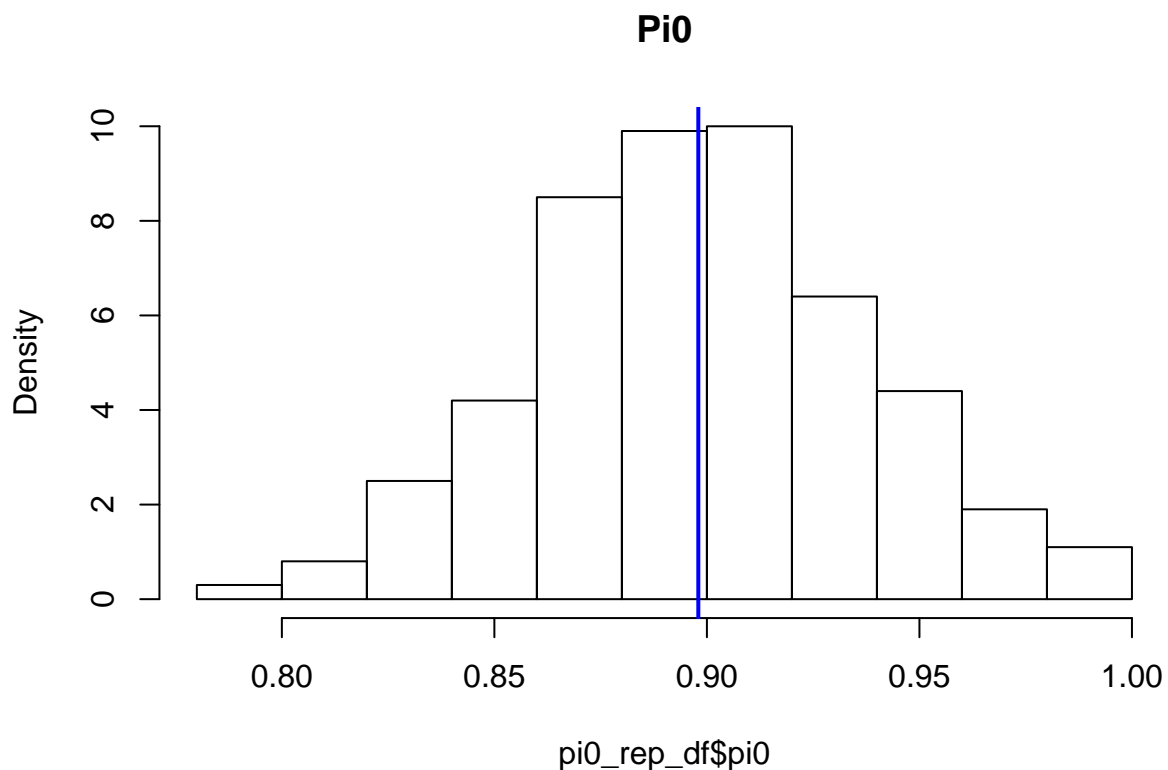
```

# FDP
FD_func <- function(x) {sum(qval_rep_df[(m + 1):p] < alpha)}
FD <- lapply(1:ncol(pval_new), FD_func)
FD_rep_df <- data.frame(matrix(unlist(FD), nrow=length(FD), byrow=T))
colnames(FD_rep_df) <- "False Discoveries"
FDP <- as.data.frame(ifelse(Disc_rep_df$Discoveries == 0, 0, FD_rep_df$`False Discoveries`/Disc_rep_df$Discoveries))
colnames(FDP) <- "FD_Proportion"

# Power
val <- which(Disc_rep_df$Discoveries < alpha)
Disc_corr <- sum(eff[val] == "TRUE")
pow <- Disc_corr/m

# Histograms
pi_mean <- mean(pi0_rep_df$pi0)
hist(pi0_rep_df$pi0, main = "Pi0", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)

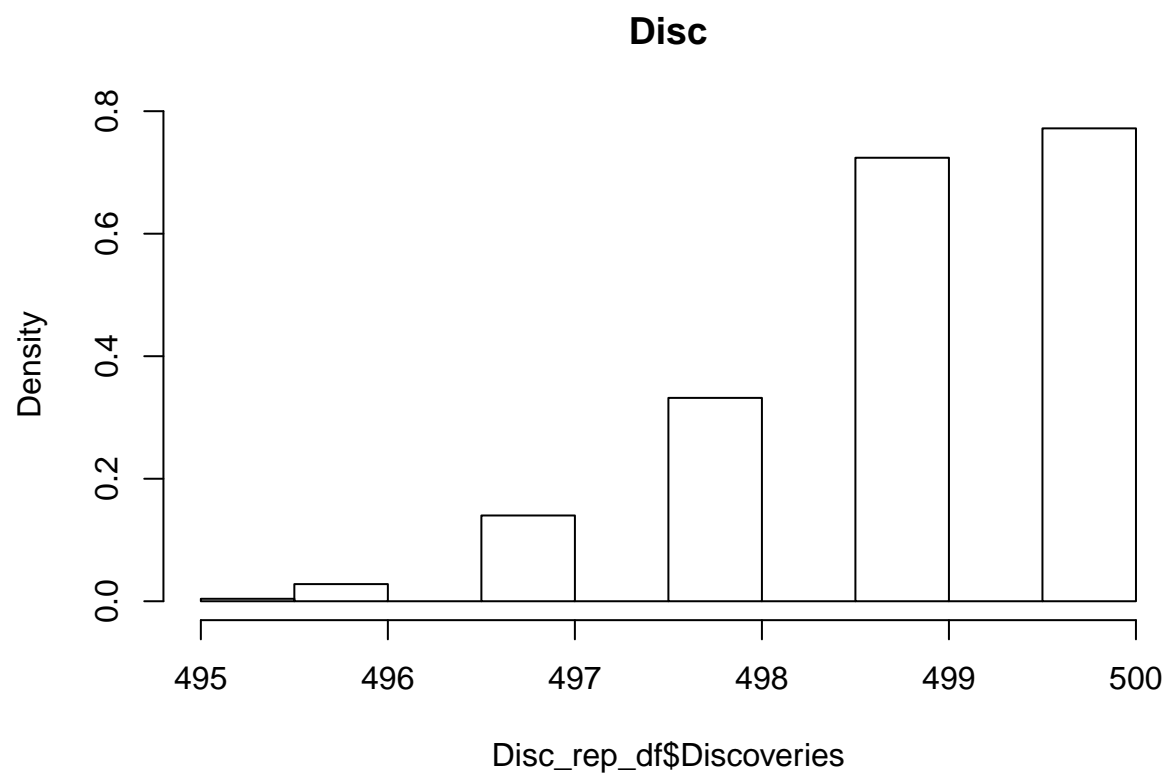
```



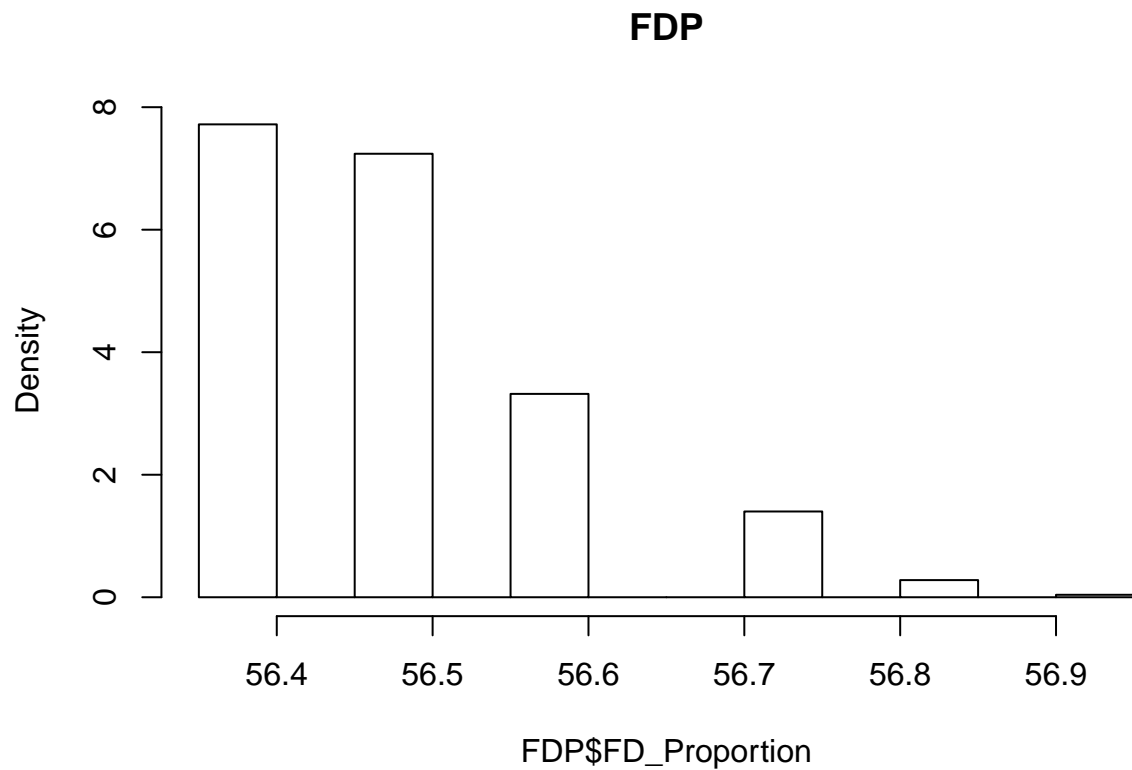
```

Disc_mean <- mean(Disc_rep_df$Discoveries)
hist(Disc_rep_df$Discoveries, main = "Disc", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)

```



```
FDP_mean <- mean(FDP$FD_Proportion)
hist(FDP$FD_Proportion, main = "FDP", prob = T)
abline(v = pi_mean, col = "blue", lwd = 2)
```



Problem 4 – Solutions

(a)

```
# For 1 coin toss
```

```
p <- 1
```

```
# Fair coin
```

```
null_prob <- 0.5
```

```
# possible number of heads after 2 tosses
```

```
y <- c(0, 1, 2)
```

```
# Null hypothesis for 3 possible outcomes aka heads in the tosses
```

```
null_prob <- dbinom(y, 2, null_prob)
```

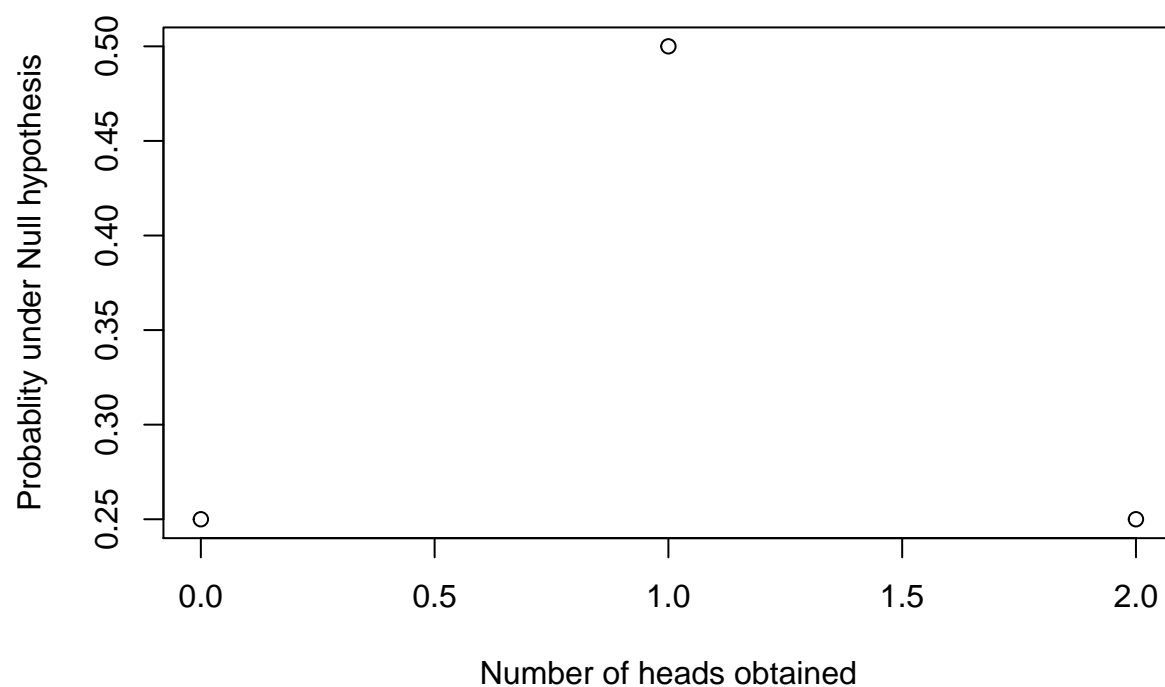
```
null_prob
```

```
## [1] 0.25 0.50 0.25
```

```
# Null distribution of outcome v/s the number of heads obtained
```

```
plot(y, null_prob, type='p', main = "Null distribution of outcome from a single coin tossed 2 times", xlab = "Number of heads", ylab = "Probability")
```

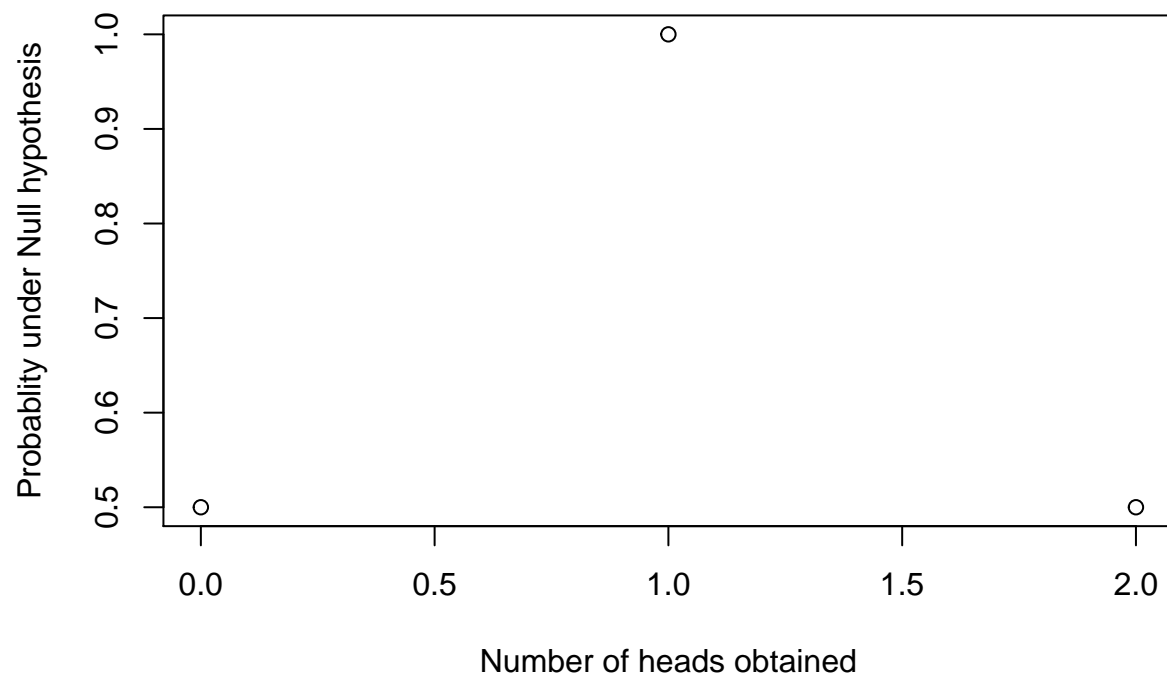
Null distribution of outcome from a single coin tossed 2 times



```
# Null distribution of p-values from tossing coin twice, assuming its a fair coin
null_p_distr <- c(0.5, 1, 0.5)

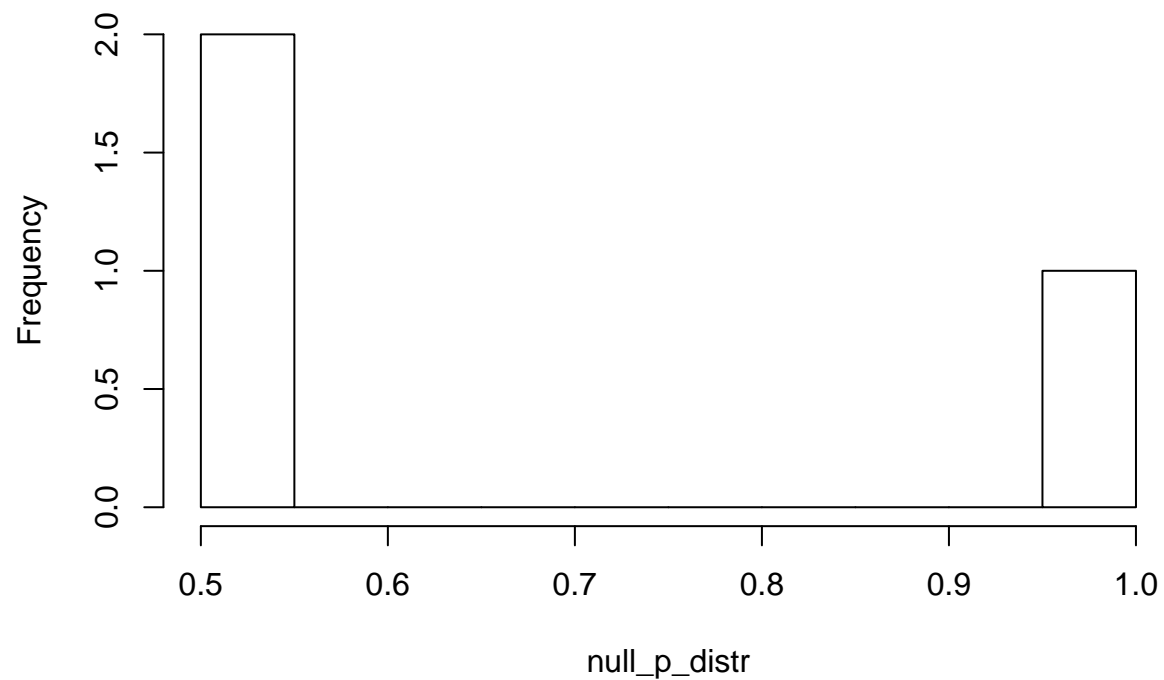
# Null distribution v/s the number of heads obtained
plot(y, null_p_distr, type='p', main = "Null distribution of p-values from tossing coin twice", xlab =
```

Null distribution of p-values from tossing coin twice



```
# Histogram of the p-values  
hist(null_p_distr, 10, main="Null distribution of p-values from tossing coin twice, assuming its a fair
```

ull distribution of p-values from tossing coin twice, assuming its a fair



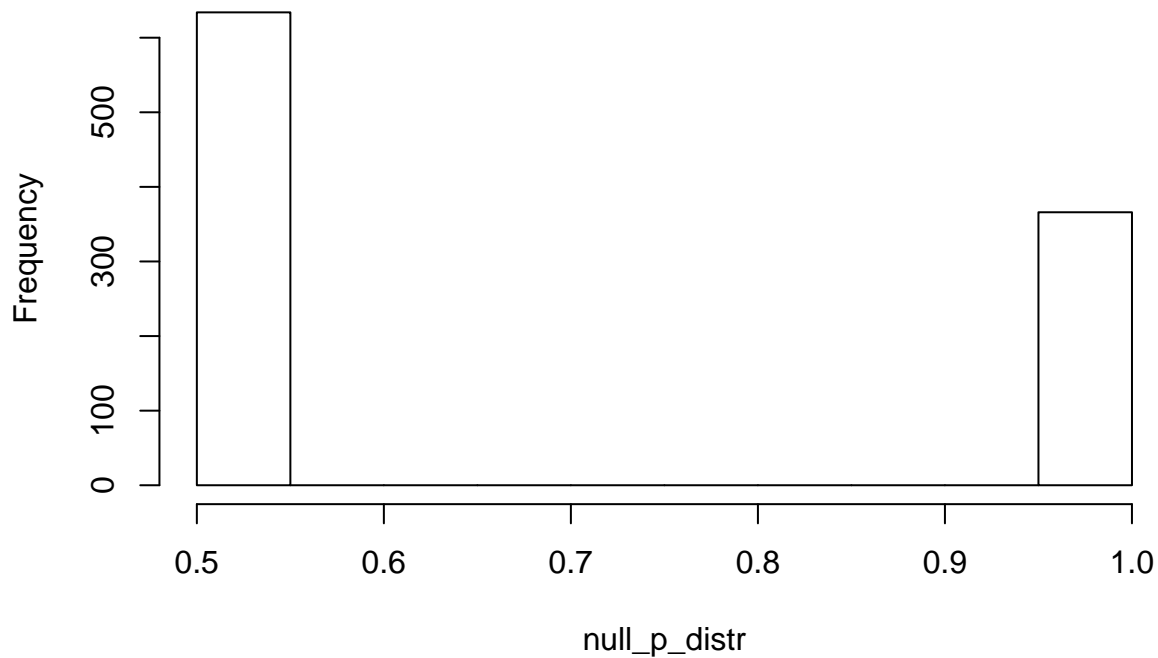
(b)

```
# For 1000 coins
p <- 1000

# Observed counts of outcomes
y <- c(180, 366, 454)

# Distribution of tp-values for the above 1000 coins
null_p_distr <- rep(c(0.5, 1, 0.5), y)
hist(null_p_distr, 10, main="Null distribution of p-values")
```


Null distribution of p-values



From the null distribution from part(a) of the question we see that the prob of 1 head is 1 as this is

*# $Pi0 = (2 * \text{Fair coins}) / \text{Total coins}$*

In the above example, estimate of $Pi0$ would be

```
Pi0_est <- (2*366)/p
Pi0_est
```

```
## [1] 0.732
```

(c)

Given values of lambda

```
lambda <- c(0.4, 0.5, 0.9)
```

Estimate of $Pi0$

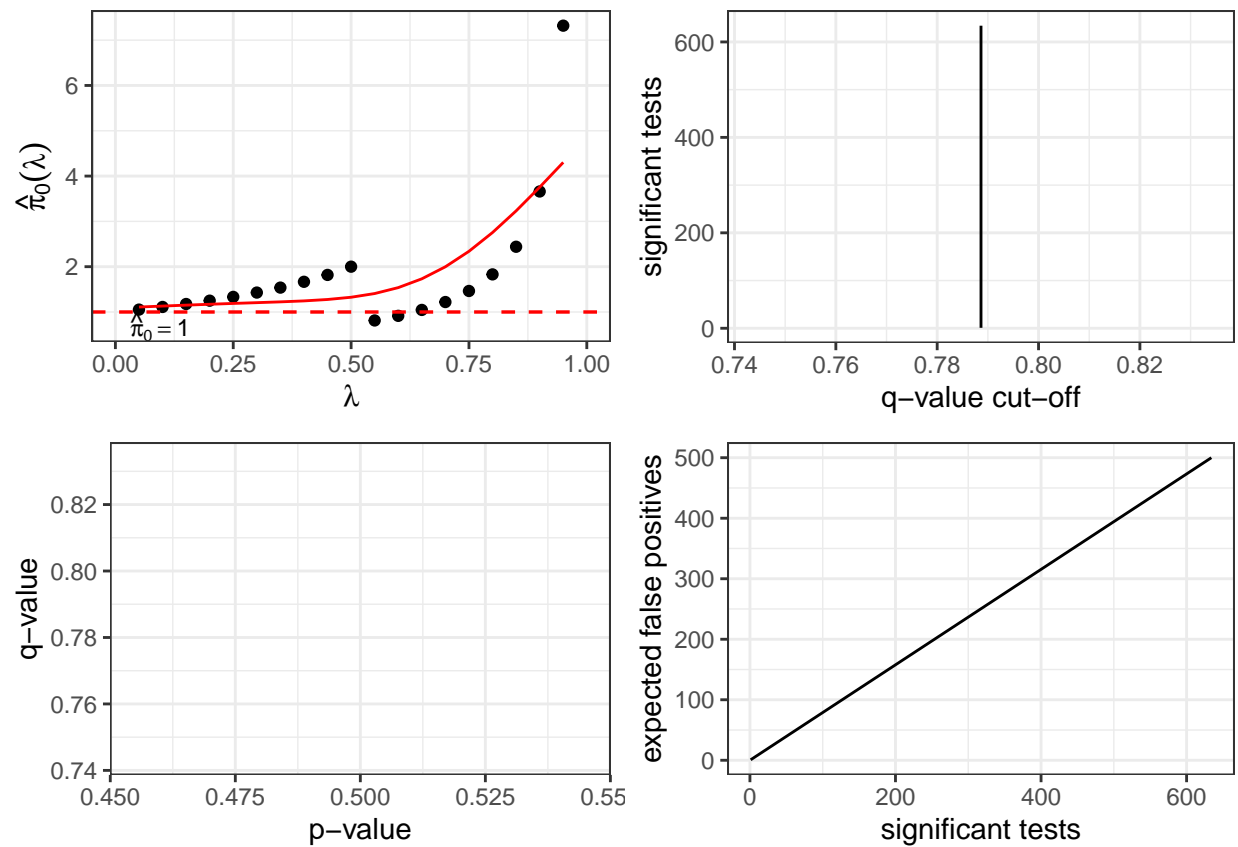
```
Pi0_est <- as.data.frame(sapply(lambda,function(x){sum(null_p_distr > x)})/p/(1 - lambda))
colnames(Pi0_est) <- "lambda"
```

```
View(Pi0_est)
```

Only for lambda = 0.5 estimate agrees with part(a).

#Q-values

```
qVal <- qvalue(null_p_distr)
plot(qVal)
```



Q-values dont seem to work with discrete data at all, as Q-value for obsevation whose pvalue=0.5 is around 0.79. This value is higher than that we got in (b)