R training

Zhuyu Qiu

11/12/2019

Question 1

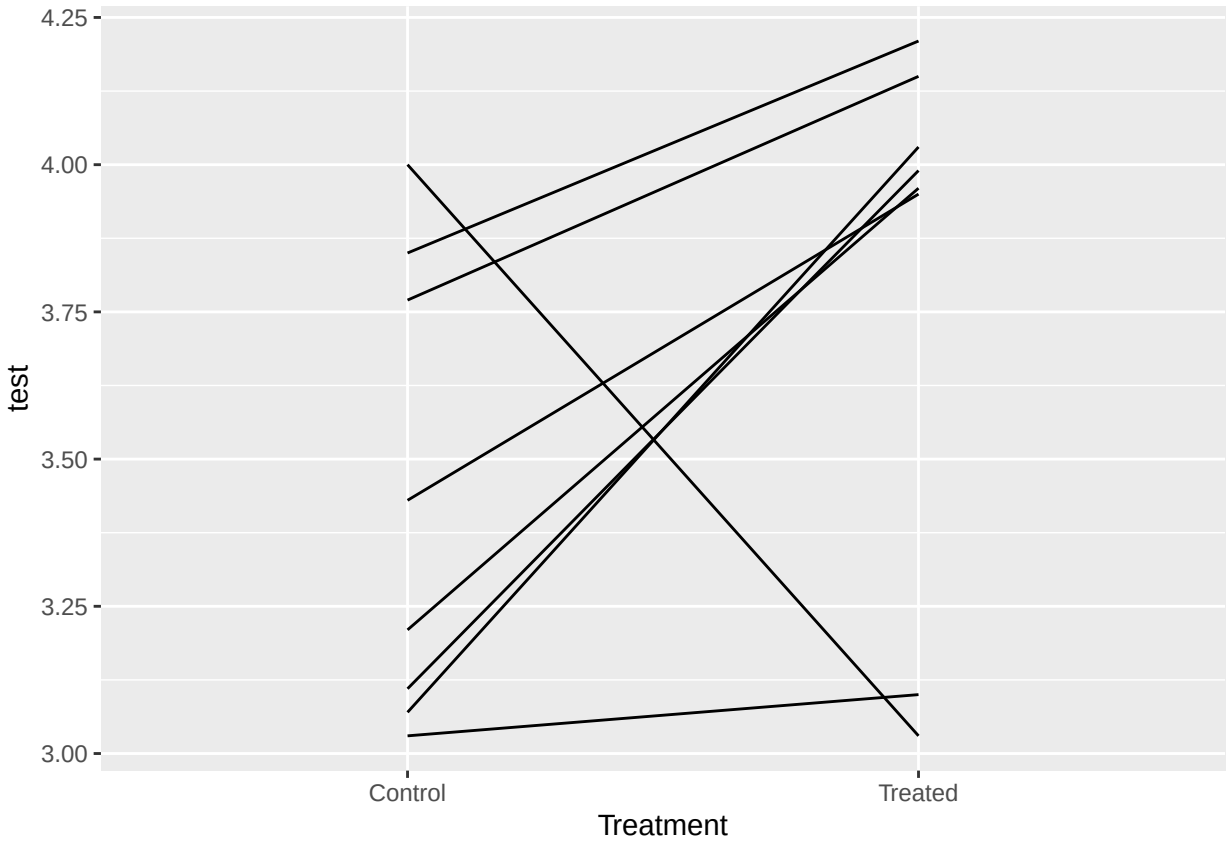
1.Read the file AHR-test-file.txt

```
ahr.test = read.table("/cloud/project/AHR-test-file.txt", header=T);
summary(ahr.test);
```

```
##     Control        Treated
##  Min.   :3.03   Min.   :3.030
##  1st Qu.:3.11   1st Qu.:3.737
##  Median :3.21   Median :3.975
##  Mean   :3.40   Mean   :3.803
##  3rd Qu.:3.77   3rd Qu.:4.060
##  Max.   :4.00   Max.   :4.210
##                 NA's   :1
```

```
#convert data to long format
ahr.test$number = paste("mouse", 1:9, sep = "");
ahr.test.long=gather(ahr.test, Treatment, test, Control:Treated,
                     factor_key=TRUE);

#plot induvidual change curve
ggplot(data=ahr.test.long, aes(x=Treatment, y=test, group=number)) +
  geom_line();
```

```
## Warning: Removed 1 rows containing missing values (geom_path).
```



2.Perform a t-test between control and treated

```
# paired t-test
t.test(ahr.test$Control, ahr.test$Treated, paired = T);
```

```
##
##  Paired t-test
##
## data:  ahr.test$Control and ahr.test$Treated
## t = -1.6917, df = 7, p-value = 0.1345
## alternative hypothesis: true difference in means is not equal to 0
## 95 percent confidence interval:
##  -0.8841939  0.1466939
## sample estimates:
## mean of the differences
##                -0.36875
```

Paired t-test gave the result that the p-value=0.1345>0.05, therefore, we failed to reject the null at the significance level of 0.05. And we can conclude that there is no significant difference between the Control group and the Treated group

3. Perform a wilcoxon test between control and treated

```
# wilconxon signed-rank test, asssume an underlying continuous symmetric distribution
wilcox.test(ahr.test$Control, ahr.test$Treated, paired = T);
```

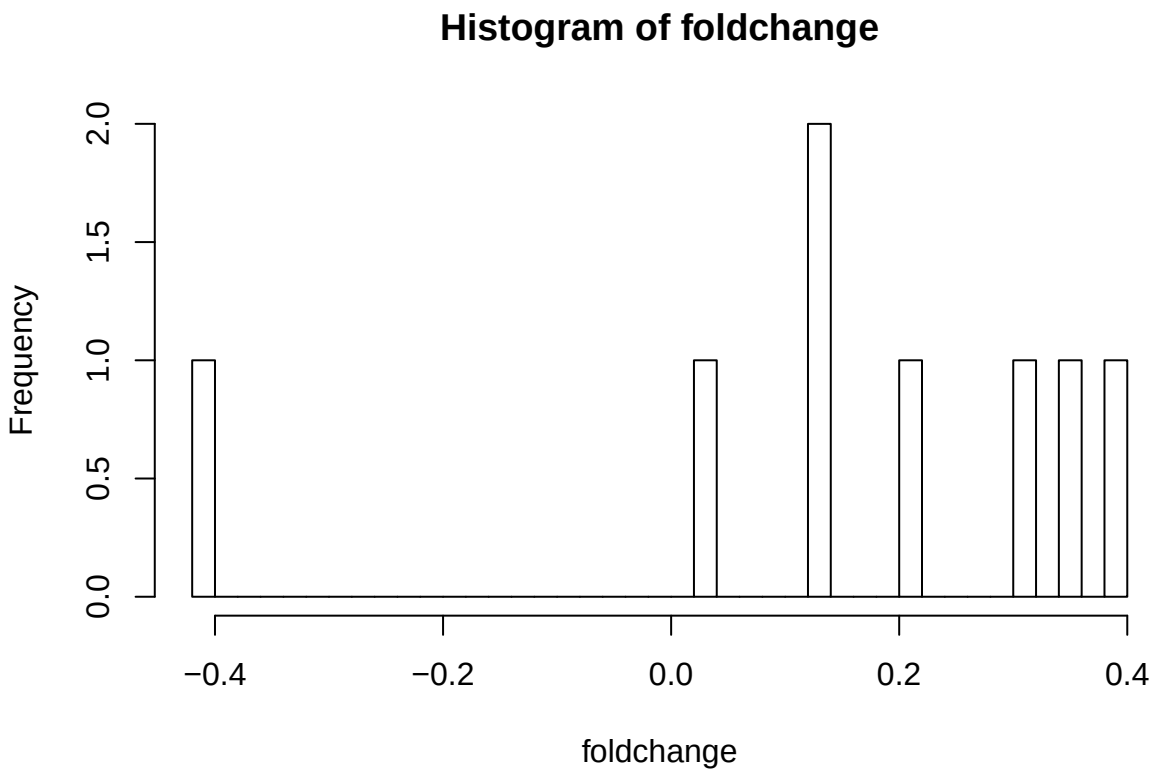
```
##
```

```
##  Wilcoxon signed rank test
##
## data:  ahr.test$Control and ahr.test$Treated
## V = 8, p-value = 0.1953
## alternative hypothesis: true location shift is not equal to 0
```

Wilcoxon test gave the result that the p-value=0.1953>0.05, which also suggests there is no significant difference between the Control group and the Treated group.

4. Calculate a fold-change between control and treated

```
foldchange = log2(ahr.test$Treated/ahr.test$Control);
hist(foldchange,breaks = 50);
```



The log2 fold change result showed that most mouses showed an AHR increasement after being treated and only 1 mouse had decreased.

- Theory

1. different type of ttest 1 sample/ paired/ 2 sample(equal variance? var.test(x,y))
2. different two sample tests paired t test/Wilcoxon signed-rank test(median) 2 sample t test/Wilcoxon rank-sum test(median) McNemar's Test(two sample test for binomial proportions for matched-pair data)
3. Which test to use and when t-test follows the asumption that the sample follows normal distribution,when the assumption is violated, like the sample size is too small,use non-parametric tests.

Question 2

1.Read the files

```
tumor1 = read.table("/cloud/project/input1.txt", header=T);
tumor2 = read.table("/cloud/project/input2.txt", header=T);
```

2.Combine two files

```
# 2.1 Sort the data and cbind
tumor1.sort = arrange(tumor1,GeneID);
tumor2.sort = arrange(tumor2,GeneID);
tumor.sort.cbind = cbind(tumor1.sort, tumor2.sort);
tumor.sort.cbind = tumor.sort.cbind[,-5];

#2.2 Merge
tumor.merge = merge(tumor1, tumor2, all=T);

#2.3 Verify if they get the same results
table(tumor.sort.cbind == tumor.merge, useNA = 'ifany');
```

```
##
## TRUE
## 6500
```

After removing the repeated gene ID colums in the sort&cbind way, the two ways gave the same results.

3. Perform a t-test comparing the first three tumours to the last nine tumours for each gene using a for-loop

```
### Function 1
#Input variables:
#number of row, number of columns, column data type, and column names
#Output variables:
#empty dataset
#Description:
#Function that create empty dataframe

emptydf = function(numrow, numcol, type, name){
  df = data.frame(matrix(NA, nrow=numrow, ncol=numcol));
  for (i in 1:numcol){
    print(type[i])
    if('numeric' == type[i]) {df[,i] = as.numeric(df[,i])
    colnames(df)[i] = name[i]};
    if('character' == type[i]) {df[,i] = as.character(df[,i])
    colnames(df)[i] = name[i]};
    if('logical' == type[i]) {df[,i] = as.logical(df[,i])
    colnames(df)[i] = name[i]};
    if('factor' == type[i]) {df[,i] = as.factor(df[,i])
    colnames(df)[i] = name[i]};
  }
  return(df);
}
```

```
# create empty dataframe to put p values
tumor.ttest = emptydf(500, 2, c('character','numeric'), c('GeneID', 'pvalue'));
```

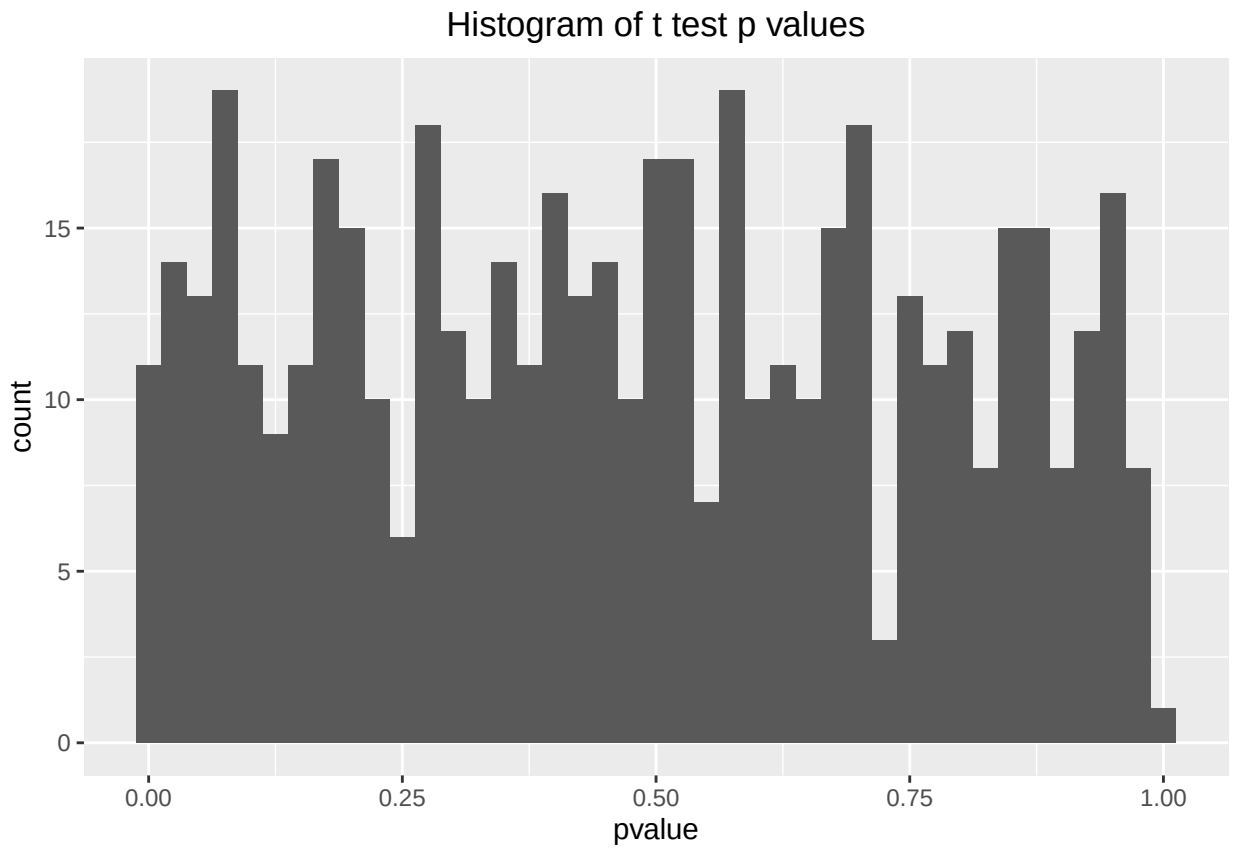
```
## [1] "character"
```

```
## [1] "numeric"
```

```
tumor.ttest$GeneID=tumor.merge$GeneID;

# perform ttest
type.a = c("Patient1","Patient2","Patient3");
type.b = c(paste0("Patient", 4:12));

for (i in 1:nrow(tumor.merge)){
  tumor.ttest[i,2] = tidy(t.test(tumor.merge[i,type.a],tumor.merge[i,type.b]))$p.value;
}
```

4. Histogram of the p values

```
ggplot(tumor.ttest,aes(pvalue)) +
  geom_histogram(binwidth = 0.025)+
  ggtitle("Histogram of t test p values")+
  theme(plot.title = element_text(hjust=0.5));
```



The histogram shows a sparse distribution of p values.

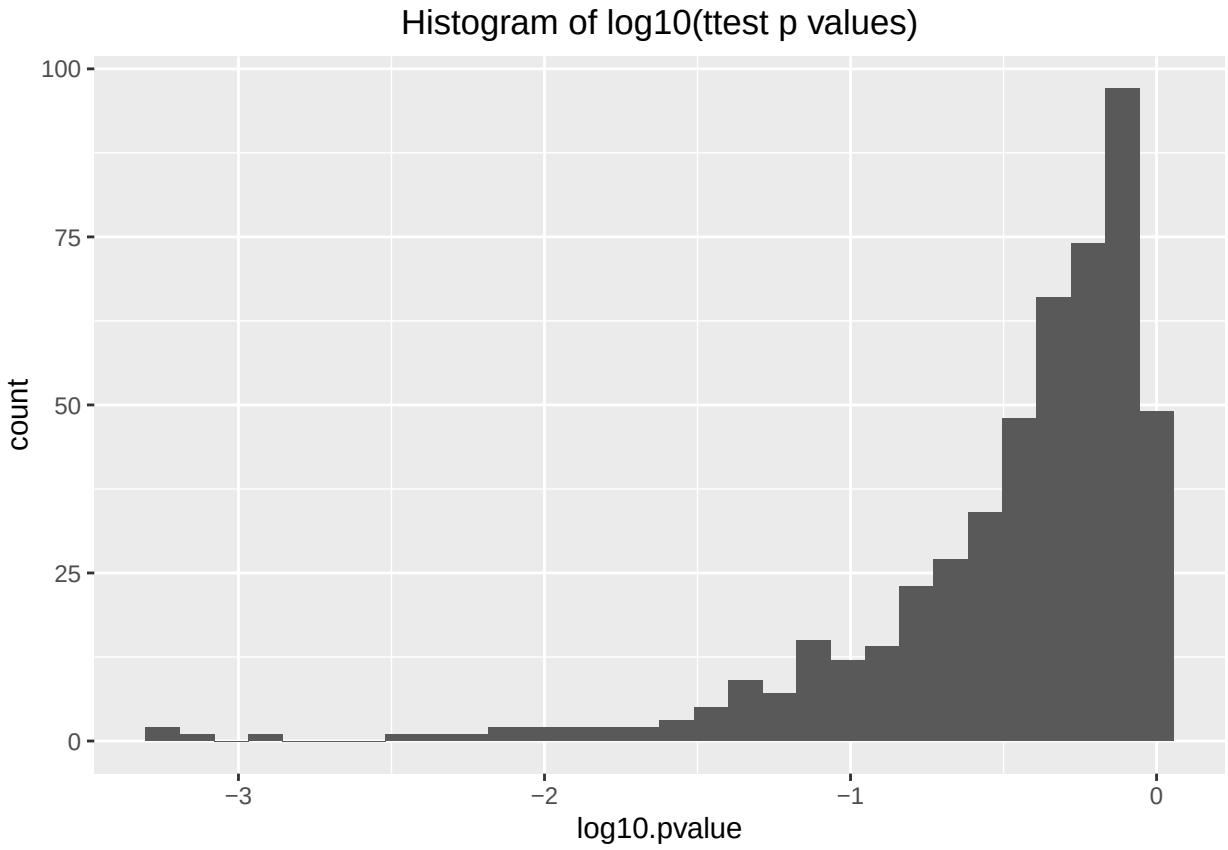
5. It's not rotated

6. Plot pvalues in log space

```
tumor.ttest$log10.pvalue = log10(tumor.ttest$pvalue);
ggplot(tumor.ttest, aes(log10.pvalue)) +
```

```
  geom_histogram()+
  ggtitle("Histogram of log10(ttest p values)")+
  theme(plot.title = element_text(hjust=0.5));
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



7. What does the distribution tell you

Since log(0.05)=-3, from the plot we can find out that most p values from the ttest are greater than 0.05. Therefore, among the 500 genes, most genes don't have a significant mRNA levels difference between Tumor type A and type B. Only 33 of them show a significant mRNA levels difference.

```
nrow(tumor.ttest[tumor.ttest$pvalue<=0.05,]);
```

```
## [1] 33
```

Question 3

1.Wilcoxon test

```
# create empty dataframe
tumor.wilcoxon = emptydf(500, 2, c('character','numeric'), c('GeneID', 'pvalue'));
```

```
## [1] "character"
## [1] "numeric"
```

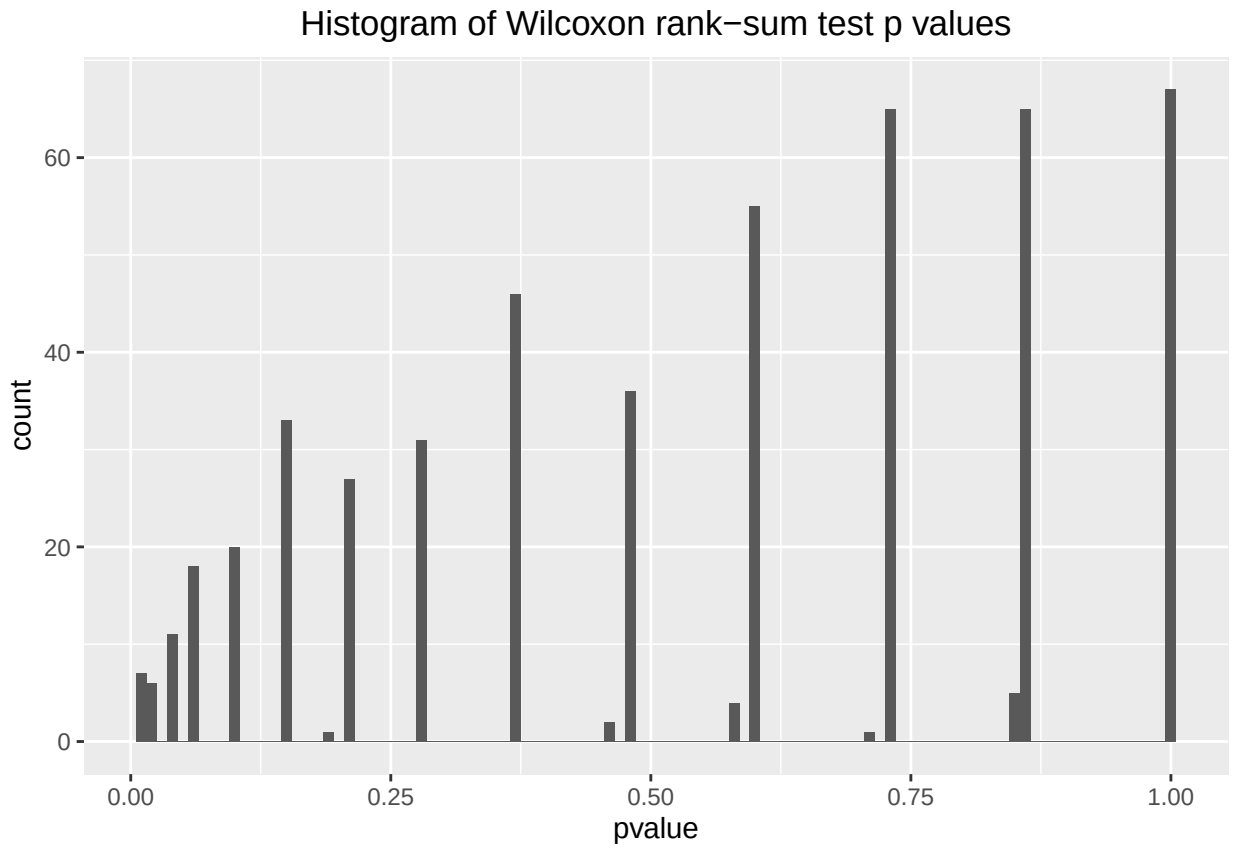
```
tumor.wilcoxon$GeneID = tumor.merge$GeneID;
```

```r
# perform wilcoxon rank sum test
for (i in 1:nrow(tumor.merge)){
  tumor.wilcoxon[i,2] = tidy(wilcox.test(unlist(tumor.merge[i,type.a]), unlist(tumor.merge[i,type.b]),
                          alternative = "two.sided"))$p.value;
}
```

```
## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties

## Warning in wilcox.test.default(unlist(tumor.merge[i, type.a]),
## unlist(tumor.merge[i, : cannot compute exact p-value with ties
```

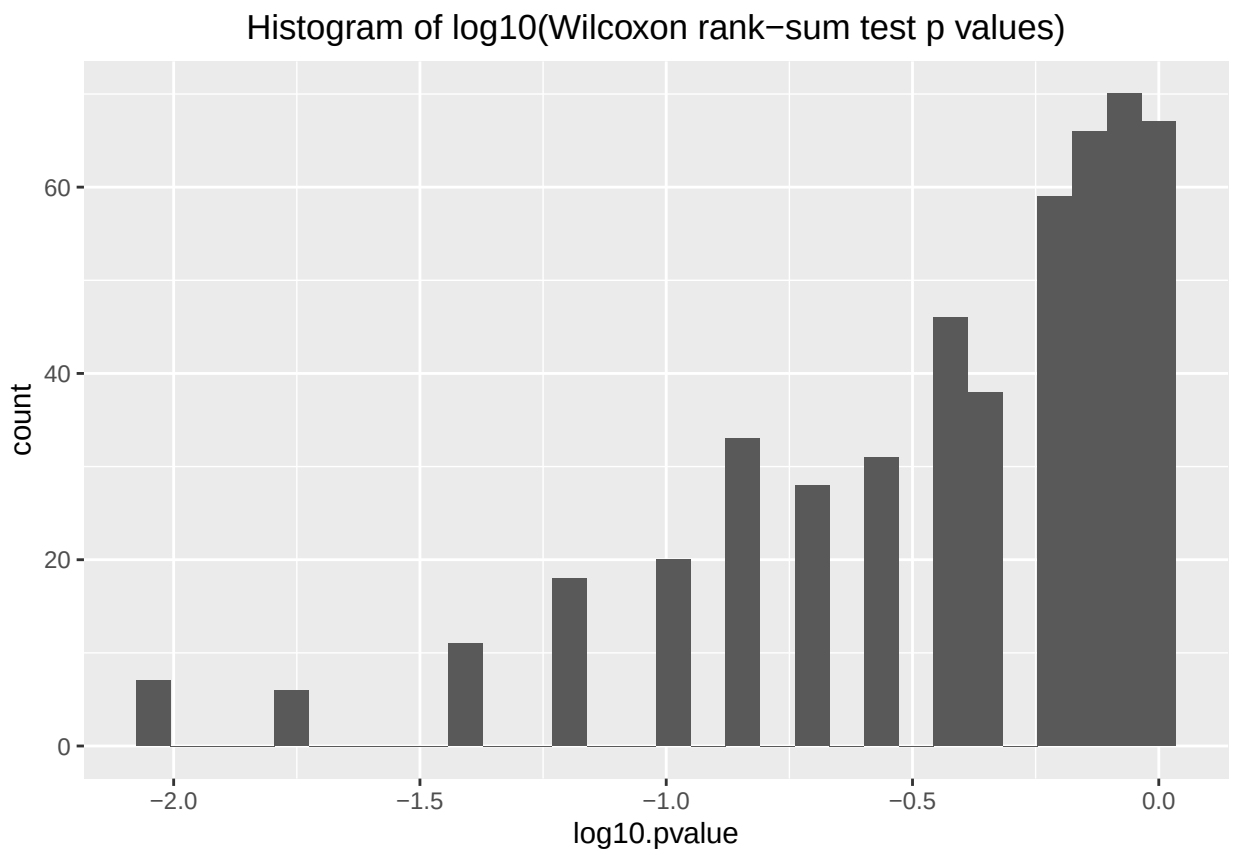
```r
# 14 warnings show there are tied data

# plot histogram
ggplot(tumor.wilcoxon,aes(pvalue)) +
  geom_histogram(binwidth = 0.01)+
  ggtitle("Histogram of Wilcoxon rank-sum test p values")+
```

```
  theme(plot.title = element_text(hjust=0.5));
```




```
#log transform the p values and plot the histogram
tumor.wilcoxon$log10.pvalue = log10(tumor.wilcoxon$pvalue);

ggplot(tumor.wilcoxon, aes(log10.pvalue)) +
  geom_histogram()+
  ggtitle("Histogram of log10(Wilcoxon rank-sum test p values)")+
  theme(plot.title = element_text(hjust=0.5));
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

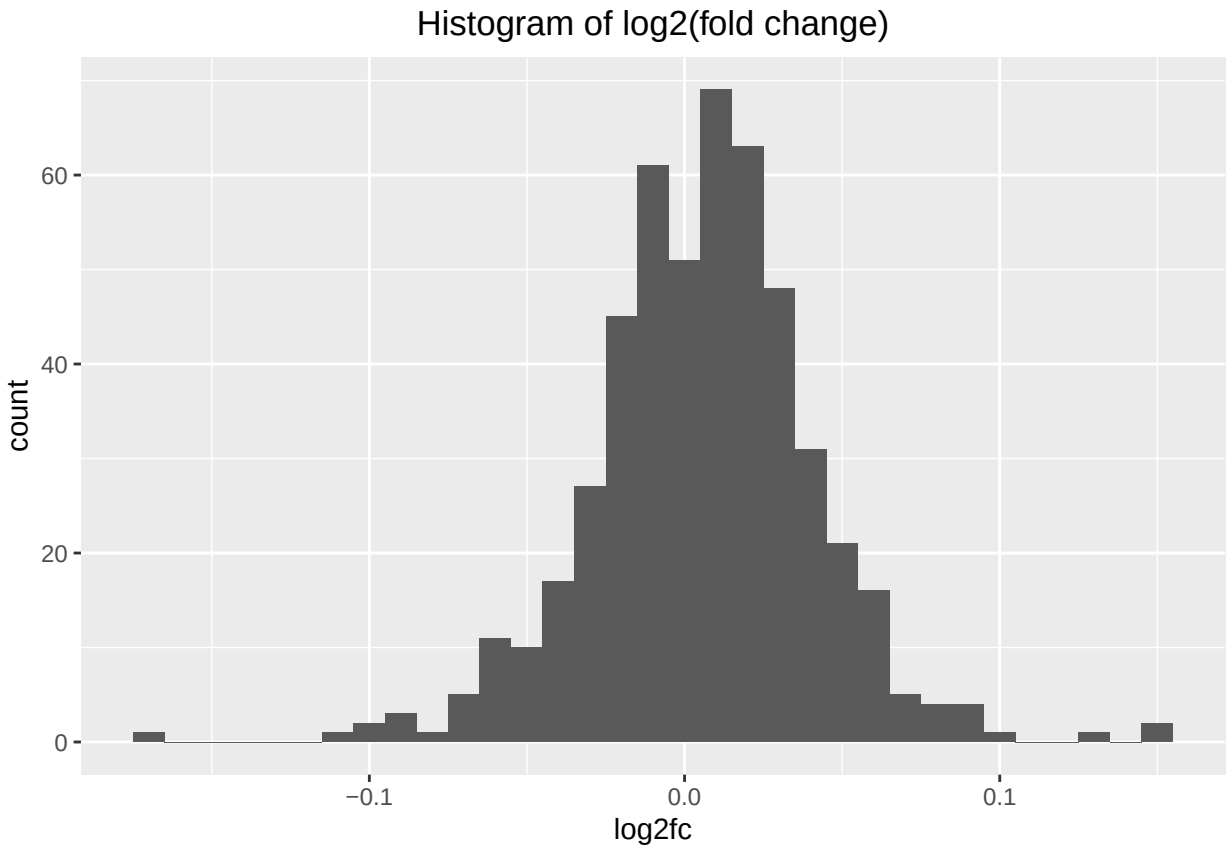



```
nrow(tumor.wilcoxon[tumor.wilcoxon$pvalue<=0.05,]);
```

```
## [1] 24
```

2. Fold change

```
tumor.fc = tumor.merge %>%
  mutate(mean.a = (Patient1+Patient2+Patient3)/3,
            mean.b = rowSums(.[type.b])/9,
         log.mean.a = log2(mean.a),
         log.mean.b = log2(mean.b),
         log2fc=log.mean.b - log.mean.a) %>%
  select(GeneID, log2fc);

ggplot(tumor.fc, aes(log2fc)) +
  geom_histogram(binwidth = 0.01) +
  ggtitle("Histogram of log2(fold change)")+
  theme(plot.title = element_text(hjust=0.5));
```



Both the t test and wilcoxon log p values histgrams shows a log normal trend and gave similar results about the p value. The fold change histgram shows a normal trend with mean around 0, showing that most genes do not have a big difference in mRNA level between type A and B.

3. Use apply

apply(X, MARGIN, FUN)

- x: an array or matrix
- MARGIN: take a value or range between 1 and 2 to define where to apply the function:
- MARGIN=1‘: the manipulation is performed on rows
- MARGIN=2‘: the manipulation is performed on columns
- MARGIN=c(1,2)‘ the manipulation is performed on rows and columns
- FUN: tells which function to apply. Built functions like mean, median, sum, min, max and even user-defined functions can be applied>

```
### Function 2
# Input variables:
# filename test type
# output variables:
# p values
# description:
# function that read in the dataset and test type, then conduct the test and output p values

sample.test = function(df, test.type){
    if('ttest' == test.type ) {
      result <-tidy(t.test(df[2:4], df[5:13]))$p.value;
      };
```

```
    if('wilcoxon' == test.type ) {
      result <- tidy(wilcox.test(df[2:4], df[5:13], alternative = "two.sided"))$p.value;
      };
    if('fc' == test.type ) {
      result <- log2(mean(df[5:13])/mean(df[2:4]));
      };
  return(result);
}
```

```
result.ttest = apply(as.matrix(sapply(tumor.merge, as.numeric)), 1, FUN =  sample.test,  test.type='tte

result.wilcoxon = apply(as.matrix(sapply(tumor.merge, as.numeric)), 1, FUN =  sample.test,  test.type='
```

```
## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties

## Warning in wilcox.test.default(df[2:4], df[5:13], alternative =
## "two.sided"): cannot compute exact p-value with ties
```

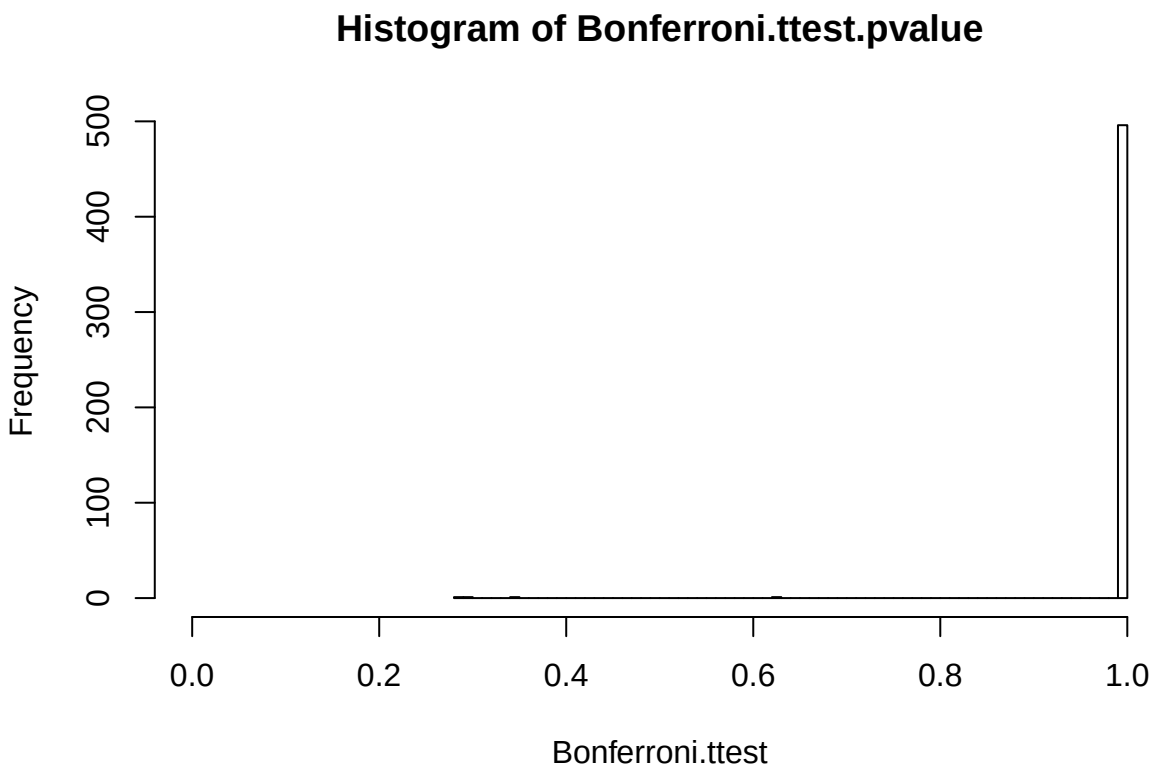
```
result.fc = apply(as.matrix(sapply(tumor.merge, as.numeric)), 1, FUN =  sample.test,  test.type='fc');
```

Question 4

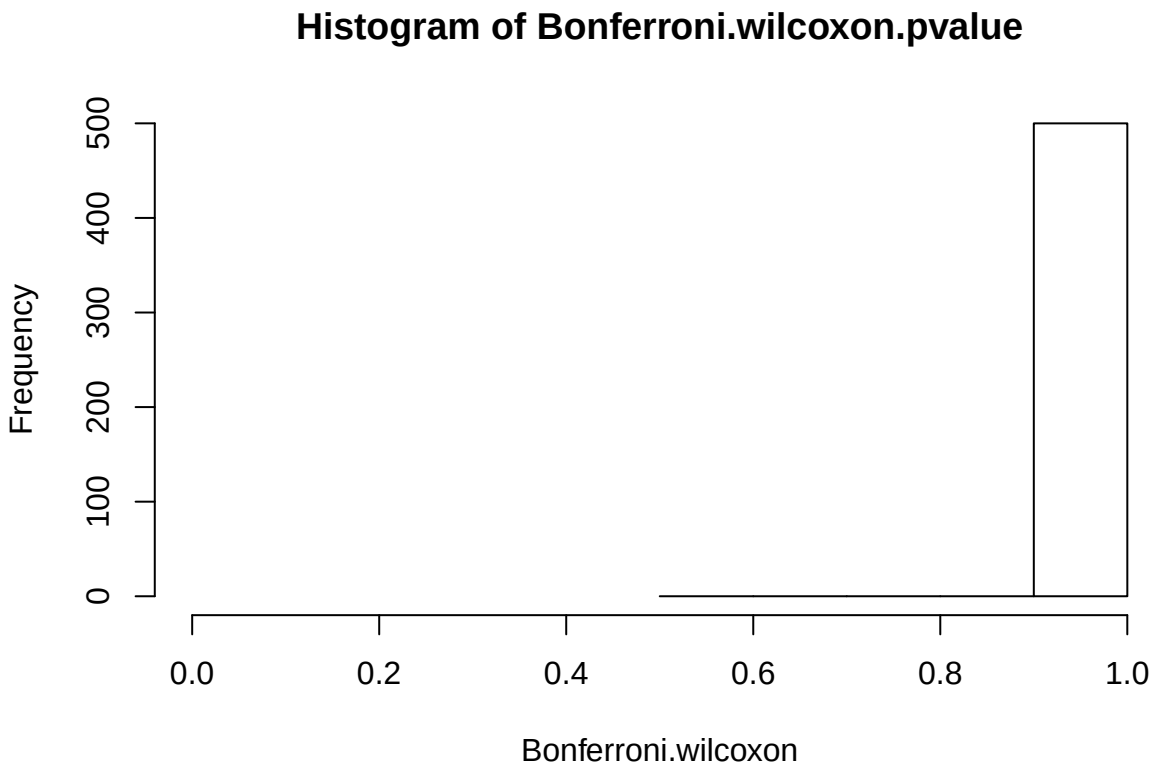
Use FDR and Bonferonni to adjust for multiple testing

- Adjustment for multiple comparisons are needed to avoid spurious associations
- Muiltiple comparison can increase type I error
- Bonferroni is conservative, less powerful: alpha* = alpha/k

```
Bonferroni.ttest =p.adjust(tumor.ttest$pvalue, method = "bonferroni");
hist(Bonferroni.ttest,
     breaks = 100,
     xlim = c(0,1),
     main = "Histogram of Bonferroni.ttest.pvalue");
```

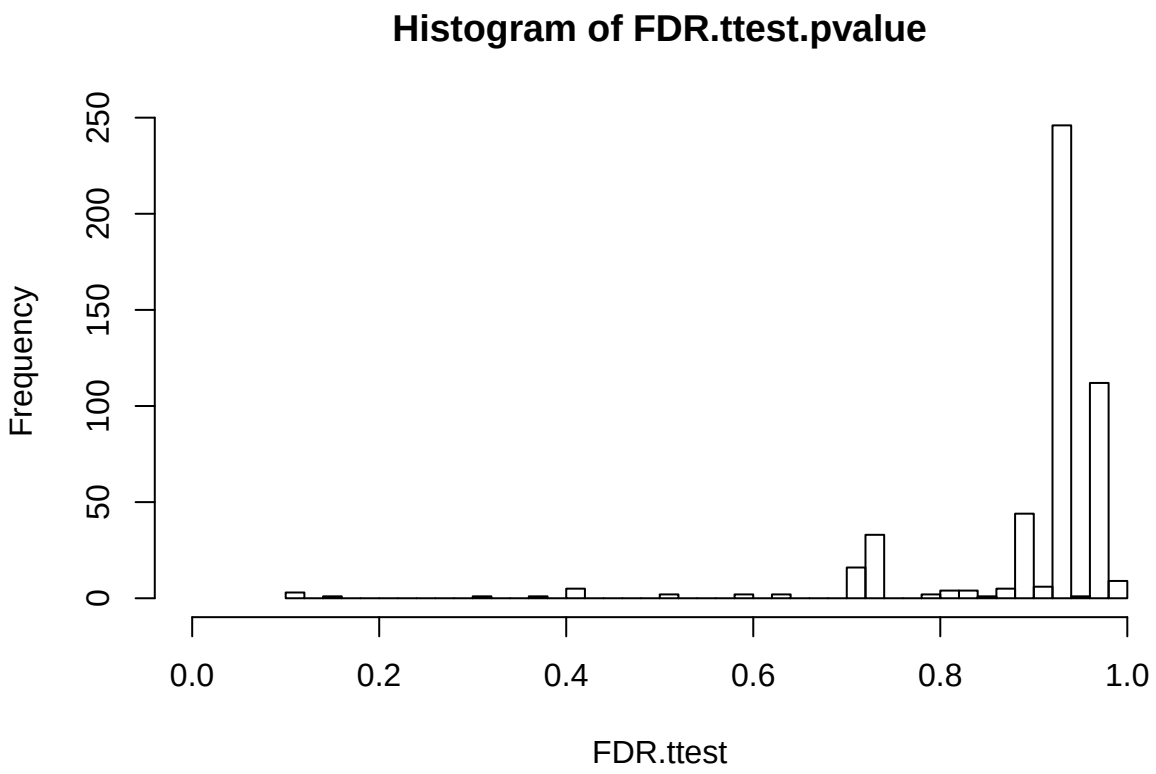



```
Bonferroni.wilcoxon =p.adjust(tumor.wilcoxon$pvalue, method = "bonferroni");
hist(Bonferroni.wilcoxon,
     xlim = c(0,1),
     breaks = c(0.5,0.6,0.7,0.8,0.9,1),
     main = "Histogram of Bonferroni.wilcoxon.pvalue");
```

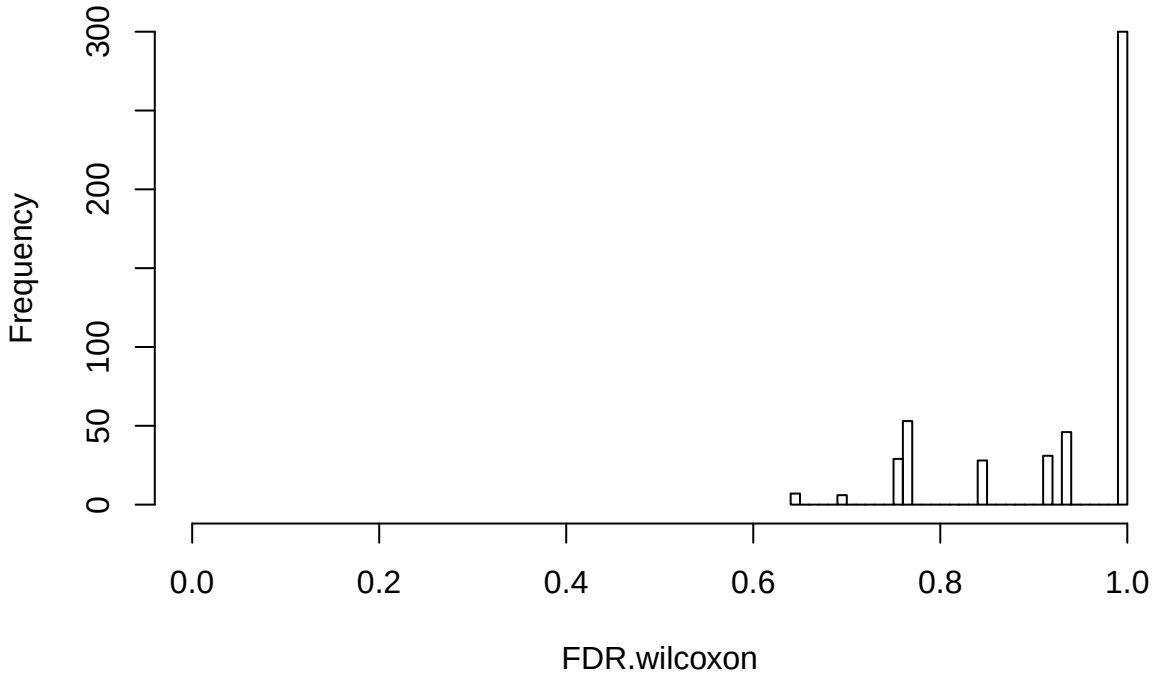


- False Discover Rate controls the number of false positives copared to the total number of positives FDR = expected(#false predictions/# total predictions)
- The methods BH (Benjamini–Hochberg, which is the same as FDR in R) and BY control the false discovery rate.

```
FDR.ttest =p.adjust(tumor.ttest$pvalue, method = "BH");
hist(FDR.ttest,
     breaks = 50,
     xlim = c(0,1),
     main = "Histogram of FDR.ttest.pvalue");
```




```
FDR.wilcoxon =p.adjust(tumor.wilcoxon$pvalue, method = "BH");
hist(FDR.wilcoxon,
     breaks = 50,
     xlim = c(0,1),
     main = "Histogram of FDR.wilcoxon.pvalue");
```



```
x=tumor.ttest$pvalue;
y=cbind(Bonferroni.ttest, FDR.ttest);
```

After the adjustment, the number of significant results are shrinked to 0. And the Bonferroni correction shows a more big effect on the original p value.

Question 5

1. Calculate the median of the first three columns for each gene

```
random.typea = apply(tumor.merge[,2:4],1,median);
```

2. Use a permutation test to estimate the expected value for each gene

```
#a. Randomly select three columns from amongst all 12
set.seed(11);
random.3 = tumor.merge[, sample(2:ncol(tumor.merge),3, replace = F)];

#b. Calculate their median
random.median = apply(random.3,1,median);

#c. Determine if this value is larger or smaller than that of the first 3 columns
median.2 = cbind(random.typea,random.median);

### Function_3
# Input variables:
# filename
# output variables:
# number of genes which have larger median than the first 3 columns
# description:
# function that read in the dataset compare the two columns of medians, output 1 if the
# gene has larger median in the first 3 columns, else output 0.

compare = function(df){
    if(df[1] > df[2]){
      i=1
    }else if(df[1] <= df[2]){
      i=0
    }
  return(i);
}

median.compare = apply(median.2, 1, compare);
median.compare;
```

```
##   [1] 1 1 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0
##  [36] 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0
##  [71] 0 0 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 0 0
## [106] 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0
## [141] 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 1
## [176] 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0
## [211] 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 0
## [246] 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1
```

```
## [281] 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0
## [316] 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0
## [351] 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1
## [386] 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0
## [421] 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0 1 0 0
## [456] 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0
## [491] 0 1 1 1 1 0 1 1 1 1
```

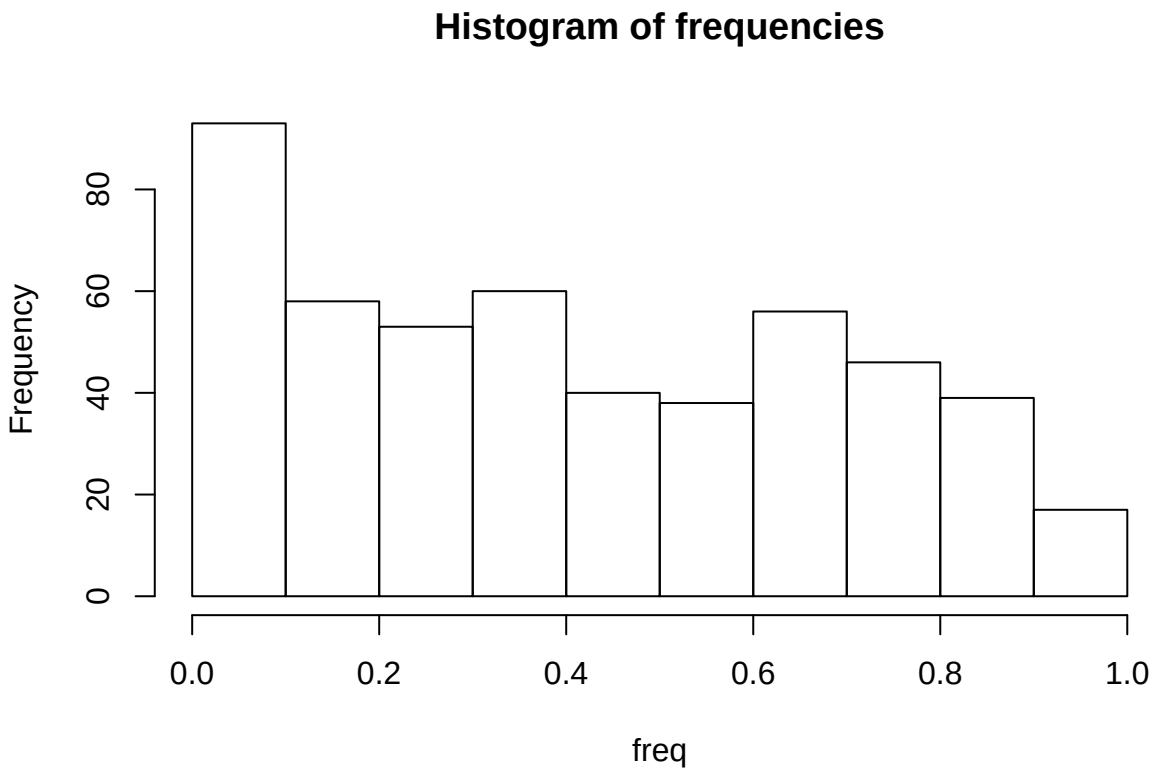
```
table(median.compare);
```

```
## median.compare
##   0   1
## 300 200
```

```
# We can find out among 500 genes, 200 have a smaller median in the first 3 columns.

# d. Repeat 1000 times

### function_4
# Input variables:
# filename
# output variables:
# comparison results of two groups of median
# description:
# function that read in the dataset, generate the median of first 3 columns and random 3 columns,
# compare the two columns of medians and if the random group median is bigger, i=1, if not, i=0.
# The final output is a list of 1 and 0

sample.median = function(df){
  # select 3 random columns from the dataframe
  df1 = df[,sample(2:ncol(df), 3, replace = T)];
  # calculate the median of the 3 values
  median.1 = apply(df1, 1, median);
  # calculate the median of first 3 column
  median.a = apply(df[,2:4],1,median);
  #put the two columns of median together
  df2 = cbind(median.a, median.1);
  # compare the two columns
  num = apply(df2, 1, compare);
  return(num);
}

set.seed(16);
rep = tumor.merge$GeneID;
for (i in 1:1000){
  rep = cbind(rep,sample.median(tumor.merge));
}
```

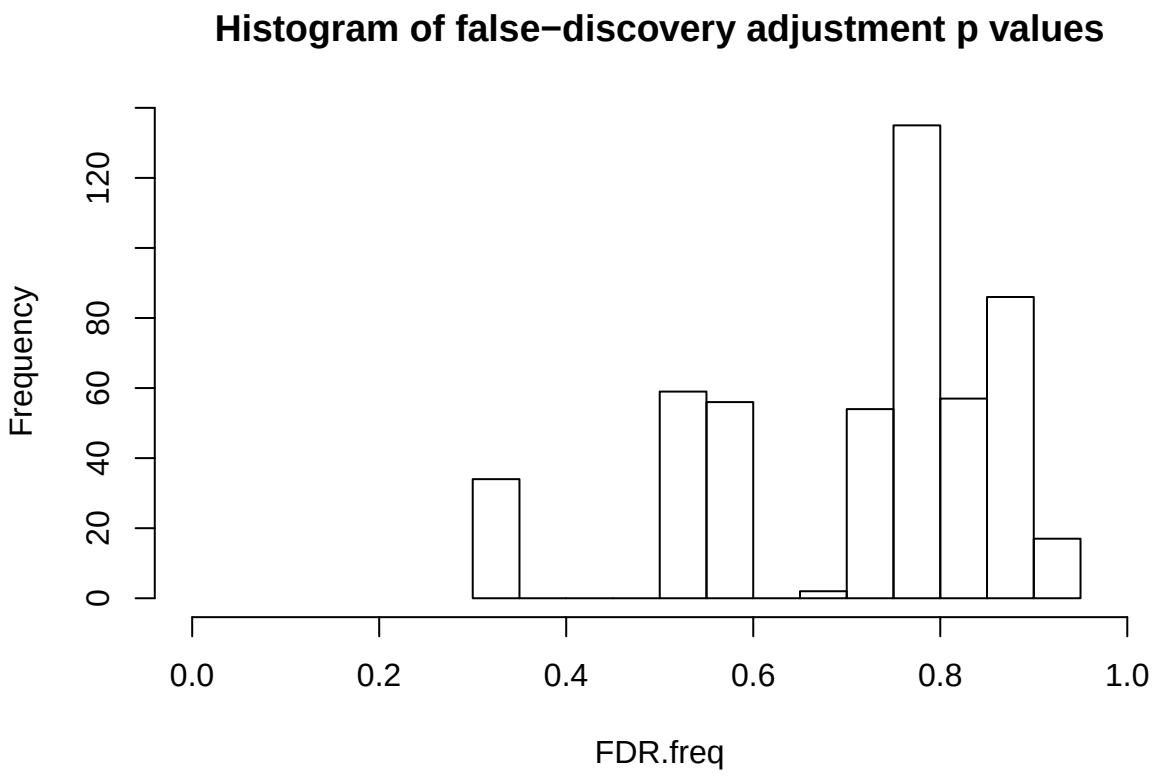
3. Use the frequencies in 2. to estimate a p-value for each gene

```
freq = apply(rep[, 2:1001], 1, sum)/1000;
hist(freq,
     main = "Histogram of frequencies"
     );
```



4.Perform a false-discovery adjustment on the p-values

```
FDR.freq =p.adjust(freq, method = "BH");
hist(FDR.freq,
     xlim = c(0,1),
     main = "Histogram of false-discovery adjustment p values"
     );
```

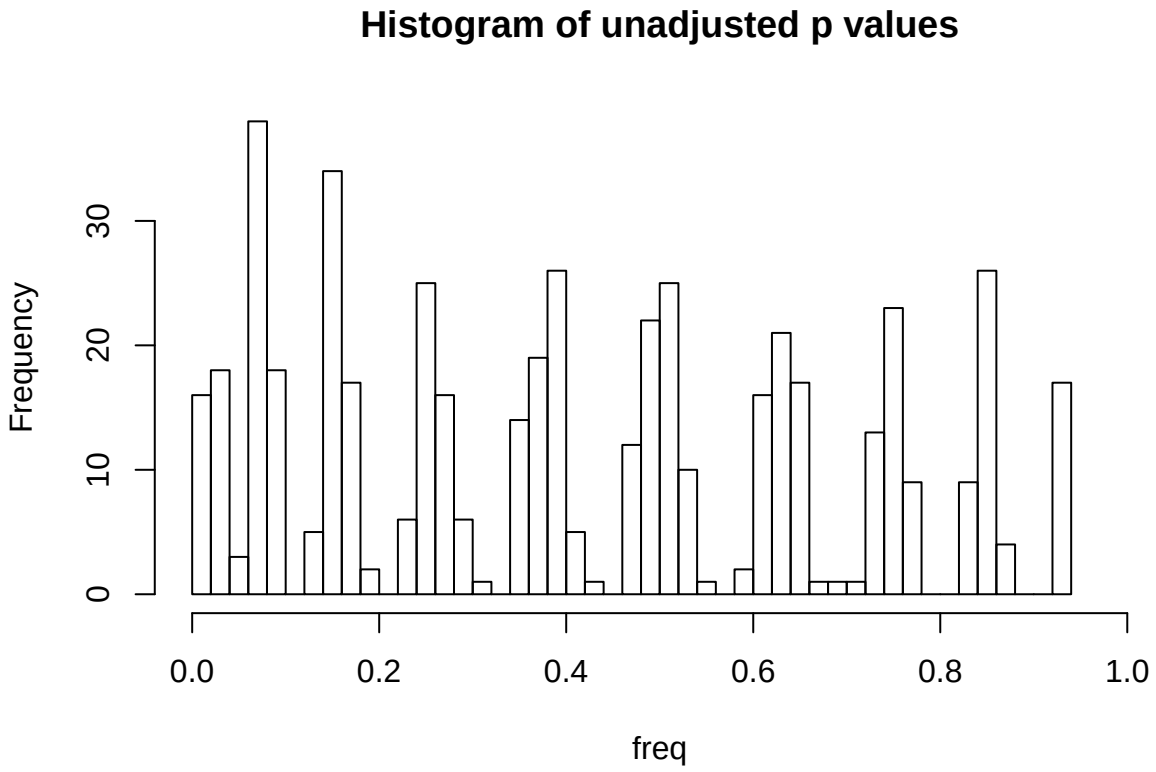


5.Write your results to file in a tab-delimited format

```
output = cbind(as.character(tumor.merge$GeneID), random.typea);
output = cbind(output, apply(tumor.merge[,2:13],1,median));
output = cbind(output,freq);
output = cbind(output, FDR.freq);
colnames(output) = c("GeneID", "Oberserved Median", "Expected Median", "P-value", "Adjusted P-value");

 write.table(output, "/cloud/project/result_q5.txt",
             append = FALSE,
             sep = " ",
             dec = ".",
             col.names = TRUE
             );
```

6.Plot a histogram of the (unadjusted) p-values. What does this tell you?

```
hist(freq,
    breaks = 50,
    xlim = c(0,1),
    main = "Histogram of unadjusted p values"
    );
```



h0: ma>mb h1: ma<mb Among 500 genes, most genes do not have a higher level of mRNA in typeA tumor.

In the case where a set of observations can be assumed to be from an independent and identically distributed population, this can be implemented by constructing a number of resamples with replacement, of the observed dataset (and of equal size to the observed dataset). Bootstrap would underestimate the skewed tails. so the distribution would be more centered than the true distribution.

Question 6

Make modifications for all questions from Q2 onwards using BoutrosLab Package.

```
# 1.read file
tumor1 = read.table("/cloud/project/input1.txt", header=T);
tumor2 = read.table("/cloud/project/input2.txt", header=T);

# 2.combine two files
# 2.1 Sort the data and cbind
tumor1.sort = arrange(tumor1,GeneID);
tumor2.sort = arrange(tumor2,GeneID);
tumor.sort.cbind = cbind(tumor1.sort, tumor2.sort);
tumor.sort.cbind = tumor.sort.cbind[,-5];

#2.2 Merge
tumor.merge = merge(tumor1, tumor2, all=T);

#2.3 Verify if they get the same results
table(tumor.sort.cbind == tumor.merge, useNA = 'ifany');
```

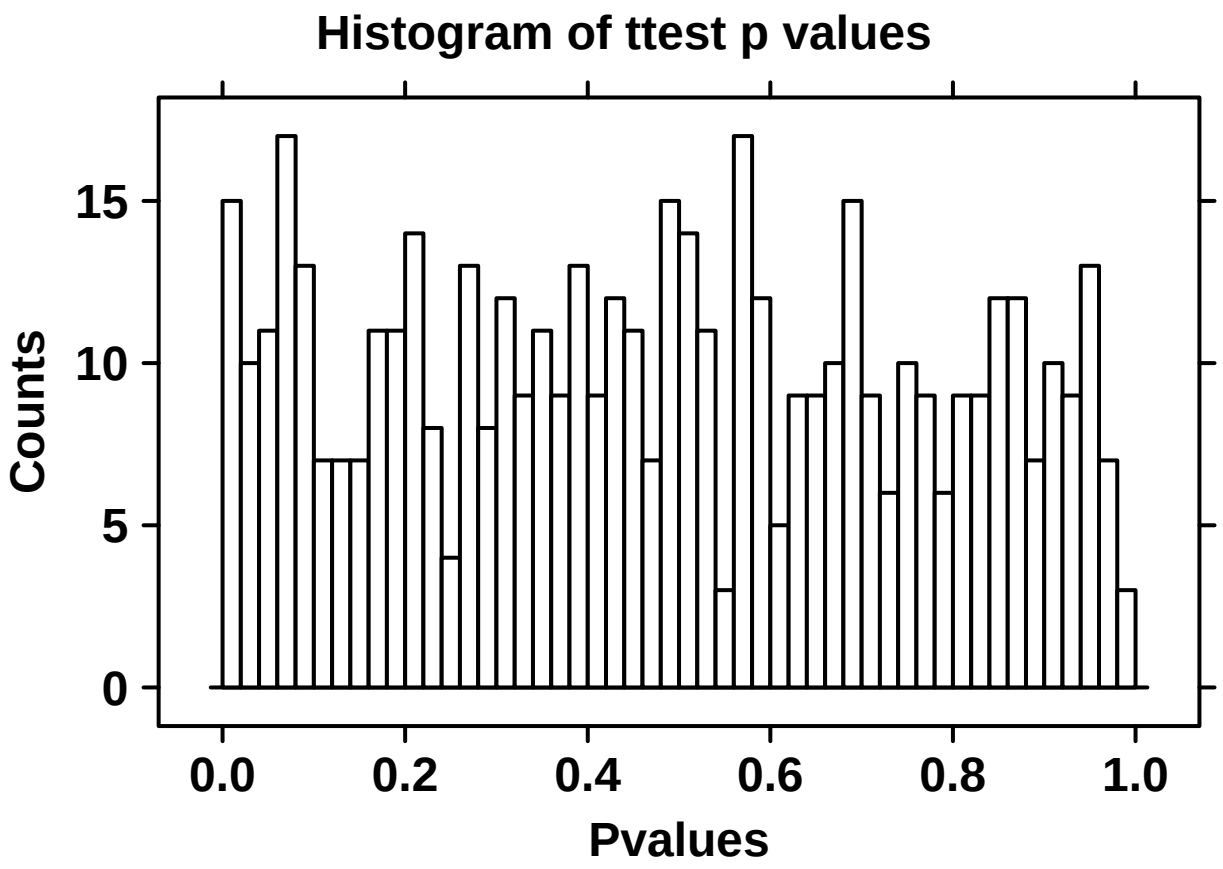
```
##
## TRUE
```

```
## 6500
```

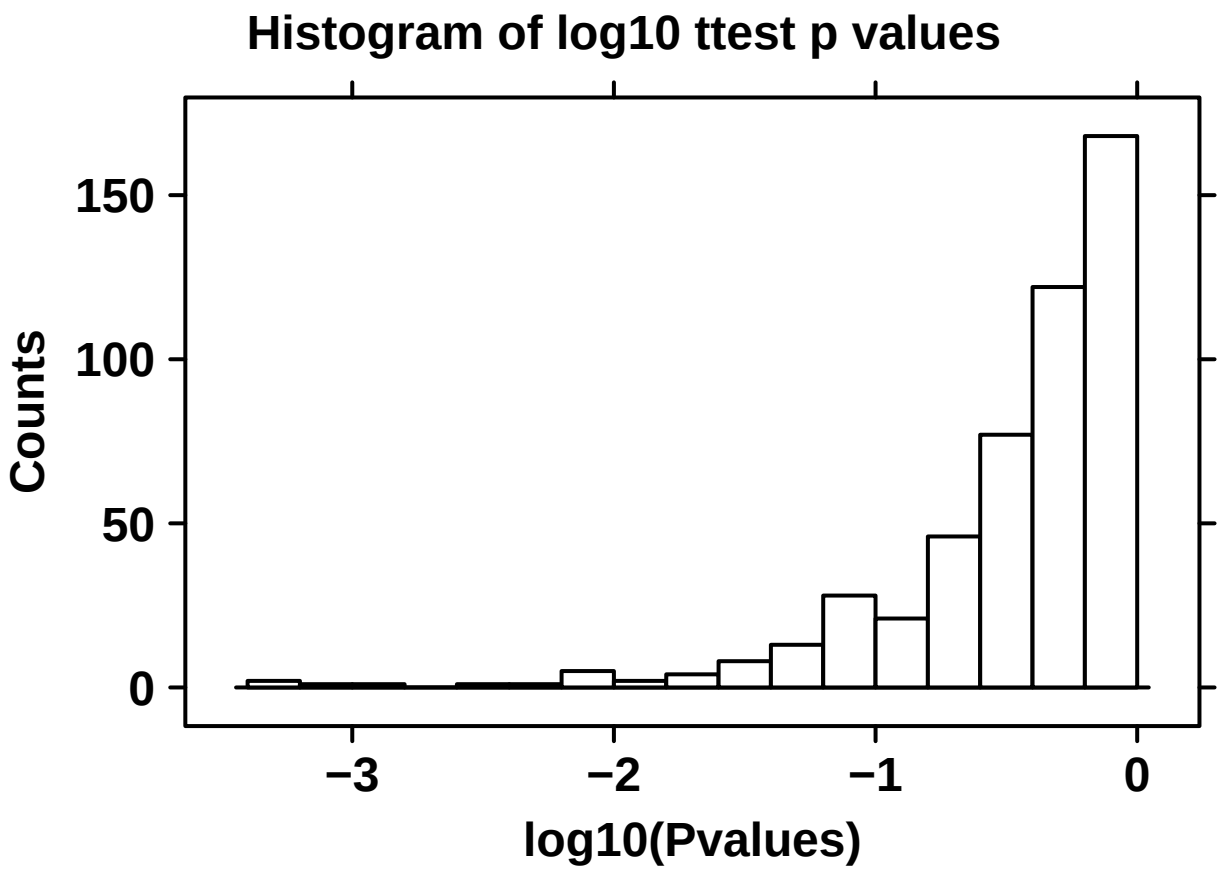
```
# after removing the repeated gene ID colums in the sort&cbind way, the two ways gave the same results.

# 3. Perform a t-test comparing the first three tumours to the last nine tumours for *each* gene using
# create empty dataframe to put p values
tumor.ttest2 = emptydf(500, 2, c('character','numeric'), c('GeneID', 'pvalue'));
```

```
## [1] "character"
## [1] "numeric"
```

```
tumor.ttest2$GeneID=tumor.merge$GeneID;

# perform ttest
for (i in 1:nrow(tumor.merge)){
  tumor.ttest2[i,2] = get.ttest.p(tumor.merge[i,],
                                  group1 = c(F, T, T, T, rep(F,9)),
                                  group2 = c(rep(F,4), rep(T,9)),
                                  paired = FALSE,
                                  var.equal = FALSE,
                                  alternative = 'two.sided'
                                  );
}


# 4. histogram of the p values
create.histogram(tumor.ttest2$pvalue,
                 breaks = 50,
                 type = 'count',
                 ylab.label = 'Counts',
                 xlab.label = 'Pvalues',
                 main = "Histogram of ttest p values",
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main.cex = 1.5
                 );
```




```
# 5. It's not rotated

# 6. plot pvalues in log space
tumor.ttest2$log10.pvalue = log10(tumor.ttest2$pvalue);
create.histogram(tumor.ttest2$log10.pvalue,
                 type = 'count',
                 breaks = 20,
                 ylab.label = 'Counts',
                 xlab.label = 'log10(Pvalues)',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of log10 ttest p values",
                 main.cex = 1.5
                 );
```




```
# 7. Since log(0.05)=-3, from the plot we can find out that most p values from the ttest
# are greater than 0.05. Therefore, among the 500 genes, there don't exsit a significant
# mRNA levels differnece between Tumor type A and type B.
nrow(tumor.ttest[tumor.ttest$pvalue<=0.05,]);#33
```

```
## [1] 33
```

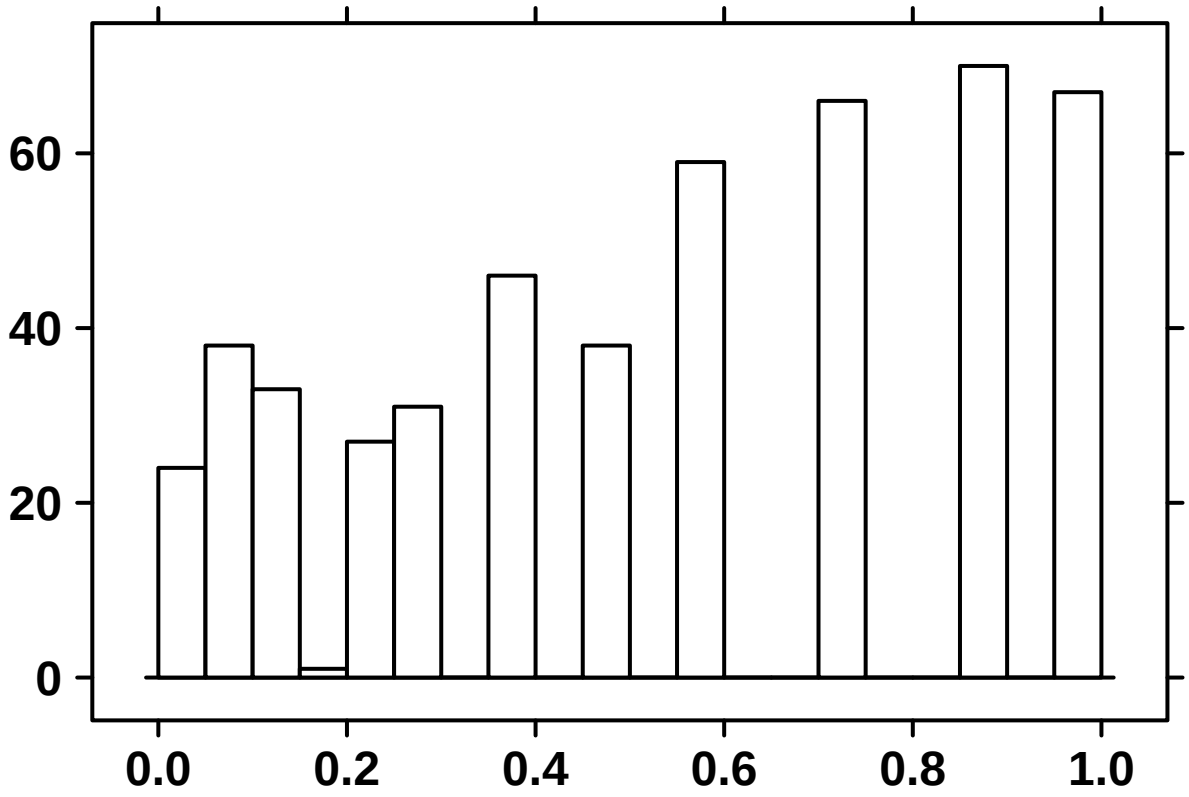
```
### Question3 ##################################################################################

# 1.Wilcoxon test
# create empty dataframe
tumor.wilcoxon2 = emptydf(500, 2, c('character','numeric'), c('GeneID', 'pvalue'));
```

```
## [1] "character"
## [1] "numeric"
```

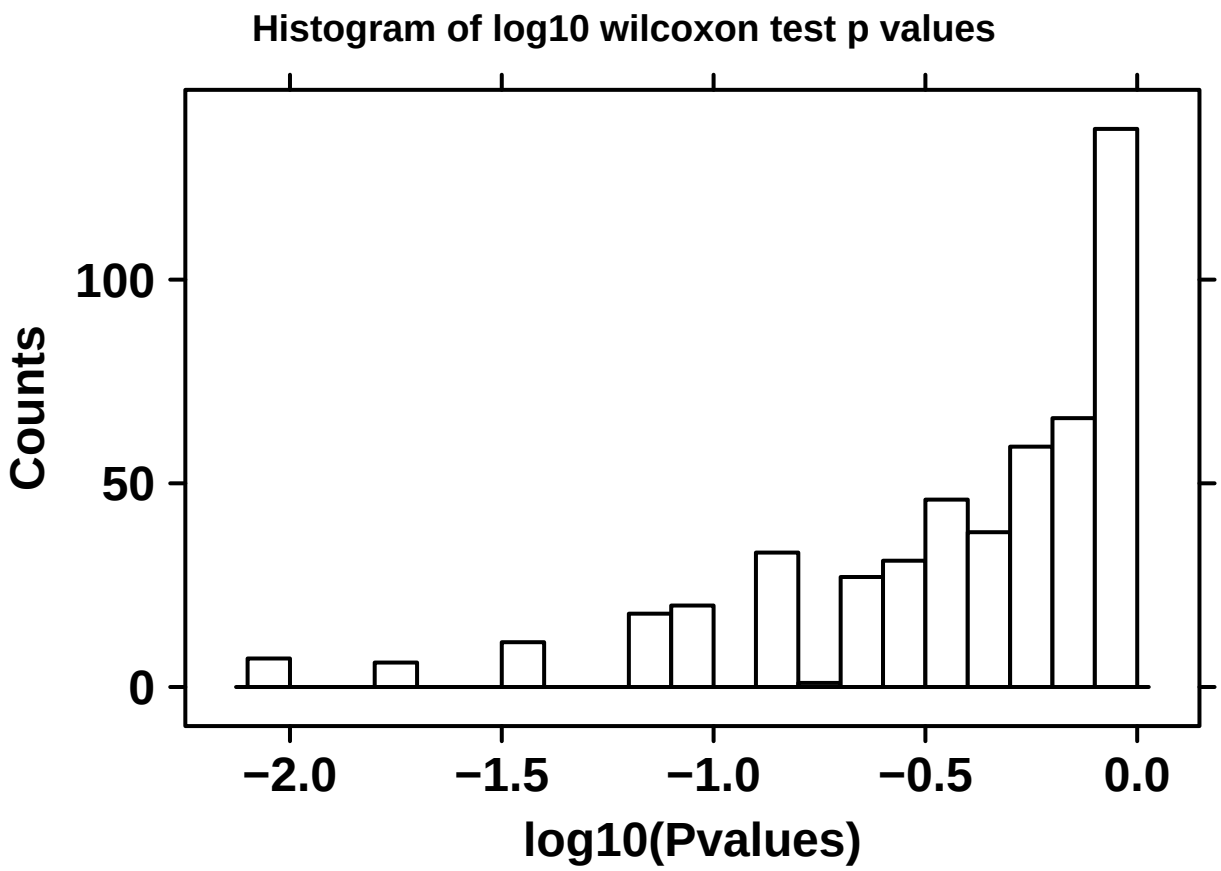
```
tumor.wilcoxon2$GeneID = tumor.merge$GeneID;

# perform wilcoxon rank sum test
for (i in 1:nrow(tumor.merge)){
  tumor.wilcoxon2[i,2] = get.utest.p(as.matrix(sapply(tumor.merge[i,], as.numeric)),
                                     group1 = c(F, rep(TRUE, 3), rep(FALSE, 9)),
                                     group2 = c(rep(FALSE,4), rep(TRUE, 9)),
                                     paired = FALSE,
                                     alternative = 'two.sided'
                                     );
}
```

```
## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties
```

```r
# 14 warnings show there are tied data

# plot histogram
create.histogram(tumor.wilcoxon2$pvalue, breaks = 20, type = 'count');
```




```
#log transform the p values and plot the histogram
tumor.wilcoxon2$log10.pvalue = log10(tumor.wilcoxon2$pvalue);
create.histogram(tumor.wilcoxon2$log10.pvalue,
                 breaks = 20,
                 type = 'count',
                 ylab.label = 'Counts',
                 xlab.label = 'log10(Pvalues)',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of log10 wilcoxon test p values",
                 main.cex = 1.2
                 );
```




```
nrow(tumor.wilcoxon[tumor.wilcoxon$pvalue<=0.05,]);#24
```

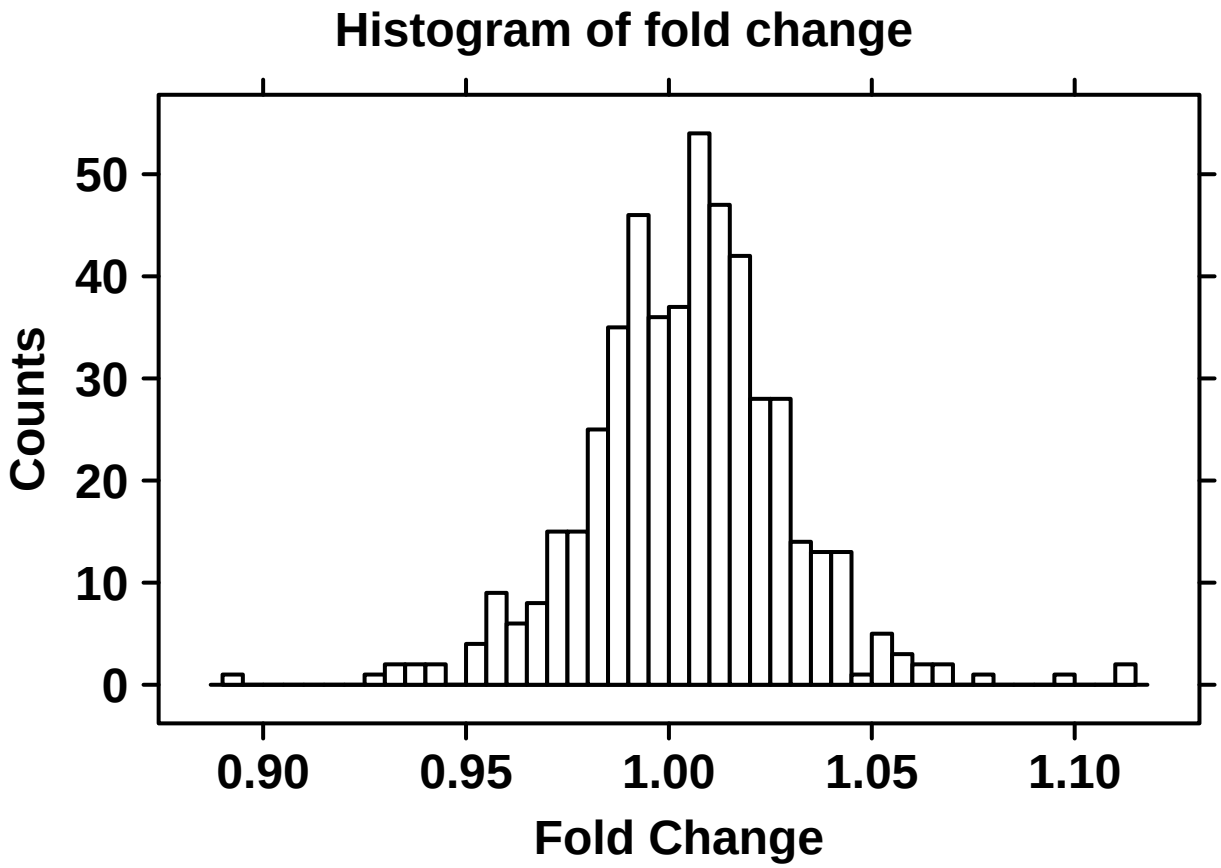
```
## [1] 24
```

```
# 3. fold change
tumor.fc2 = emptydf(500, 2, c('character','numeric'), c('GeneID', 'fc'));
```

```
## [1] "character"
## [1] "numeric"
```

```
for (i in 1:nrow(tumor.merge)){
tumor.fc2[i,] = get.foldchange(as.matrix(sapply(tumor.merge[i,], as.numeric)),
                           group1 = c(F, rep(TRUE, 3), rep(FALSE, 9)),
                            group2 = c(rep(FALSE,4), rep(TRUE, 9)),
                            logged = F
                           );
}


create.histogram(tumor.fc2$fc,
                 type = 'count',
                 breaks = 40,
                 ylab.label = 'Counts',
                 xlab.label = 'Fold Change',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of fold change",
                 main.cex = 1.5
```

```
                );
```



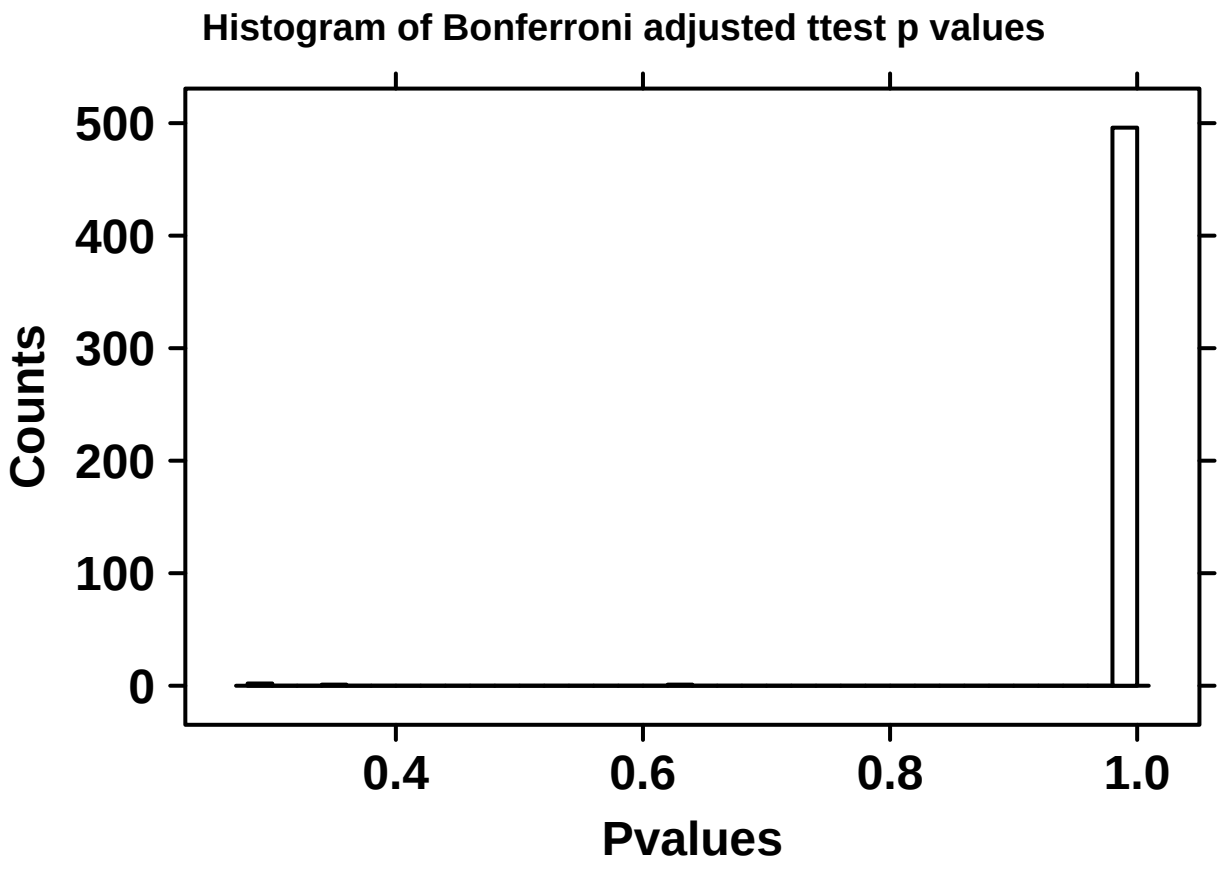

```
# 4. Use apply
# apply(X, MARGIN, FUN)
# -x: an array or matrix
# -MARGIN:  take a value or range between 1 and 2 to define where to apply the function:
# -MARGIN=1`: the manipulation is performed on rows
# -MARGIN=2`: the manipulation is performed on columns
# -MARGIN=c(1,2)` the manipulation is performed on rows and columns
# -FUN: tells which function to apply. Built functions like mean, median, sum, min, max and even user-d

### Function_2 ################################################################################
# Input variables:
# filename test type
# output variables:
# p values
# description:
# function that read in the dataset and test type, then conduct the test and output p values

result.ttest2 =apply(as.matrix(sapply(tumor.merge, as.numeric)), 1,
                     FUN = get.ttest.p,
                     group1 = c(rep(FALSE, 1), rep(TRUE, 3), rep(FALSE, 9)),
                     group2 = c(rep(FALSE,4), rep(TRUE, 9)),  paired = FALSE, var.equal = FALSE,
                     alternative = 'two.sided'
                     );
```

```r
result.wilcoxon2 = apply(as.matrix(sapply(tumor.merge, as.numeric)), 1,
                         FUN = get.utest.p,
                         group1 = c(rep(FALSE, 1), rep(TRUE, 3), rep(FALSE, 9)),
                         group2 = c(rep(FALSE,4), rep(TRUE, 9)),
                         paired = FALSE,
                         alternative = 'two.sided'
                         );
```

```
## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties

## Warning in wilcox.test.default(x = x[group1], y = if (is.null(group2)) {:
## cannot compute exact p-value with ties
```

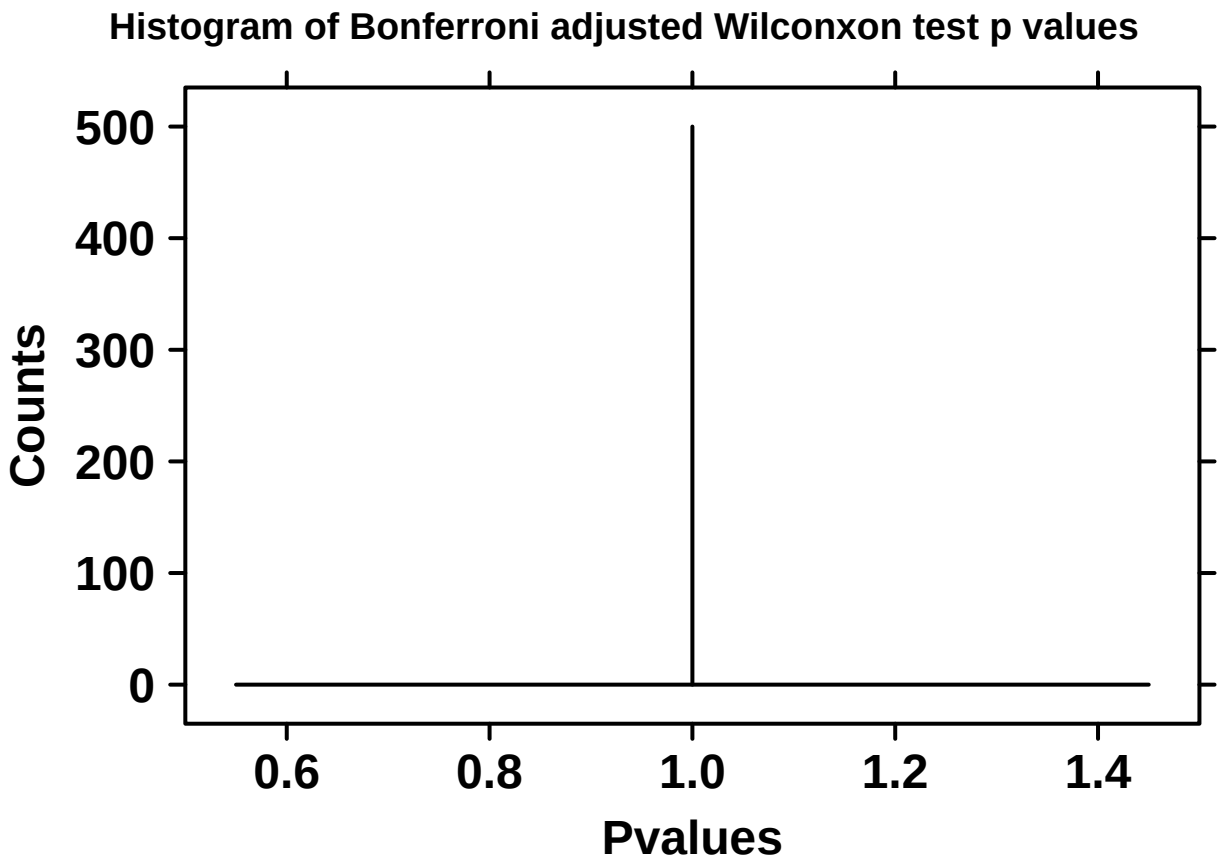
```r
result.fc2 = apply(as.matrix(sapply(tumor.merge, as.numeric)), 1,
                   FUN =  get.foldchange,
                   group1 = c(rep(FALSE, 1), rep(TRUE, 3), rep(FALSE, 9)),
                   group2 = c(rep(FALSE,4), rep(TRUE, 9)),
```

```
                  logged = F
                  );

### Question 4 ##################################################################################################
# Adjustment for multiple comparisons are needed to avoid spurious associations.
# MUiltiple comparison can increase type I error
# Bonferroni is conservative, less powerful: alpha* = alpha/(k  2)
Bonferroni.ttest =p.adjust(tumor.ttest$pvalue, method = "bonferroni");
create.histogram(Bonferroni.ttest,
                  type = 'count',
                  breaks=50,
                  ylab.label = 'Counts',
                  xlab.label = 'Pvalues',
                  xlab.cex = 1.5,
                  ylab.cex = 1.5,
                  main = "Histogram of Bonferroni adjusted ttest p values",
                  main.cex = 1.2
                  );
```

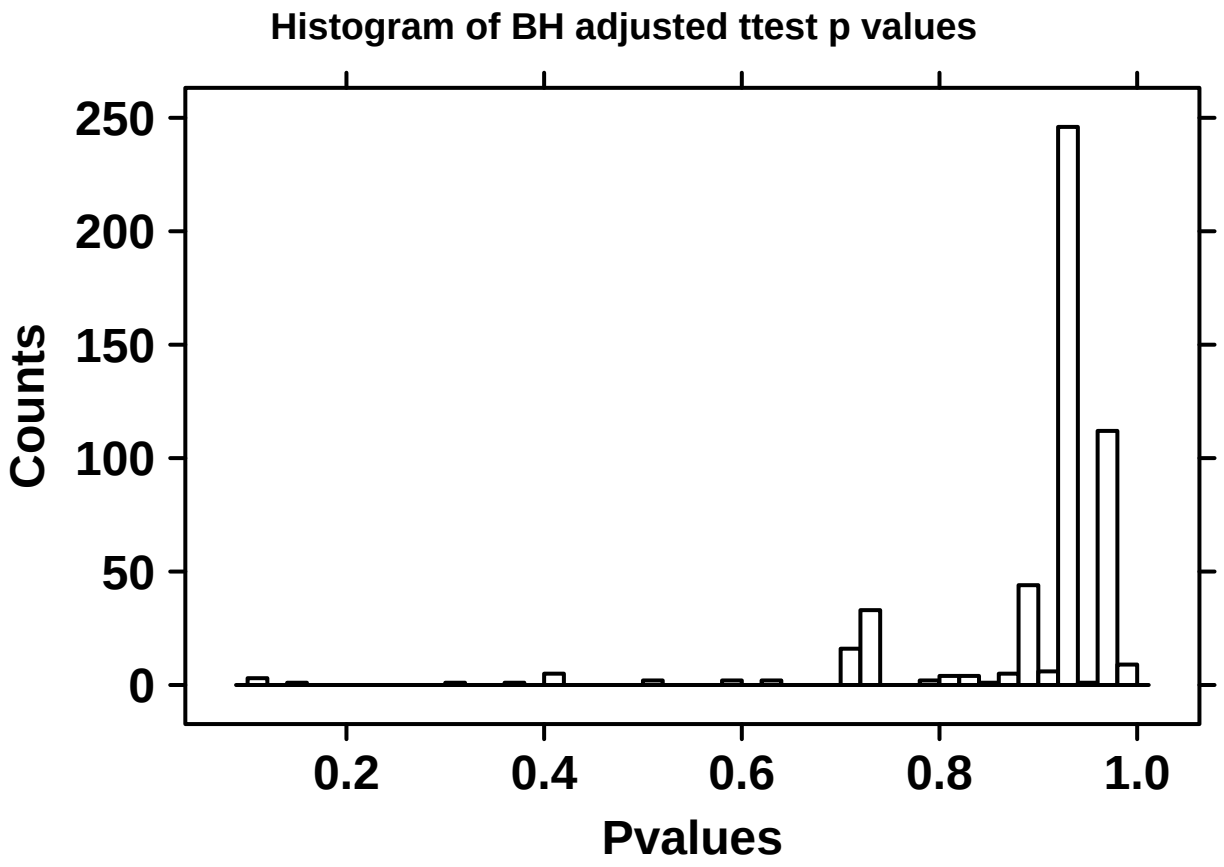



```
Bonferroni.wilcoxon =p.adjust(tumor.wilcoxon$pvalue, method = "bonferroni");
create.histogram(Bonferroni.wilcoxon,
                  type = 'count',
                  ylab.label = 'Counts',
                  xlab.label = 'Pvalues',
                  xlab.cex = 1.5,
                  ylab.cex = 1.5,
                  main = "Histogram of Bonferroni adjusted Wilconxon test p values",
```

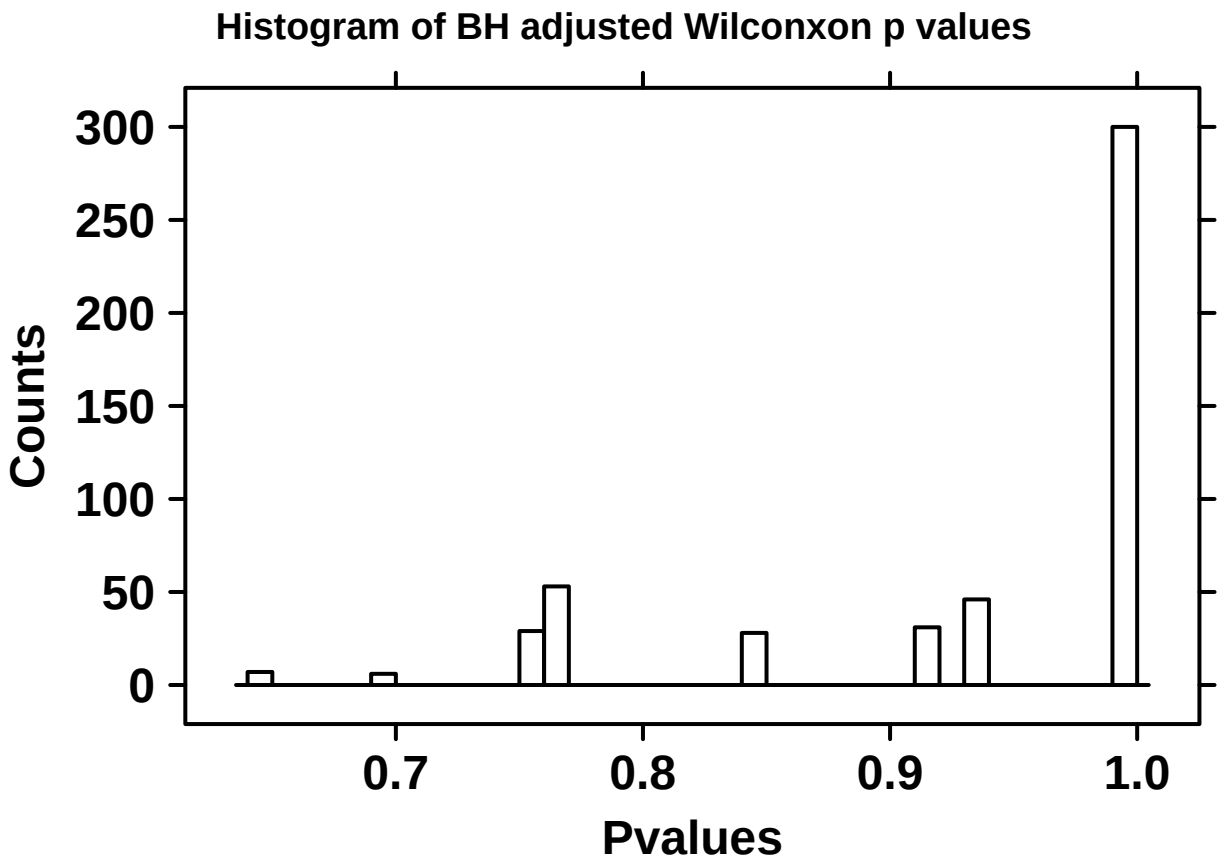
```
                main.cex = 1.2
                );
```




```
# False Discover Rate controls the number of false positives copared to the total number of positives
# The methods BH (Benjamini-Hochberg, which is the same as FDR in R) and BY control the false discovery
FDR.ttest =p.adjust(tumor.ttest$pvalue, method = "BH");
create.histogram(FDR.ttest,
                breaks=50,type = 'count',
                ylab.label = 'Counts',
                xlab.label = 'Pvalues',
                xlab.cex = 1.5,
                ylab.cex = 1.5,
                main = "Histogram of BH adjusted ttest p values",
                main.cex = 1.2
                );
```




```
FDR.wilcoxon =p.adjust(tumor.wilcoxon$pvalue, method = "BH");
create.histogram(FDR.wilcoxon,
                 type = 'count',
                 breaks=50,
                 ylab.label = 'Counts',
                 xlab.label = 'Pvalues',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                main = "Histogram of BH adjusted Wilconxon p values",
                 main.cex = 1.2
                );
```




```
x=tumor.ttest$pvalue;
y=cbind(Bonferroni.ttest, FDR.ttest);


# After the adjustment, the number of significant results was shrinked to 0. And the Bonferroni correct
# shows a more big effect on the original p value

### question 5 ###############################################################################
# 1. Calculate the median of the first three columns for each gene
random.typea = apply(tumor.merge[,2:4],1,median);

# 2. Use a permutation test to estimate the expected value for each gene:
# a. Randomly select three columns from amongst all 12
set.seed(11)
random.3 = tumor.merge[, sample(2:ncol(tumor.merge),3, replace = F)];

# b. Calculate their median
random.median = apply(random.3,1,median);

# c. Determine if this value is larger or smaller than that of the first 3 columns
median.2 = cbind(random.typea,random.median);

### Function_3 ###############################################################################
# Input variables:
# filename
# output variables:
# number of genes which have larger median than the first 3 columns
```

```
# description:
# function that read in the dataset compare the two columns of medians, output 1 if the
# gene has larger median in the first 3 columns, else output 0.
compare = function(df){
  if(df[1] > df[2]){
    i=1;
  }else if(df[1] <= df[2]){
    i=0;
  }
  return(i);
}

median.compare = apply(median.2, 1, compare);
median.compare;
```

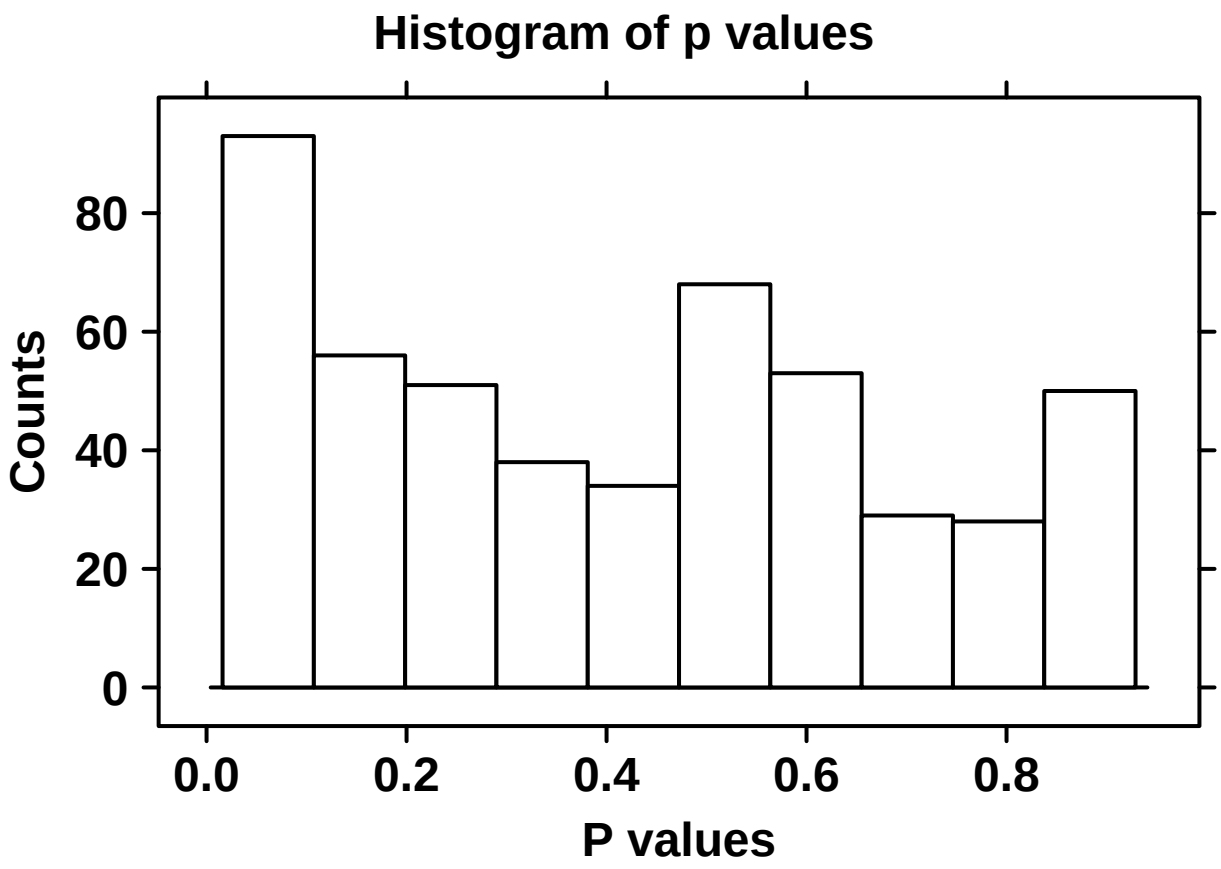
```
##   [1] 1 1 0 0 0 1 0 0 1 0 1 1 1 0 0 1 0 1 0 1 0 0 0 0 0 0 1 1 0 0 1 0 0 0 0
##  [36] 0 0 1 1 1 0 1 0 1 1 1 0 0 1 0 1 0 1 1 0 1 0 0 0 0 0 0 0 0 1 0 0 1 0
##  [71] 0 0 0 1 1 1 1 0 1 0 0 1 0 0 1 1 0 0 0 1 1 1 0 0 0 1 0 0 0 0 1 0 1 0 0
## [106] 1 0 0 1 1 0 1 0 1 0 0 0 0 0 0 1 1 1 0 1 1 1 0 0 0 0 0 1 1 0 0 0 0 1 0
## [141] 0 0 1 1 0 0 1 0 0 1 1 0 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 0 1 1 0 0 1 0 1
## [176] 0 0 1 1 0 0 0 0 0 0 0 0 1 1 1 1 0 1 0 1 0 0 0 1 0 0 1 0 1 1 0 1 1 0 0
## [211] 0 0 1 1 0 0 0 0 1 1 1 0 1 1 1 1 0 1 0 0 1 0 1 1 0 0 0 0 1 0 0 0 1 1 0
## [246] 0 1 1 0 0 1 0 0 0 0 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 0 0 0 1 0 1 1 0 1 1
## [281] 0 0 0 0 0 0 1 1 1 0 0 1 1 1 0 1 1 1 0 0 0 0 0 0 0 1 0 0 1 1 0 0 1 1 0
## [316] 0 0 0 0 0 0 1 0 0 1 1 1 1 0 0 0 1 0 1 0 0 0 0 1 1 0 0 0 0 1 0 0 1 1 0
## [351] 0 0 0 1 1 0 0 1 0 0 0 0 0 0 0 0 0 0 1 0 1 0 0 0 0 0 1 1 0 0 1 0 0 1 1
## [386] 0 0 0 1 1 0 1 0 0 0 0 1 0 1 1 1 1 1 0 1 0 0 1 0 0 1 0 0 0 1 0 1 0 1 0
## [421] 1 0 0 0 0 0 1 0 0 0 0 1 1 1 1 1 0 0 0 0 1 1 1 0 1 1 1 0 0 0 1 0 1 0 0
## [456] 0 1 1 0 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 1 0 0 0 1 0 1 1 1 0 0 0 0
## [491] 0 1 1 1 1 0 1 1 1 1
```

```
table(median.compare);
```

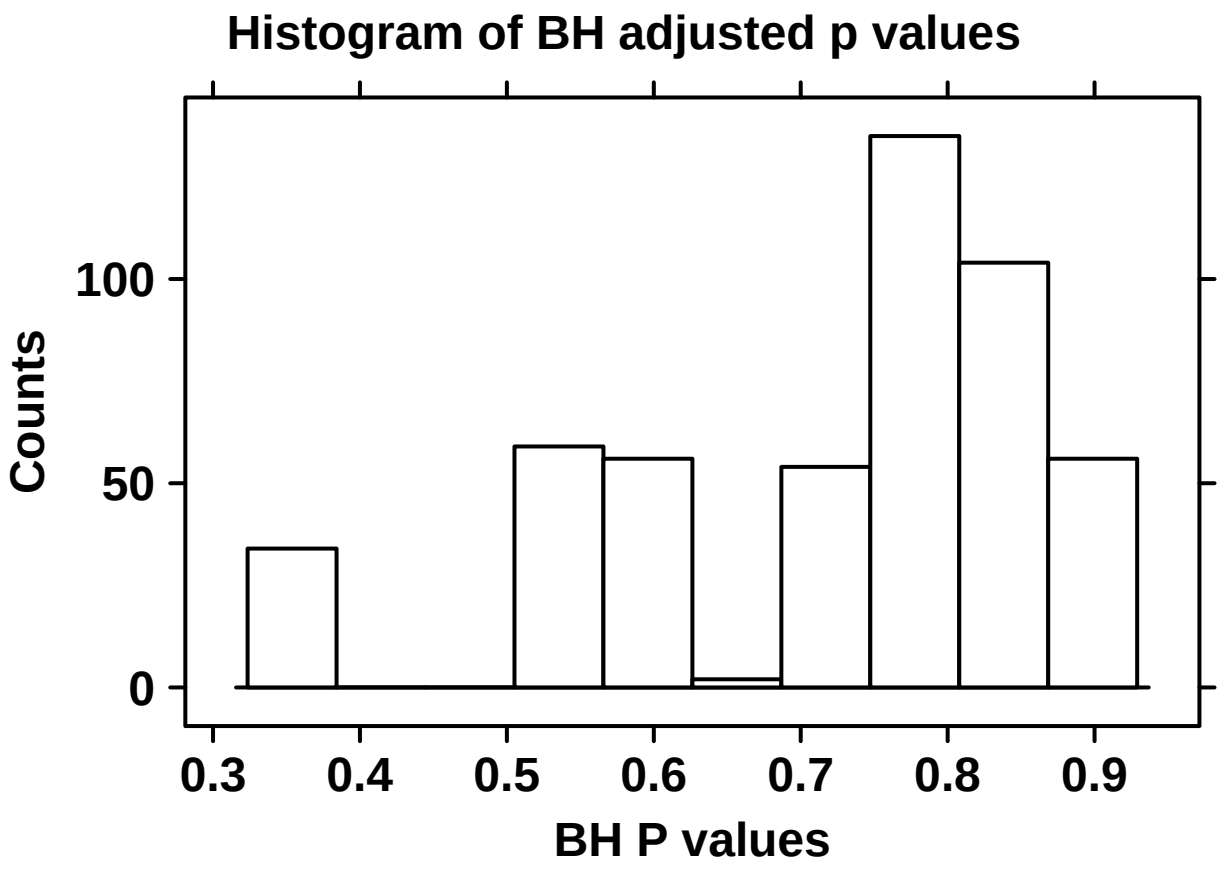
```
## median.compare
##   0   1
## 300 200
```

```
# We can find out among 500 genes, 200 have a smaller median in the first 3 columns.

# d. Repeat 1000 times


### function_4 ################################################################################
# Input variables:
# filename
# output variables:
# comparison results of two groups of median
# description:
# function that read in the dataset, generate the median of first 3 columns and random 3 columns,
# compare the two columns of medians and if the random group median is bigger, i=1, if not, i=0.
# The final output is a list of 1 and 0

sample.median = function(df){
  # select 3 random columns from the dataframe
  df1 = df[,sample(2:ncol(df), 3, replace = T)];
```

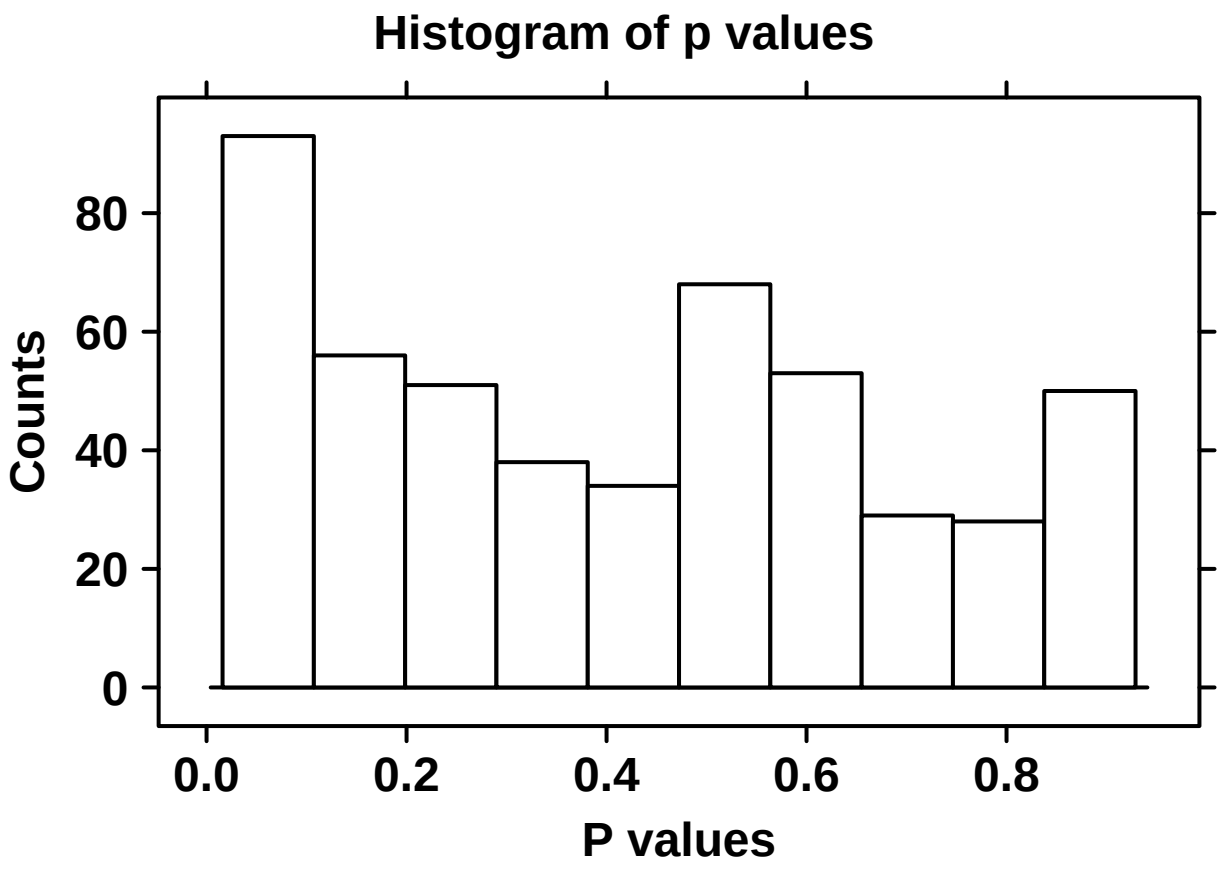
```
  # calculate the median of the 3 values
  median.1 = apply(df1, 1, median);
  # calculate the median of first 3 column
  median.a = apply(df[,2:4],1,median);
  #put the two columns of median together
  df2 = cbind(median.a, median.1);
  # compare the two columns
  num = apply(df2, 1, compare);
  return(num);
}

set.seed(16);
rep = tumor.merge$GeneID;
for (i in 1:1000){
  rep = cbind(rep,sample.median(tumor.merge));
}

# 3. Use the frequencies in 2. to estimate a p-value for each gene
freq = apply(rep[, 2:1001], 1, sum)/1000;

create.histogram(freq, type = 'count',
                 ylab.label = 'Counts',
                 xlab.label = 'P values',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of p values",
                 main.cex = 1.5
                 );
```




```
# 4.Perform a false-discovery adjustment on the p-values (?p.adjust)
FDR.freq =p.adjust(freq, method = "BH");
create.histogram(FDR.freq,
                 type = 'count',
                 ylab.label = 'Counts',
                 xlab.label = 'BH P values',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of BH adjusted p values",
                 main.cex = 1.5
                 );
```




```
# 5.Write your results (gene ID, observed median, expected median, p-value, adjusted p-value) to file i
output = cbind(as.character(tumor.merge$GeneID), random.typea);
output = cbind(output, apply(tumor.merge[,2:13],1,median));
output = cbind(output,freq);
output = cbind(output, FDR.freq);
colnames(output) = c("GeneID", "Oberserved Median", "Expected Median", "P-value", "Adjusted P-value");
write.table(output, "/cloud/project/result_q5.txt",
            append = FALSE,
            sep = " ",
            dec = ".",
            col.names = TRUE
            );

# 6.Plot a histogram of the (unadjusted) p-values. What does this tell you?
create.histogram(freq,
                 type = 'count',
                 ylab.label = 'Counts',
                 xlab.label = 'P values',
                 xlab.cex = 1.5,
                 ylab.cex = 1.5,
                 main = "Histogram of p values",
                 main.cex = 1.5
                 );
```




```
# h0: ma>mb h1: ma<mb
# among 500 genes, most genes do not have a higher level of mRNA in typeA tumor.
```

Methods

In this study, we used a small microarray platform to measure the mRNA levels of 500 genes on 12 tumour samples, three of which are subtype A and the rest are subtype B. To compare the mRNA levels of the two subtype A and B, t test, Wilcoxon rank-sum test and fold changes calculation were applied. The t test compared the difference in means, but the normality assumption was not met due to the small sample size. (1) Therefore, we conducted the Wilcoxon rank-sum test, which doesn't require any assumption about underlying population distribution. (2) We also calculated the fold change to see how much a quantity changes between the two subtypes. In order to adjust for multiple testing, the Bonferroni's correction was conducted, dividing the significance level by the number of comparisons. (3) And we also applied Benjamini-Hochberg procedure to adjust for false discovery rate. (4) Then we developed a bootstrap - like test. We repeated drawing samples from the original dataset and compare the median mRNA level with the median mRNA level of subtype A. And we performed the same adjustments to the p valued estimated from the frequencies. All statistical analyses were carried out in R using the BoutrosLab statistical and plotting packages.

1. Philip L. Witt, Peter McGrain, Comparing Two Sample Means t Tests, Physical Therapy, Volume 65, Issue 11, 1 November 1985, Pages 1730–1733, https://doi.org/10.1093/ptj/65.11.1730
2. Kim HY. Statistical notes for clinical researchers: Nonparametric statistical methods: 1. Nonparametric methods for comparing two groups. Restor Dent Endod. 2014;39(3):235–239. doi:10.5395/rde.2014.39.3.235
3. Etymologia: Bonferroni correction. Emerg Infect Dis. 2015;21(2):289. doi:10.3201/eid2102.et2102
4. Glueck DH, Mandel J, Karimpour-Fard A, Hunter L, Muller KE. Exact calculations of average power for the Benjamini-Hochberg procedure. Int J Biostat. 2008;4(1):11. doi:10.2202/1557-4679.1103