7659 HW3

Guannan Shen

October 4, 2018

Contents


```
## set up workspace
library(knitr)
library(tidyverse)
library(samr)
library(impute)
library(limma)
library(gtools)
library(qvalue)
options(stringsAsFactors = F)
options(dplyr.width = Inf)
getwd()
```

```
## [1] "/home/guanshim/Documents/Stats/CIDA_OMICs/7659Stats_Genetics/HW3"
```

1 T-statistics of microarray data

1.1 1. For each gene, calculate the fold change between the knock-out and wildtype groups. List the top 10 genes that show the largest fold change (positive or negative).

The fold change is calculated by knock-out over wildtype groups.

```
# read in raw data
apodata <- read.table("hw3arraydata.txt", header = TRUE)
dim(apodata)
```

```
## [1] 6384   16
```

```
aponame <- read.table("hw3genenames.txt", header = FALSE, blank.lines.skip = FALSE)
dim(aponame)
```

```
## [1] 6384    1
```

```
colnames(aponame) <- "genenames"

# combine the names and intensity
ai <- cbind(aponame, apodata)

## the raw data is log2 transformed the fold change is
## calculated from the subtraction
ailogratio <- (base::rowSums(ai[, 10:17]) - base::rowSums(ai[,
    2:9]))/8
aifc <- logratio2foldchange(ailogratio)
aifc <- data.frame(ai$genenames, aifc)

# test missing data
kable(apply(ai[, 2:17], 2, function(x) {
    sum(is.na(x))
}), caption = "Sparsity Summary", col.names = "No. Missing Values")
```

Table 1: Sparsity Summary

	No. Missing Values
c1	0
c2	0
c3	0
c4	0
c5	0
c6	0
c7	0
c8	0
k1	0
k2	0
k3	0
k4	0
k5	0
k6	0
k7	0

	No. Missing Values
k8	0

```
## find the top10
kable(head(aifc[order(abs(aifc$aifc), decreasing = TRUE), ],
    10), caption = "Top10 Genes by Fold Change (Knock-out vs. WT)")
```

Table 2: Top10 Genes by Fold Change (Knock-out vs. WT)

	ai.genenames	aifc
2149	ApoAI,lipid-Img	-26.894639
540	EST,HighlysimilartoA	-23.798944
5356	CATECHOLO-METHYLTRAN	-6.831720
4139	EST,WeaklysimilartoC	-2.908813
2537	ESTs,Highlysimilarto	-2.857429
1496	est	-2.762808
4941	similartoyeaststerol	-2.699054
1739	ApoCIII,lipid-Img	-2.636956
1337	psoriasis-associated	-2.389508
5986	Cy3RT	2.286730

1.2 2. Obtain the p-values from a two sided t-test for differential expression. How many genes are significant at the 0.01 level? List the top 10 genes that have the largest t-statistics and their corresponding p-value.

There are 75 genes are significant at the 0.01 level, by gene-specific t-test (two-sided Welch's t-test).

```
# carry out individual t-tests

## welch t-test

indi <- lapply(1:6384, function(row) {
    test = t.test(ai[row, 2:9], ai[row, 10:17], alternative = "two.sided")
    test.sum = c(ai[row, 1], test$p.value, test$statistic)
    test.sum
})

indi_t <- data.frame(matrix(unlist(indi), ncol = 3, byrow = TRUE))
indi_t[1:5, ]
```

```
##      X1                 X2                 X3
## 1 Cy3RT 0.0199004390758332   -2.6304781013414
## 2 Cy5RT  0.004566702365611  -3.42047448834445
## 3 mSRB1  0.334997882015293 -0.999485143261964
## 4 BLANK  0.729888031547083 -0.352273406845818
## 5 BLANK  0.119708278431491  -1.65744205011889
```

```
colnames(indi_t) <- c("genenames", "pvalue", "tstatistic")
indi_t <- indi_t %>% mutate(pvalue = as.numeric(pvalue), tstatistic = as.numeric(tstatistic))
head(indi_t$tstatistic)
```

```
## [1] -2.63047810 -3.42047449 -0.99948514 -0.35227341 -1.65744205  0.08024395
```

```
indi_p <- indi_t %>% filter(pvalue <= 0.01)
dim(indi_p)
```

```
## [1] 75  3
```

```
indi_p
```

```
##               genenames       pvalue tstatistic
## 1                 Cy5RT 4.566702e-03  -3.420474
## 2  EST,HighlysimilartoA 1.876635e-06  11.762486
## 3  FATTYACID-BINDINGPRO 3.741625e-03   3.551820
## 4               MDB0145 7.578463e-03  -3.179141
## 5               MDB0147 4.842420e-03  -3.342350
## 6               MDB0743 9.233185e-03   3.175647
## 7               MDB1376 4.212120e-03   3.417618
## 8                 BLANK 7.940787e-03   3.098509
## 9  Glucosidase,alpha,ac 9.294964e-03   3.023174
## 10                      1.608792e-03  -3.916250
## 11                      3.212788e-03  -3.549489
## 12 5'.gi|2187189|gb|AA4 3.305770e-03   3.780523
## 13 NEUROMEDIN-BRECEPTOR 7.386923e-03   3.227046
## 14 EST,WeaklysimilartoF 7.885582e-04   4.434295
## 15   Caspase7,heart-Img 5.342799e-04   4.578842
## 16                      4.997382e-03  -3.542296
## 17         lin-7homolog 3.771712e-03  -3.525414
## 18 psoriasis-associated 3.260670e-03   3.550547
## 19 RETINOICACIDRECEPTOR 5.013382e-03   3.332464
## 20 Musmusculustranscrip 3.350759e-03   3.788603
## 21                  est 3.060106e-06   9.087422
## 22                  est 3.798684e-03   3.498564
## 23                      5.940580e-03   3.248841
## 24    ApoCIII,lipid-Img 1.996372e-06  10.430072
## 25              MDB0225 9.318352e-03   3.146282
## 26                BLANK 3.754311e-03  -3.564419
## 27                  est 5.936523e-03   3.543207
## 28                BLANK 3.764140e-03   3.505280
## 29 5'.gi|1285734|gb|W11 9.124837e-03  -3.144202
## 30                 Sox5 8.533774e-03   3.077784
## 31      ApoAI,lipid-Img 3.592676e-10  23.104347
## 32 SECRETORYGRANULEPROT 3.596075e-03   3.516831
## 33 ESTs,Moderatelysimil 6.884956e-03   3.180593
## 34 Mmot1(Olf-1/EBF-like 8.353765e-03  -3.082905
## 35 ESTs,Highlysimilarto 6.068227e-06   9.018613
## 36 T-TypeCalciumChannel 9.509782e-03   3.022290
## 37 5'similartoPIR:S5501 5.829532e-03   3.365194
## 38          c-srkkinase 6.505920e-03   3.194647
## 39 APOLIPOPROTEINC-IPRE 4.197242e-03  -3.432859
## 40            novelgene 8.879113e-03  -3.053596
## 41                      5.932761e-03   3.282792
## 42                 Tbx6 5.625222e-03   3.267318
## 43 R1-typemRNAforcholin 3.976066e-03  -3.442291
## 44 subtractivehybridiza 9.891878e-03  -2.995911
## 45 MousemRNAfortypeIIDN 8.107178e-03  -3.312068
## 46 ESTs,Weaklysimilarto 1.244889e-03  -4.127311
```

```
## 47                      6.315273e-03   3.403035
## 48 EST,WeaklysimilartoC 7.193693e-09  12.982368
## 49         APXL2,5q-Img 2.874482e-03   3.984500
## 50  Egr-1mRNA,Brain-Img 3.111565e-03   3.617862
## 51              MDB0090 7.926633e-03   3.283190
## 52 Musmusculusperoxisom 5.930127e-03   3.240855
## 53 ESTs,Highlysimilarto 6.835963e-03  -3.187429
## 54 5'similartogb:X02747 2.662321e-03  -3.686857
## 55 EST,Moderatelysimila 1.987705e-03   3.791406
## 56 BRAINPROTEIND3,Brain 2.060540e-03   3.783490
## 57              MDB1430 2.519536e-03  -3.719741
## 58                BLANK 5.684980e-03  -3.264050
## 59               Idlike 6.766519e-03   3.310876
## 60               FGF12A 9.128584e-03  -3.286791
## 61 similartoyeaststerol 1.232158e-05   7.208906
## 62 ADENOSINEA1RECEPTOR, 5.573243e-03   3.542397
## 63                      5.071253e-03   3.344238
## 64   NCAM-120,Brain-Img 7.499427e-03   3.355423
## 65 CATECHOLO-METHYLTRAN 1.212987e-08  11.759068
## 66                BLANK 4.919812e-03  -3.631626
## 67                      9.489768e-04   4.250919
## 68               Olf-1 3.986948e-03  -3.504417
## 69               Meox2 4.355167e-03  -3.582474
## 70 CytosolicPLA2,heart- 7.883052e-03   3.110776
## 71 longchainfattyacidCo 1.930798e-03   3.805312
## 72                 est 9.434527e-03   3.238730
## 73               Cy3RT 9.582301e-03  -3.001422
## 74          estrogenrec 1.588935e-03  -3.957601
## 75                BLANK 8.945316e-03   3.053093
```

```
kable(head(indi_p[order(abs(indi_p$tstatistic), decreasing = TRUE),
    ], 10), caption = "Top10 Genes by gene-specific t-test")
```

Table 3: Top10 Genes by gene-specific t-test

	genenames	pvalue	tstatistic
31	ApoAI,lipid-Img	0.0000000	23.104347
48	EST,WeaklysimilartoC	0.0000000	12.982368
2	EST,HighlysimilartoA	0.0000019	11.762486
65	CATECHOLO-METHYLTRAN	0.0000000	11.759068
24	ApoCIII,lipid-Img	0.0000020	10.430072
21	est	0.0000031	9.087422
35	ESTs,Highlysimilarto	0.0000061	9.018613
61	similartoyeaststerol	0.0000123	7.208906
15	Caspase7,heart-Img	0.0005343	4.578842
14	EST,WeaklysimilartoF	0.0007886	4.434296

1.3 c1. Calculate the modified t-statistic and corresponding p-value using the samr package in R used in Homework2. How many genes are significant at the 0.01 level? List the top 10 genes that have the largest penalized t-statistics.

29 genes were significant at the 0.01 level by the samr modified t-test.

```
## prepare sam data
sam.test <- SAM(x = as.matrix(apodata), y = rep(c(1, 2), each = 8),
    genenames = ai$genenames, resp.type = "Two class unpaired",
    logged2 = T)
```

```
## perm= 1
## perm= 2
## perm= 3
## perm= 4
## perm= 5
## perm= 6
## perm= 7
## perm= 8
## perm= 9
## perm= 10
## perm= 11
## perm= 12
## perm= 13
## perm= 14
## perm= 15
## perm= 16
## perm= 17
## perm= 18
## perm= 19
## perm= 20
## perm= 21
## perm= 22
## perm= 23
## perm= 24
## perm= 25
## perm= 26
## perm= 27
## perm= 28
## perm= 29
## perm= 30
## perm= 31
## perm= 32
## perm= 33
## perm= 34
## perm= 35
## perm= 36
## perm= 37
## perm= 38
## perm= 39
## perm= 40
## perm= 41
## perm= 42
```

```
## perm= 43
## perm= 44
## perm= 45
## perm= 46
## perm= 47
## perm= 48
## perm= 49
## perm= 50
## perm= 51
## perm= 52
## perm= 53
## perm= 54
## perm= 55
## perm= 56
## perm= 57
## perm= 58
## perm= 59
## perm= 60
## perm= 61
## perm= 62
## perm= 63
## perm= 64
## perm= 65
## perm= 66
## perm= 67
## perm= 68
## perm= 69
## perm= 70
## perm= 71
## perm= 72
## perm= 73
## perm= 74
## perm= 75
## perm= 76
## perm= 77
## perm= 78
## perm= 79
## perm= 80
## perm= 81
## perm= 82
## perm= 83
## perm= 84
## perm= 85
## perm= 86
## perm= 87
## perm= 88
## perm= 89
## perm= 90
## perm= 91
## perm= 92
## perm= 93
## perm= 94
## perm= 95
## perm= 96
```

```
## perm= 97
## perm= 98
## perm= 99
## perm= 100
##
## Computing delta table
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
## 9
## 10
## 11
## 12
## 13
## 14
## 15
## 16
## 17
## 18
## 19
## 20
## 21
## 22
## 23
## 24
## 25
## 26
## 27
## 28
## 29
## 30
## 31
## 32
## 33
## 34
## 35
## 36
## 37
## 38
## 39
## 40
## 41
## 42
## 43
## 44
## 45
## 46
## 47
## 48
```

```
## 49
## 50
```

```
## get t statistic and p-value from sam.test
sam.t <- sam.test$samr.obj$tt
## calculate the 2-sided t statistic
sam.p <- pt(-abs(sam.t), 14) * 2
sam.result <- data.frame(genenames = ai$genenames, p.value = sam.p,
    t.statistic = sam.t) %>% .[order(abs(.$t.statistic), decreasing = TRUE),
    ]

kable(head(sam.result, 10), caption = "Top10 Genes by samr t-test")
```

Table 4: Top10 Genes by samr t-test

	genenames	p.value	t.statistic
2149	ApoAI,lipid-Img	0.0000000	-20.592874
540	EST,HighlysimilartoA	0.0000000	-11.049934
4139	EST,WeaklysimilartoC	0.0000000	-10.717909
5356	CATECHOLO-METHYLTRAN	0.0000000	-10.628833
1739	ApoCIII,lipid-Img	0.0000005	-8.787524
1496	est	0.0000017	-7.865276
2537	ESTs,Highlysimilarto	0.0000017	-7.847305
4941	similartoyeaststerol	0.0000165	-6.401300
947	EST,WeaklysimilartoF	0.0015262	-3.924562
954	Caspase7,heart-Img	0.0026059	-3.653656

```
## p 0.01 cutoff
n_sam <- nrow(filter(sam.result, p.value <= 0.01))
n_sam
```

```
## [1] 29
```

1.4 c2. Calculate the 'moderated' t-statistic and corresponding p-value using the limma package from BioConductor

8 genes are significant at the 0.01 level.

```
## study design, design matrix
lim_sample <- as.factor(rep(c("C", "K"), each = 8))
design <- model.matrix(~0 + lim_sample)
colnames(design) <- levels(lim_sample)

## contrast matrix
lim_contrast <- makeContrasts(diff = K - C, levels = design)

## Estimate the fold changes and standard errors by fitting a
## linear model for each gene
lim_fit <- lmFit(as.matrix(apodata), design = design)
lim_fit2 <- contrasts.fit(lim_fit, lim_contrast)
lim_fit3 <- eBayes(lim_fit2)

## get the test statistic for column of interest
```

```
lim_result <- topTable(lim_fit3, coef = "diff", genelist = ai$genenames,
    adjust.method = "BH", number = nrow(ai))

##
kable(head(lim_result, 10), caption = "Top10 genes by limma moderated t")
```

Table 5: Top10 genes by limma moderated t

	ID	logFC	AveExpr	t	P.Value	adj.P.Val	B
2149	ApoAI,lipid-Img	-4.749247	5.773086	-23.976817	0.0000000	0.0000000	14.9269328
540	EST,HighlysimilartoA	-4.572826	5.959409	-12.963071	0.0000000	0.0000005	10.8150265
5356	CATECHOLO-METHYLTRAN	-2.772249	6.617134	-12.439908	0.0000000	0.0000006	10.4483231
4139	EST,WeaklysimilartoC	-1.540431	6.817930	-11.749992	0.0000000	0.0000012	9.9246200
1739	ApoCIII,lipid-Img	-1.398874	7.081690	-9.831229	0.0000000	0.0000157	8.1890866
2537	ESTs,Highlysimilarto	-1.514718	7.077908	-9.012972	0.0000000	0.0000423	7.3031534
1496	est	-1.466135	6.971799	-8.999811	0.0000000	0.0000423	7.2881051
4941	similartoyeaststerol	-1.432454	6.640370	-7.440210	0.0000007	0.0005617	5.3097967
947	EST,WeaklysimilartoF	-0.855885	7.517514	-4.553948	0.0002495	0.1769590	0.5618636
5604		-0.549536	7.325818	-3.961031	0.0009254	0.5284860	-0.5563623

```
##
n_limma <- sum(lim_result$adj.P.Val <= 0.01)
n_limma
```

```
## [1] 8
```

1.5 Compare and contrast the results for the four methods for ranking genes. Explain the differences in how the different t-statistics are calculated.

1.5.1 Fold Change Approach

No statistical test was done by this method. This is just a descriptive way to present the difference in gene expression levels between two groups. However, the rank of genes derived by fold change method should be similar with following methods.

1.5.2 gene-specific t-test

This is a two-sided Welch's t-test on individual gene level. Within each gene, the variances were evaluated separately for each group, and the degree of freedom was calculated by Welch–Satterthwaite equation. There is no information (variance) shared across genes and no adjustment for multiple comparison. Hence, this method is not appropriate.

1.5.3 samr modified t-statistic

In this method, the pooled standard error for each gene SE_g is increased by a positive constant c, the 5% percentile of s.e. of all genes. Let R_g be the mean log ratio of one gene, the statistic now is s = Rg / (c + SEg). In this way, the standard error is increased and the t-statistic is shrinked.

1.5.4 limma method

In limma moderated t-statistic, the empirical Bayes method is used to estimte the within gene variance. And the s.e. in this method is the variance over the square root of $(1/n_1 + 1/n_2)$. This method is more conservative.

2 P-values and Multiple Testing

2.1 Calculate p-values for the t-statistics using permutations (B=12870 possibilities).

First, test the permuation with the gene1, then make the function and test with first 5 genes, finally use all gene to run the whole permutation test. In the *gtools::combinations()*. I assigned the *set = FALSE*, which means do not remove the duplicates from the 16 values for each gene. 213 genes are significant at the 0.01 level.

```
## get all numeric
ai_com <- as.matrix(data.frame(apodata, indi_t$tstatistic))
dim(ai_com)
```

```
## [1] 6384   17
```

```
N <- choose(16, 8)
## get the complement vector
"%nin%" <- Negate("%in%")

## get the combinations for gene1
gene1_com <- combinations(16, 8, ai_com[1, 1:16], set = FALSE)
dim(gene1_com)
```

```
## [1] 12870     8
```

```
## put the long vector at the first place to make the
## complement vector
ai_com[1, 1:16][ai_com[1, 1:16] %nin% gene1_com[1, ]]
```

```
##       k1       k2       k3       k4       k5       k6       k7       k8
## 8.106206 7.397413 9.318577 7.106212 8.068457 7.772820 8.431817 7.555878
```

```
## get the matrix of the other group
gene1_paired_t <- apply(gene1_com, 1, function(gene1_com) {
    ai_com[1, 1:16][ai_com[1, 1:16] %nin% gene1_com]
})

dim(gene1_paired_t)
```

```
## [1]     8 12870
```

```
gene1_paired <- t(gene1_paired_t)

## test of the combination and its complement
ai_com[1, 1:16] %in% c(gene1_com[2, ], gene1_paired[2, ])
```

```
##  [1] TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE TRUE
## [15] TRUE TRUE
```

```r
## the individual t statistic
ai_com[1, 17]
```

```
## [1] -2.630478
```

```r
# The permutaion t statistic
gene1 <- cbind(gene1_com, gene1_paired)

head(gene1)
```

```
##        [,1]     [,2]     [,3]     [,4]     [,5]     [,6]    [,7]     [,8]
## [1,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 6.391309
## [2,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 8.106206
## [3,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 7.397413
## [4,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 9.318577
## [5,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 7.106212
## [6,] 7.6699 6.721563 7.077078 5.818323 8.257974 6.753425 7.39597 8.068457
##          [,9]    [,10]    [,11]    [,12]    [,13]   [,14]    [,15]
## [1,] 8.106206 7.397413 9.318577 7.106212 8.068457 7.77282 8.431817
## [2,] 6.391309 7.397413 9.318577 7.106212 8.068457 7.77282 8.431817
## [3,] 6.391309 8.106206 9.318577 7.106212 8.068457 7.77282 8.431817
## [4,] 6.391309 8.106206 7.397413 7.106212 8.068457 7.77282 8.431817
## [5,] 6.391309 8.106206 7.397413 9.318577 8.068457 7.77282 8.431817
## [6,] 6.391309 8.106206 7.397413 9.318577 7.106212 7.77282 8.431817
##         [,16]
## [1,] 7.555878
## [2,] 7.555878
## [3,] 7.555878
## [4,] 7.555878
## [5,] 7.555878
## [6,] 7.555878
```

```r
gene1_t <- sapply(1:6384, function(row) {
    test = t.test(gene1[row, 1:8], gene1[row, 9:16], alternative = "two.sided")
    test$statistic
})
## compare the t statistic and get the p-value
p_gene1 <- sum(abs(gene1_t) > abs(ai_com[1, 17]))/N
p_gene1
```

```
## [1] 0.009168609
```

```r
## the p-value of individual t.test
indi_t$pvalue[1]
```

```
## [1] 0.01990044
```

```r
## summarise above procedures for gene1 as a function this
## function only works for genes_matrix 8 columns control, 8
## columns treatment and the last column is individual t
## statistics 6384 genes in total
combi_p <- function(gene) {
    # first 8 columns of combinations
    gene_com = combinations(16, 8, gene[1:16], set = FALSE)

    # the 2nd 8 columns of combinations
    gene_paired_t = apply(gene_com, 1, function(gene_com) {
```

```
        gene[1:16][gene[1:16] %nin% gene_com]
    })
    gene_paired = t(gene_paired_t)

    # combine to make the permutation matrix
    gene_matrix = cbind(gene_com, gene_paired)

    # permutation t statistic vector
    gene_t = sapply(1:6384, function(row) {
        test = t.test(gene_matrix[row, 1:8], gene_matrix[row,
            9:16], alternative = "two.sided")
        test$statistic
    })

    # individual t-statistic
    t = gene[17]

    ## calculate the p-value
    p_gene = sum(abs(gene_t) >= abs(t))/N
    return(p_gene)
}

## test run and system.time
matrix(apply(head(ai_com), 1, combi_p), ncol = 1, byrow = TRUE)
```

```
##              [,1]
## [1,] 0.009168609
## [2,] 0.003574204
## [3,] 0.166200466
## [4,] 0.362237762
## [5,] 0.060217560
## [6,] 0.465345765
```

```
system.time(matrix(apply(head(ai_com), 1, combi_p), ncol = 1,
    byrow = TRUE))
```

```
##    user  system elapsed
##   4.312   0.000   4.313
```

```
## compare with individual p-value
indi_t$pvalue[1:6]
```

```
## [1] 0.019900439 0.004566702 0.334997882 0.729888032 0.119708278 0.937245530
```

```
head(indi_t)
```

```
##   genenames      pvalue  tstatistic
## 1     Cy3RT 0.019900439 -2.63047810
## 2     Cy5RT 0.004566702 -3.42047449
## 3     mSRB1 0.334997882 -0.99948514
## 4     BLANK 0.729888032 -0.35227341
## 5     BLANK 0.119708278 -1.65744205
## 6     BLANK 0.937245530  0.08024395
```

```
## the full permutation
p_per <- apply(ai_com, 1, combi_p)
```

```
p_per <- matrix(p_per, ncol = 1, byrow = TRUE)

## p values with genenames
p_per <- data.frame(ai$genenames, p_per)
colnames(p_per) <- c("genenames", "pvalue")
## get the p value 0.01
p_per_1 <- p_per %>% filter(pvalue <= 0.01)
write.csv(p_per_1, "permutationP.csv")
```

2.2 multiple testing adjustment methods to gene-specific t-test

2.2.1 Bonferroni

This method is family-wise error rate (FWER) correction method, and the adjusted-p is calculated by multiplication of each p-value by the numer of tests conducted. It is very conservative.

2.2.2 Sidak

This method also belongs to FWER corrections, and the adjusted-p is $(1-(1-p_{t-test})^n)$. This method is slightly less conservative compared with the Bonferroni method.

2.2.3 Holm step-down procedure

This procedure also controls for FWER but takes the rank of of p-values into account. For all m p-values where $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(m)}$, p-values from the smallest to $p_{(j)} < \alpha/(m-j+1)$ will be rejected. This method is less conservative compared with the above two since larger p values are dealt with less stringent conditions.

2.2.4 Benjamini-Hochberg procedure

The BH step-up procedure is False Discovery Rate (FDR) correction method and also ranks all m p-values where $p_{(1)} \leq p_{(2)} \leq ... \leq p_{(m)}$. The null hypothesis will be rejected for $p_{(j)} \leq \frac{j}{m}q$.

2.2.5 Number

For Bonferroni, Sidak and Holm, 3 genes are significant at 0.01. For BH, 8 genes are significant at 0.01.

```
# the original t-test 1b
n = nrow(ai)
head(indi_t, 3)
```

```
##   genenames      pvalue tstatistic
## 1     Cy3RT 0.019900439 -2.6304781
## 2     Cy5RT 0.004566702 -3.4204745
## 3     mSRB1 0.334997882 -0.9994851
```

```
## the Bonferroni is n*p psidak_FWER = (1 - (1- pvalue)^n)
## function for holm and BH
holm <- function(p) {
    lp = length(p)
    i = seq_len(lp)
```

```
    o = order(p)
    p = p[o]  ## now p is increasing
    ro = order(o)  # the rank(p)
    q <- pa <- rep.int(min(lp * p/i), lp)
    for (j in (lp - 1):2) {
        ij = seq_len(lp - j + 1)
        i2 = (lp - j + 2):lp
        q1 = min(j * p[i2]/(2:j))
        q[ij] = pmin(j * p[ij], q1)
        q[i2] = q[lp - j + 1]
        pa <- pmax(pa, q)
    }
    pmax(pa, p)[ro]
}

BH <- function(p) {
    lp = length(p)
    i = lp:1
    o = order(p, decreasing = TRUE)
    ro = order(o)  # rank of p, increasing
    pmin(1, cummin(lp/i * p[o]))[ro]
}

indi_adjust <- indi_t %>% mutate(pBonferroni_FWER = pmin(1, n *
    pvalue), psidak_FWER = (1 - (1 - pvalue)^n), pholm_FWER = holm(pvalue),
    pBH_FDR = BH(pvalue))

## results
sum(indi_adjust$pBonferroni_FWER <= 0.01)
```

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

```
sum(indi_adjust$psidak_FWER <= 0.01)
```

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

```
sum(indi_adjust$pholm_FWER <= 0.01)
```

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

```
sum(indi_adjust$pBH_FDR <= 0.01)
```

```
## [1] 8
```

```
head(indi_t)
```

```
##   genenames      pvalue  tstatistic
## 1     Cy3RT 0.019900439 -2.63047810
## 2     Cy5RT 0.004566702 -3.42047449
## 3     mSRB1 0.334997882 -0.99948514
## 4     BLANK 0.729888032 -0.35227341
## 5     BLANK 0.119708278 -1.65744205
## 6     BLANK 0.937245530  0.08024395
```

```
q_value <- qvalue(indi_t$pvalue)
## cutoff
sum(q_value$qvalues < 0.01)
```

```
## [1] 8
```

```
##
q_value$pi0
```

```
## [1] 0.8594059
```

2.3 Calculate q-values using the qvalue library.

8 genes have a q-value less than 0.01.
π_0 is the proportion of true null hypotheses, which is estimated by the whole gene expression dataset. In this case, it is 0.8594059.