Homework 3 - BIOS 7649

Dominic Adducci

Question 1: T-statistics

Part A

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).

There are 8 mice in the control group (C57B1/6 strain) and 8 mice in the experimental group (apo AI knockout). In order to calculate the fold change, either positive or negative between the two groups, the averages for each group needs to be calculated for each gene.

Table 1: 10 Largest Absolute Fold Changes

Gene Name	Difference
ApoAI,lipid-Img	-4.74924672295693
EST,HighlysimilartoA	-4.57282566300963
CATECHOLO-METHYLTRAN	-2.77224890674299
EST,WeaklysimilartoC	-1.54043051734832
ESTs,Highlysimilarto	-1.51471758262052
est	-1.46613543694816
similartoyeaststerol	-1.43245389359317
ApoCIII,lipid-Img	-1.39887354565181
psoriasis-associated	-1.2567134509252
Cy3RT	1.19328632324561

Table 1 illustrates the largest absolute difference in fold change. The difference was found by subtracting the averaged expression of the control group from the averaged expression of the experimental group. This means that positive values indicate increased expression in the experimental group, and negative valuees indicate increased expression in the control group.

Part B

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-statistic and their corresponding p-value.

Table 2: 10 Largest t-statistics

Gene Name	t-statistic	p-value
ApoAI,lipid-Img	23.104	3.59e-10
EST,WeaklysimilartoC	12.982	7.19e-09
EST,HighlysimilartoA	11.762	1.88e-06
CATECHOLO-METHYLTRAN	11.759	1.21e-08
ApoCIII,lipid-Img	10.430	2.00e-06
est	9.087	3.06e-06
ESTs,Highlysimilarto	9.019	6.07e-06
similartoyeaststerol	7.209	1.23e-05
Caspase7,heart-Img	4.579	0.000534
EST,WeaklysimilartoF	4.434	0.000789

Table 2 illustrates the top 10 genes that have the largest t-statistic and their corresponding p-values. There are 75 genes that are significant at the 0.01 level. Results were generated using a 2 sided t-test between the control and experimental groups.

Part C: Alternative 't-statistics'

Part C.i

Calculate the modified 't-statistic' and corresponding p-value using the **samr** package in R used in Homework 2. How many genes are significant at the 0.01 level? List the top 10 genes that have the largest 'penalized' t-statistics.

Table 3: 10 Largest modified t-statistics - samr

Gene Name	Score (Modified t-statistic)
ApoAI,lipid-Img	-20.593
EST,HighlysimilartoA	-11.050
EST,WeaklysimilartoC	-10.718
CATECHOLO-METHYLTRAN	-10.629
ApoCIII,lipid-Img	-8.788
est	-7.865
ESTs,Highlysimilarto	-7.847
similartoyeaststerol	-6.401
EST,WeaklysimilartoF	-3.925
Caspase7,heart-Img	-3.654

Table 3 shows the 10 largest modified t-statistics which were calculated using the **samr** package. Because a two-sided t-test was used the absolute value of the t-score determines the significance level. There are 72 genes that are significant at the 0.01 level. Because there are two groups with 8 measurements in each the number of degrees of freedom is 14, which corresponds to a cutoff for the modified t-statistic of 2.624.

Part C.ii

Calculate the 'moderated' t-statistic and corresponding p-value using the **limma** package from BioConductor. How many genes are significant at the 0.01 level? List the top 10 genes that have the largest 'penalized'

t-statistics.

Table 4: 10 Largest moderated t-statistics - Limma

Gene Name	Modified t-statistic	p-value	Adj. p-value (BH)
ApoAI,lipid-Img	-23.977	4.74e-15	3.03e-11
EST,HighlysimilartoA	-12.963	1.56e-10	4.99e-07
CATECHOLO-METHYLTRAN	-12.440	3.05e-10	6.48e-07
EST,WeaklysimilartoC	-11.750	7.61e-10	1.21e-06
ApoCIII,lipid-Img	-9.831	1.23e-08	1.57e-05
ESTs,Highlysimilarto	-9.013	4.54e-08	4.23e-05
est	-9.000	4.64e-08	4.23e-05
similartoyeaststerol	-7.440	7.04e-07	0.000562
EST,WeaklysimilartoF	-4.554	0.000249	0.176959
	-3.961	0.000925	0.528486

Table 4 shows the 10 largest moderated t-statistics which were calculated using the **limma** package. There are 93 genes that are significant at the 0.01 level.

Part D

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

Many genes overlapped in significance between the four different methods. Between method 2 (Table 2), largest t-statistic, and method 3 (Table 3), the modified t-statistic, both top 10 tables had the same genes with the only difference being in the order of some genes. Method 4, the modified t-statistic, had almost all the same genes in its top 10 table (Table 4) as methods 2 and 3, with the exception of an unnamed gene being present instead of Capase7,heart-Img. Method 1 (Table 1), had many of the same genes present in its top 10 table as the other tables, with the exception of having psoriasis-associated and Cy3RT genes instead of Caspase7,heart-Img and ESTS,Highlysimilarto.

Additionally, the ApoAI,lipid-Img gene that was knocked out between the control and experimental groups was the most significant gene in all top 10 tables. This gene showed decreased expression between the experimental and control groups (Table 1), which makes sense as this gene was specifically knocked-out between these two groups. Method 2 simply calculates t-scores by applying a two-sample t-test to each gene individually.

The **samr** package calculates the modified t-statistic by dividing the difference of the mean of the two groups by s_i, which shifts the test statistic. Class labels are randomly permutated in order to estimate the null distribution of the scores, as well as the standard deviation for each specific gene. The absolute value of the resulting modified t-statistic is smaller than the unpenalized t-statistic.

The **limma** package calculates the moderated t-statistic using the parametric empirical Bayes method. This method utilizes information between different genes, moderating the residual variances. These moderated residuals are function of both the variance for a specific gene, the gene-wise estimator, and the global variability across all genes. This method allows for an increase in the effective degrees of freedom. The benefit of this approach is that inference on gene expression in reliable and stable for small number of replicates, a common scenario in gene expression analysis.

Question 2: P-values and Multiple Testing

Part A

Calculate p-values for the t-statistics using permutations (B = 12870 possibilities). How many genes are significant at the 0.01 level?

The permutation function utilized here worked by selecting each row from the original array and calculating every permutation for measurements between the control and experimental groups. Because there are 16 different measurements this means there are 12,870 different combinations for each gene. The p-value is calculated by determining the number of absolute value permutation t scores that are larger than the absolute value of the t score from the original data frame. This number is then divided by the number of permutations (12,870) to return the permutation p-value. After running this procedure there were 117 genes that are significant at the 0.01 level.

Part B

Apply the following multiple testing adjustment methods to the original t-statistic (#1b) p-values and list the number of genes at a cutoff level of 0.01. Compare and contrast the different methods. Why are some more or less conservative than others?

Part i: Bonferroni

The Bonferroni correction is the original significance level divided by the number of tests. The original α was 0.01, and the number of genes being analyzed is 6,384.

$$Bonferroni\ \alpha - level = \frac{\alpha}{n} = \frac{0.01}{6,384} = 1.57 \times 10^{-6}$$

After applying the Bonferroni correction there were 3 genes significant. The Bonferroni method is generally the most conservative, as the significance level is divided by the total number of hypothesis under consideration.

Part ii: Sidak

The Sidak correction is applied through the following equation:

$$Sidak\ \alpha - level = 1 - (1 - \alpha)^{\frac{1}{m}} = 1 - (1 - 0.01)^{\frac{1}{6387}} = 1.57 \times 10^{-6}$$

- Where $\alpha = 0.01$, the original significance level, and $m = 6387$, the number of genes being analyzed.

After applying the Sidak correction there are 3 genes which are significant. The Sidak correction is generally close to the Bonferroni correction, and in this case rounding to three significant figures returns the same significance level.

Part iii: Holm step-down procedure

The Holm step-down procedure is applied by first ordering the p-values in ascending fashion, and starting at the top of this ordered vector. The subsequent recalculated p-values follow the general formula:

$$p - value_k < \frac{\alpha}{m + 1 - k}$$

* Where $\alpha = 0.01$, the original significance level, $m = 6387$, the number of genes being analyzed, and k is the order in the vector of a particular p-value, where the smallest values starts at $k = 1$. This procedure continues until the $p - value_k$ currently being assessed is no longer $< \frac{\alpha}{m+1-k}$. The procedure is then finished.

The number of p-values that satisfy the inequality are the number of test where the null hypothesis is rejected after applying the Holm step-down procedure.

After applying the Holm step-down procedure there were 3 significant genes. This is the same as the previous two methods. Because the number of comparisons is in the denominator for the Holm step-down procedure this value will tend to dominate if it is large, resulting in the final correction being similar to the Bonferroni method.

Part iv: Benjamini-Hochberg procedure

The Benjamini-Hochberg procedure is applied by first ordering the p-values in ascending fashion and assigning ranks to the p-values based on their location in the data frame (smallest p-value is 1, second smallest is 2, third smallest is 3, etc.). Next, you calculated the Benjamini-Hochberg critical value using the formula:

$$BH\ Critical\ Value = \frac{i}{m}Q = \frac{i}{6384} \times 0.01$$

Where

- i: p-value's rank.
- m: total number of test, in this case there are 6,384.
- Q: false discovery rate.

After performing this procedure the highest ranked p-value that is smaller than the Benjamini-Hochberg critical value is found. All p-values above that p-value are considered significant even if a particular p-value is not smaller than the Benjamini-Hochberg critical value.

After applying the Benjamini-Hochberg correction there were 8 significant genes. This method was the least conservative among the four tested. The significance level is calculated individually for each gene based on rank. The rank is a multiplier for the significance level, which increases the number of significant genes.

Part C

Calculate q-values using the **qvalue** library. How many genes have a q-value less than 0.01. What is the π_0 parameter and what value is estimated using this package?

After using the qvalue function the number of q-values that are less than 0.01 are 8. The π_0 parameter is the estimated proportion of null p-values, and is returned from the function as 0.859.

CODE

```
library(tidyverse)
library(impute)
library(limma)
library(samr)
library(kableExtra)
library(qvalue)
library(gtools)
library(pbapply)


### START QUESTION 1 CODE ###

# Laptop Location:
# Desktop: "C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/
# Laptop: "C:/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/hw3data/hw3arraydata.tx
array_data <- read.table("C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BIOS765
                         blank.lines.skip = FALSE)


# Data Location:
#Desktop: "C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/h
#Laptop: "C:/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/hw3data/hw3genenames.txt
genenames_data <- read.table("C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BI

colnames(genenames_data) <- "gene_name"

# Combining gene names with array data.
array_genes <- cbind(genenames_data,array_data)


## START QUESTION 1 PART A CODE ##

# Making a function which averages the subjects in each group and
# finds the difference.
fold_change <- function(index,data_frame){
  # index: Refers to the index which is currently being assessed
  # in the data frame.
  # data_frame: The data frame with genes names and measurements.

  # Extracting current row.
  current_row <- data_frame[index,]

  control <- sum(current_row[,grepl("c",names(current_row))])/8
  experimental <- sum(current_row[,grepl("k",names(current_row))])/8
  gene_name <- current_row[,grepl("gene_name",names(current_row))]

  # Difference calculated by subtracting control group from experimental
  # group, e.g., positive values indicate experimental group is has
  # higher expression, negative values indicate control group has
  # higher expression.
  difference <- experimental - control
```

```
  result <- c(gene_name,difference)

  return(result)

}


# Applying function to entire data frame.
difference_result <- sapply(seq(1,nrow(array_genes),by = 1),
                            function(x) fold_change(x,array_genes)) %>%
  t() %>% as.data.frame()

colnames(difference_result) <- c("gene_name","difference")

# Ordering data frame by absolute difference.
diff_order <- difference_result %>% arrange(desc(abs(as.numeric(difference))))

diff_order_table <- kbl(head(diff_order,10),
                        col.names = c("Gene Name","Difference"),
                        caption = "10 Largest Absolute Fold Changes",
                        booktabs = TRUE, align = "lc") %>%
  kable_styling(latex_options = "HOLD_position")

diff_order_table

## FINISH QUESTION 1 PART A CODE ##


## START QUESTION 1 PART B CODE ##

# Making a function that performs a t-test between the experimental and
# control group for each gene.
fold_ttest <- function(index,data_frame){
  # index: Refers to the index which is currently being assessed
  # in the data frame.
  # data_frame: The data frame with genes names and measurements.

  # Extracting current row.
  current_row <- data_frame[index,]

  control <- current_row[,grepl("c",names(current_row))]
  experimental <- current_row[,grepl("k",names(current_row))]
  gene_name <- current_row[,grepl("gene_name",names(current_row))]

  # Running t-test on the experimental and control group.
  ttest_result <- t.test(control,experimental,alternative = "two.sided")

  # Extracting relevant information from t-test.
  statistic_value <- ttest_result$statistic
  p_value <- ttest_result$p.value

  result <- c(gene_name,statistic_value,p_value)
```

```
  return(result)

}


# Applying function to entire data frame.
ttest_result <- sapply(seq(1,nrow(array_genes),by = 1),
                       function(x) fold_ttest(x,array_genes)) %>%
  t() %>% as.data.frame()


colnames(ttest_result) <- c("gene_name","t_statistic","p_value")

# Ordering result by t-statistic.
ttest_order <- ttest_result %>% arrange(desc(as.numeric(t_statistic))) %>%
  mutate(t_statistic = as.numeric(t_statistic),
         p_value = as.numeric(p_value),
         p_value = format.pval(p_value,digits = 3)) %>%
  mutate_if(is.numeric,round,digits = 3)

ttest_tbl <- kbl(head(ttest_order,10),
                 col.names = c("Gene Name","t-statistic","p-value"),
                 caption = "10 Largest t-statistics",
                 booktabs = TRUE, align = "lc") %>%
  kable_styling(latex_options = "HOLD_position")

ttest_tbl

# Determining the number of genes that are significant at the 0.01 level.
ttest_sig <- nrow(ttest_order %>% filter(as.numeric(p_value) <= 0.01))

## FINISH QUESTION 1 PART B CODE ##


## START QUESTION 1 PART C.i CODE ##

# Setting up list for the samr object. The data was already logged2
# transformed, which is noted in the list.
samr_list <- list(x = as.matrix(array_data),y = c(rep(1,8),rep(2,8)),
                  geneid = genenames_data[,1], genenames = genenames_data[,1],
                  logged2 = TRUE)

# Setting up samr object. Assay type is array. testStatistic appears
# defaulted to "standard".
samr.obj <- samr(samr_list,resp.type = "Two class unpaired", nperms = 500,
                 assay.type = "array")

delta.table <- samr.compute.delta.table(samr.obj)

# Computing significant genes table. all genes selected in order to then
# select out the top 10.
samr_test <- samr.compute.siggenes.table(samr.obj,del = 1,all.genes = TRUE,
                                         data = samr_list,
```

```
                                        delta.table)

# Extracting information from the above variable to put in a table.
genes_low_samr <- as.data.frame((samr_test$genes.lo)[,
                                                     c("Gene ID","Score(d)")])%>%
  mutate(`Score(d)` = as.numeric(`Score(d)`))

genes_up_samr <- as.data.frame((samr_test$genes.up)[,
                                                    c("Gene ID","Score(d)")])%>%
  mutate(`Score(d)` = as.numeric(`Score(d)`))

# Combining up and down regulated genes and ordering by t-score.
genes_samr <- rbind(genes_low_samr,genes_up_samr)

genes_samr <- genes_samr[order(abs(genes_samr$`Score(d)`),decreasing = TRUE),]

samr_test_tbl <- kbl(genes_samr[c(1:10),],
                     caption = "10 Largest modified t-statistics - samr",
                     col.names = c("Gene Name","Score (Modified t-statistic)"),
                     booktabs = T, align = "lc", digits = 3) %>%
  kable_styling(latex_options = "HOLD_position")

# Determining how many genes are significant at the 0.01 level.
# 14 degrees of freedom means the critical t-score is 2.624.
genes_samr_sig <- nrow(genes_samr %>% filter(abs(`Score(d)`) >= 2.624))



samr_test_tbl

## FINISH QUESTION 1 PART C.i CODE ##


## START QUESTION 1 PART C.ii CODE ##

# Code based on page 42 of limma user guide.

# Setting up design matrix for control (c) and experimental (k) groups.
design_mat <- cbind(Con = c(1,1,1,1,1,1,1,1,0,0,0,0,0,0,0,0),
                    Exp = c(0,0,0,0,0,0,0,0,1,1,1,1,1,1,1,1))

# Setting up contrast
limma_fit <- lmFit(object = array_data, design = design_mat)
cont.mat <- makeContrasts(ExpvsCon = Exp - Con, levels = design_mat)
limma_fit2 <- eBayes(contrasts.fit(limma_fit,cont.mat))

# Extracting top 10 results.
limma_results <- topTable(limma_fit2, number = 6384)

# Using row number to extract gene name for top 10 results.
limma_genes <- genenames_data[rownames(limma_results),]

limma_table_df <- cbind(limma_genes,limma_results[,c(3,4,5)]) %>%
```

```
  mutate(P.Value = format.pval(P.Value,digits = 3),
         adj.P.Val = format.pval(adj.P.Val,digits = 3))

row.names(limma_table_df) <- NULL

limma_table <- kbl(head(limma_table_df,10),
                   caption = "10 Largest moderated t-statistics - Limma",
                   col.names = c("Gene Name","Modified t-statistic",
                                 "p-value","Adj. p-value (BH)"),
                   booktabs = TRUE, align = "lccc",digits = 3) %>%
  kable_styling(latex_options = "HOLD_position")

limma_table

genes_limma_sig <- nrow(limma_results %>% filter(as.numeric(P.Value) <= 0.01))

## FINISH QUESTION 1 PART C.ii CODE ##

### FINISH QUESTION 1 CODE ###


### START QUESTION 2 CODE ###

## START QUESTION 2 PART A CODE ##

permutation_test <- function(index,gene_data,ttest_statistic){
  # index: Refers to the index which is currently being assessed
  # in the data frame.
  # gene_data: The data frame with genes names and measurements.
  # ttest_statistic: The data frame that holds the results of the t-test from
  # part 1B.

  # Extracting current row from data frame.
  current_row <- gene_data[index,]

  # Extracting gene name.
  gene_name <- current_row[,"gene_name"]

  # Creating possible combinations
  current_comb <- combinations(16,8,unlist(current_row[,c(2:17)]))

  # Selecting out the initial t-value.
  t_value <- as.numeric(ttest_statistic[index,"t_statistic"])

  # Initializing a count to keep track of number of t_star values > t_value.
  count <- 0

  for(i in 1:nrow(current_comb)){
    control <- current_comb[i,]
    experimental <- setdiff(as.numeric(current_row[,c(2:17)]),control)
    t_test_perm <- t.test(control,experimental,alternative = "two.sided")
    t_star <- as.numeric(t_test_perm$statistic)
    if(abs(t_star) > abs(t_value)){
```

```
      count = count + 1
    }
    else count = count
  }

  permutation_p <- count/nrow(current_comb)

  result <- c(gene_name,permutation_p)

  return(result)
}




# Applying permutation function.
perm_test <- pbsapply(seq(1,nrow(array_genes),by = 1),
                              function(x) permutation_test(x,array_genes,
                                                           ttest_result))

# Saving results of the permutation to an RDS file.
#saveRDS(perm_test,"C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Hom


# Data Location:
# Desktop: "C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/
# Laptop: "C:/Biostatistics Masters Program/Spring 2024/SMG-BIOS7659/Homework 3/hw3data/permutation_res

# Loading in previously run permutation data.
permutation_results <- readRDS("C:/Users/domin/Documents/Biostatistics Masters Program/Spring 2024/SMG-
  as.data.frame %>% mutate(V2 = as.numeric(V2))

colnames(permutation_results) <- c("gene_name","p_value")


permutation_sig <- nrow(permutation_results %>% filter(p_value <= 0.01))

## FINISH QUESTION 2 PART A CODE ##


## START QUESTION 2 PART B CODE ##

# Converting p-value into a numeric value.
ttest_numeric <- ttest_result %>%
  mutate(p_value = as.numeric(p_value))

ttest_numeric <- ttest_numeric[order(ttest_numeric$p_value),]

# Calculating Bonferroni correction.
bonf_sig <- 0.01/6384

bonferroni_corr <- nrow(ttest_numeric %>% filter(p_value <= bonf_sig))
```

```r
# Calculating Sidak correction.
sidak_sig <- 1 - (1 - 0.01)^(1/6387)

sidak_corr <- nrow(ttest_numeric %>% filter(p_value <= sidak_sig))


# Making a function to use the Holm step-down procedure.
holm_sdp <- function(data_frame,alpha){
  # data_frame: The data frame containing the previously calculated p-values.
  # alpha: The original significance level

  m = nrow(data_frame)
  # Initializing the number of significant tests
  # after applying the Holm procedure.
  count = 0

  for(k in 1:m){
    p_value <- data_frame[k,"p_value"]
    holm_corr <- alpha/(m + 1 - k)
    if(p_value < holm_corr){
      count <- count + 1
    }
    else break
  }
  sig_data <- data_frame[c(1:count),]
  return(sig_data)
}


# Using Holm step-down procedure function.
holm_df <- holm_sdp(ttest_numeric,0.01)

holm_corr <- nrow(holm_df)


# Applying the Benjamini-Hochberg procedure.
bh_data_frame <- ttest_numeric

# Adding a row of ranks.
bh_data_frame$rank <- seq.int(nrow(bh_data_frame))

# Calculating critical values, where alpha = 0.01.
bh_data_frame$critical_value <- (bh_data_frame$rank/6384)*0.01

# Selecting out significant genes. All p-values above the highest p-value
# that is smaller than the critical value are also smaller than their
# corresponding critical values.
bh_corr <- nrow(bh_data_frame %>% filter(p_value < critical_value))

## FINISH QUESTION 2 PART B CODE ##


## START QUESTION 2 PART C ##
```

```r
qvalues <- qvalue(p = ttest_numeric$p_value)

# Calculating number of q_values less than 0.01.
qvalues_sig <- sum(qvalues$qvalues < 0.01)

# Extracting the pi0 values.
pi0 <- round(qvalues$pi0,3)

## FINISH QUESTION 2 PART C CODE ##

### FINISH QUESTION 2 CODE ###
```