QCB455/COS551 Homework 2

eQTL, Polygenic Risk Score, RNAseq, and CRISPR

William Svoboda (wsvoboda)

Last edited October 07, 2021

Contents

Collaboration Statement

I talked with Brendan McManamon (bm18, student), Debby Park (debbyp, student), and Stephanie Monson (smonson, student) about this homework.

1 eQTL

1. Read the dataset into R and average the two replicates for each individual.

```
# Create a data frame corresponding to each data set
rma_data <- read.table("./hw2data/q1/RMA_Dataset.txt", header = TRUE)
genotypes_data <- read.table("./hw2data/q1/Genotypes.txt", header = TRUE)

# Average the two replicates for each individual in the RMA data set
rma_data_averaged <- data.frame(cbind(rma_data$ProbeSet, sapply(seq(2, ncol(rma_data),
    2), function(i) {
    rowMeans(rma_data[, c(i, i + 1)], na.rm = T)
}))) %>%
    `colnames<-`(c(colnames(rma_data[1]), genotypes_data$SampleID)) %>%
    mutate(across(CEU1:YRI8, as.numeric))
```

2. Perform ANOVA using the genotype data from all 16 individuals for each probeset (i.e., expression data for each gene measured). What is the most significant probeset and what is its p-value?

```
# Genotype list will be same for each row
g <- as.character(genotypes_data$Genotype)

# One-way ANOVA given a vector of gene expressions y and a vector of genotype
# data g
perform_anova <- function(y, g) {
    fit <- aov(y ~ g)
    p <- summary(fit)[[1]][["Pr(>F)"]][[1]]
    return(p)
}

# Perform ANOVA for each probeset (row) in expression data NOTE: I tried
# vectorizing this but it made my head hurt
for (i in 1:nrow(rma_data_averaged)) {
    y <- unlist(rma_data_averaged[i, 2:17])
    rma_data_averaged$pvalues[i] <- perform_anova(y = y, g = g)
}

# Select the probeset with the lowest p-value, which corresponds with the
# greatest significance
most_significant_probeset <- rma_data_averaged %>%
    slice(which.min(pvalues)) %>%
    select(ProbeSet, pvalues)
```

- **The most significant probeset is 212751_at and its p-value is** 3.6981306×10^{-5}**.**

3. This analysis is problematic for two important reasons. What are they and how would you address them to make the inference more robust?

- **Not only is the sample size extremely small—with only eight individuals from each racial group—but there is also a confounding factor. The analysis treats the two origins (European and African-American individuals) as equivalent, rather than separating them out. To address this, a more complex linear model could use another variable to account for origin.**

2 A Polygenic Risk Score for Eye Color

1. Compute p_{brown} for an individual with genotype $0 = X_1 = X_2 = X_3 = X_4 = X_5 = X_6$.

```
# Given estimates for brown eye liability
alpha_1 <- 3.94
alpha_2 <- 0.65

# Compute pbrown NOTE: Since Xk is all 0 the summations for each log-odd is
# just 0. This means each log-odd ratio is equal to a1 and b1 respectively
p_brown <- 1/(1 + exp(alpha_1) + exp(alpha_2))
```

- p_{brown} **is equal to 0.0184046**

2. Under this model, which genotype has the highest value of $\ln(\frac{p_{brown}}{1-p_{brown}})$?

```
# Given estimates
beta_1 <- c(-4.81, 1.4, -0.58, -1.3, 0.47, 0.7)
beta_2 <- c(-1.78, 0.87, -0.03, -0.5, 0.27, 0.73)

count <- 6
all_genotypes <- rep(list(0:2), count)
all_genotypes <- as.data.frame(expand.grid(all_genotypes))

k <- 1:6
for (i in 1:nrow(all_genotypes)) {
    # Calculate log odds ratios
    pbl_pbr_log <- alpha_1 + sum(beta_1[k] * all_genotypes[i, k])
    po_pbr_log <- alpha_2 + sum(beta_2[k] * all_genotypes[i, k])

    # Calculate p-brown using ratios
    p_brown_i <- 1/(1 + exp(pbl_pbr_log) + exp(po_pbr_log))

    # Calculate log-ratio using p-brown
    p_brown_log_ratio <- log((p_brown_i)/(1 - p_brown_i))
    all_genotypes$log_p_brown[i] <- p_brown_log_ratio
}

highest_val_genotype <- all_genotypes %>%
    slice(which.max(log_p_brown))

knitr::kable(highest_val_genotype, caption = "Genotype with highest value")
```

Table 1: Genotype with highest value

Var1	Var2	Var3	Var4	Var5	Var6	log_p_brown
2	0	2	2	0	0	3.965798

- **This code finds the genotype with the highest value of** $\ln(\frac{p_{brown}}{1-p_{brown}})$ **through an exhaustive search.**

3. (a) Load the dataset 'geno_prs.txt'

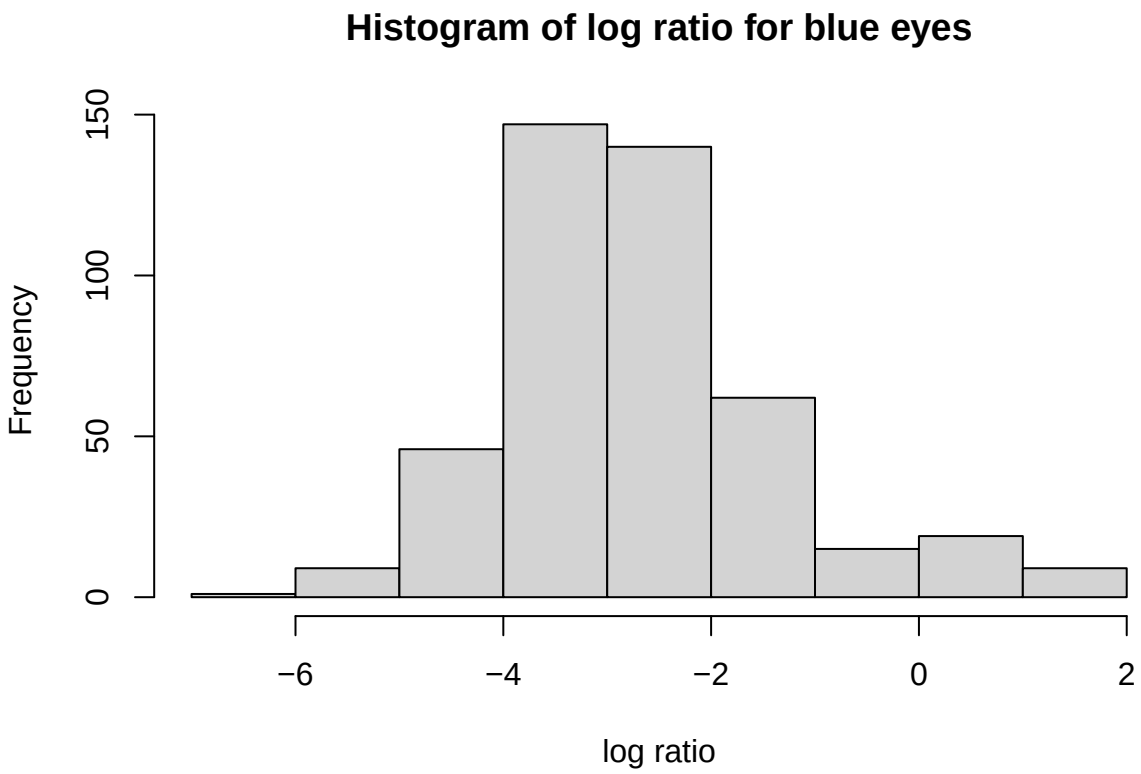
```
geno_prs <- read.csv("./hw2data/q2/geno_prs.txt", header = TRUE)
```

(b) Compute $\ln(\frac{p_{brown}}{1-p_{brown}})$

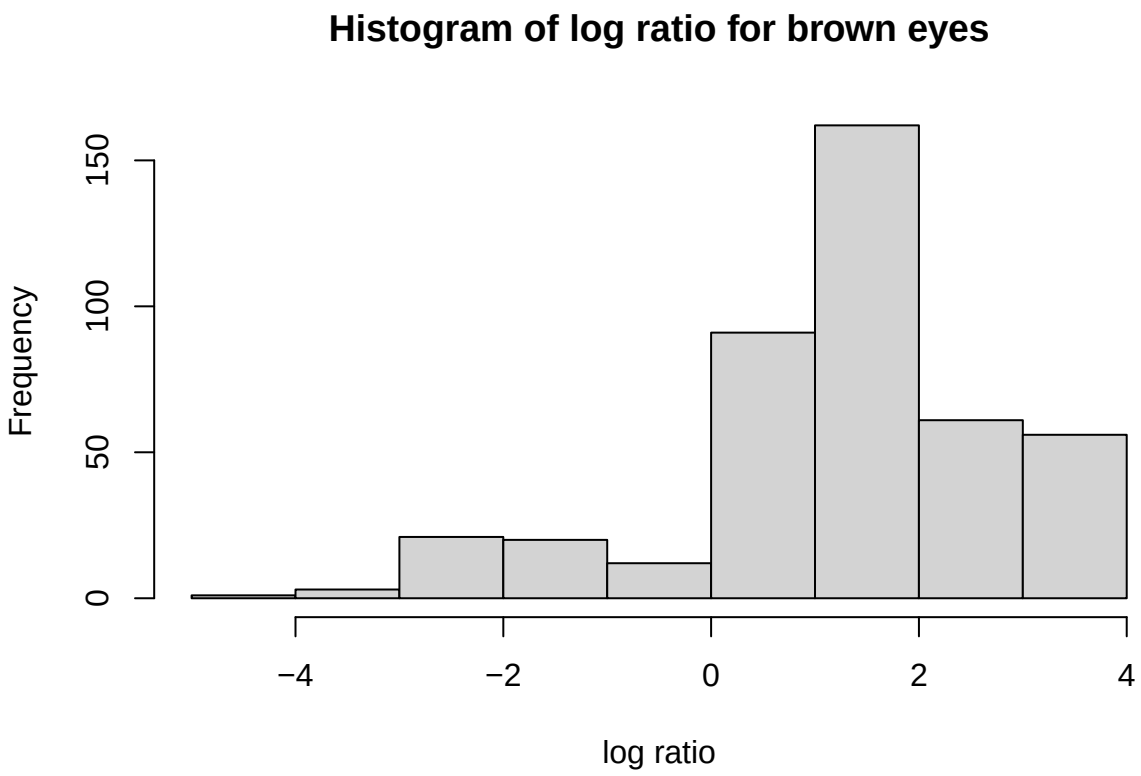
```r
k <- 1:6
for (i in 1:nrow(geno_prs)) {
    # Calculate log odds ratios
    pbl_pbr_log <- alpha_1 + sum(beta_1[k] * geno_prs[i, k])
    po_pbr_log <- alpha_2 + sum(beta_2[k] * geno_prs[i, k])

    # Calculate p-brown using ratios
    p_brown_i <- 1/(1 + exp(pbl_pbr_log) + exp(po_pbr_log))

    # Calculate log-ratio using p-brown
    p_brown_log_ratio <- log((p_brown_i)/(1 - p_brown_i))
    geno_prs$log_p_brown[i] <- p_brown_log_ratio
}
```

(c) Plot a histogram of $\ln(\frac{p_{brown}}{1-p_{brown}})$ for individuals with each eye color.

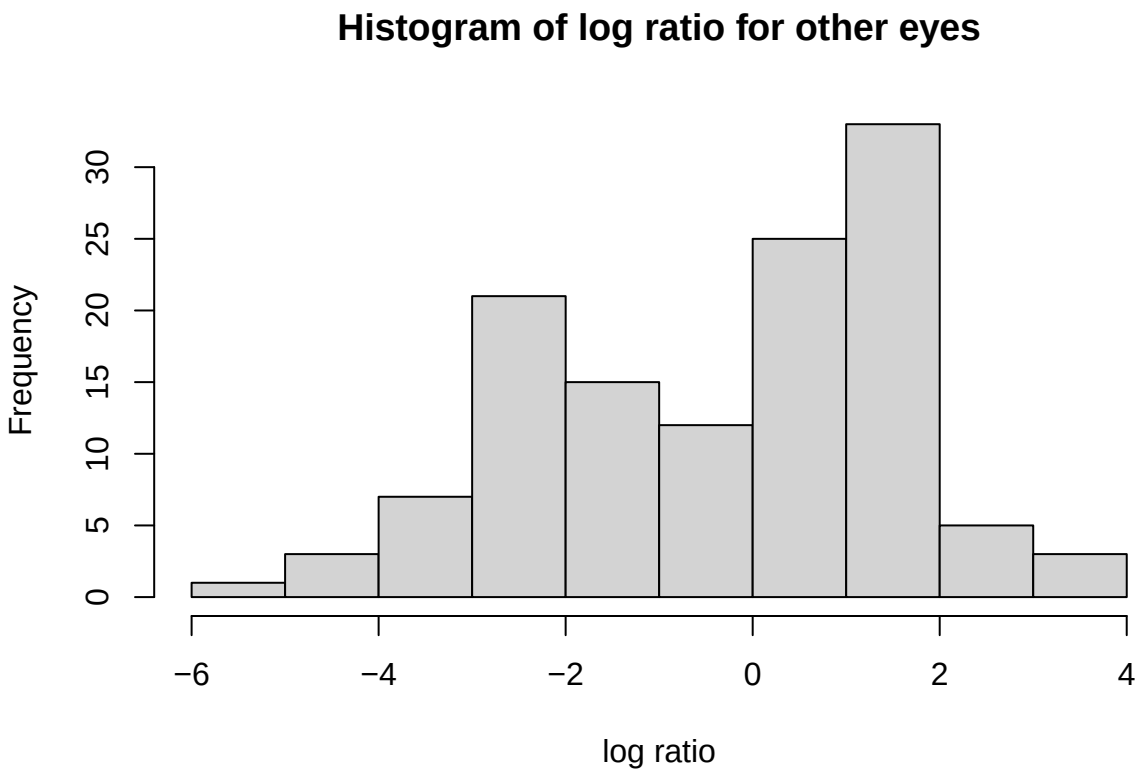
```r
# Separate each individual into separate data frames by eye color
blue_eyes <- geno_prs %>%
    slice(which(Y == "blue"))
brown_eyes <- geno_prs %>%
    slice(which(Y == "brown"))
other_eyes <- geno_prs %>%
    slice(which(Y == "other"))

# Plot histograms for individuals with each eye color
hist(blue_eyes$log_p_brown, main = "Histogram of log ratio for blue eyes", xlab = "log ratio")
```




```r
hist(brown_eyes$log_p_brown, main = "Histogram of log ratio for brown eyes", xlab = "log ratio")
```




```
hist(other_eyes$log_p_brown, main = "Histogram of log ratio for other eyes", xlab = "log ratio")
```



3 RNA-Seq Analysis

1. Read this downloaded file into R using read.table. Use the edgeR package to test whether these genes are differentially expressed between the ethanol (E) and glucose (G) conditions.

```
rna_seq <- read.table("./hw2data/q3/rnaseq.txt", header = TRUE)

# From Section 1.4 of edgeR Quick Start Guide
group <- factor(c(1, 1, 2, 2))
y <- DGEList(counts = rna_seq, group = group)
keep <- filterByExpr(y)
y <- y[keep, , keep.lib.sizes = FALSE]
y <- calcNormFactors(y)
design <- model.matrix(~group)
y <- estimateDisp(y, design)
fit <- glmQLFit(y, design)
qlf <- glmQLFTest(fit, coef = 2)

# Extract table from top tags object and print p-values for each gene
top_tags_table <- topTags(qlf, n = nrow(rna_seq))[["table"]]
knitr::kable(cbind(rownames(top_tags_table), top_tags_table$PValue) %>%
    `colnames<-`(c("gene", "p-value")), caption = "Gene p-values")
```

Table 2: Gene p-values

gene	p-value
YLR070C	2.40301517560778e-05
YLR075W	0.00158354340693725
YLR079W	0.00158896350608023
YLR084C	0.00863499208594918
YLR078C	0.0225281688351836
YLR082C	0.0340625989844454
YLR077W	0.0818134534053452
YLR072W	0.135696722916177
YLR081W	0.189427579138668
YLR073C	0.251838024339884
YLR067C	0.28076697381737
YLR074C	0.311897229685934
YLR069C	0.314581408572567
YLR071C	0.363035114785894
YLR083C	0.377172569120943
YLR080W	0.642014579307543
YLR068W	0.69474442411075

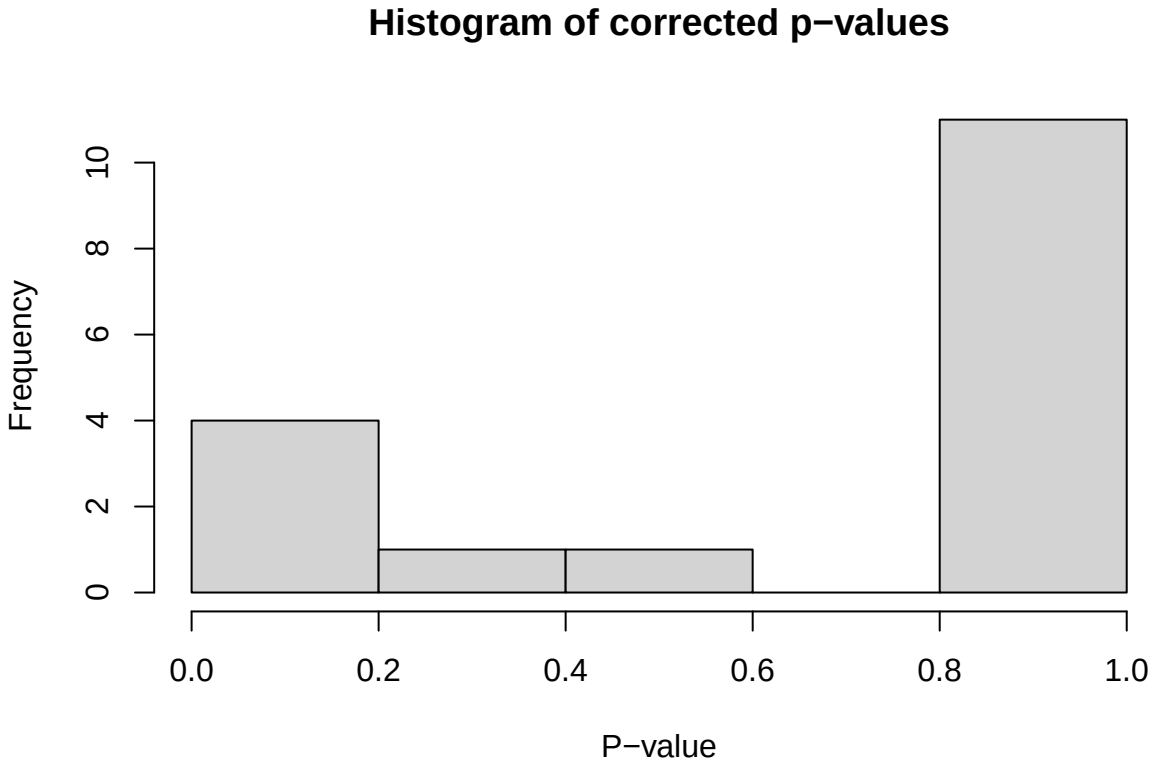
2. edgeR calculates a p-value for each gene, but when you have multiple p-values you need to adjust for multiple hypothesis testing.

   (a) In 1-2 sentences, why is this necessary?

   - **The larger the number of hypotheses, the more likely it is that the null hypothesis will be incorrectly rejected. What the Bonferroni correction does is compensate for this increase by making the significance threshold more strict.**

   (b) Correct those p-values using the Bonferroni correction. Show a histogram of your corrected p-values. After Bonferroni correction, how many are significant at a threshold of .05?

```
top_tags_table_adjusted <- topTags(qlf, n = nrow(rna_seq), adjust.method = "bonferroni")[["table"]]

hist(top_tags_table_adjusted$FWER, main = "Histogram of corrected p-values", xlab = "P-value")
```

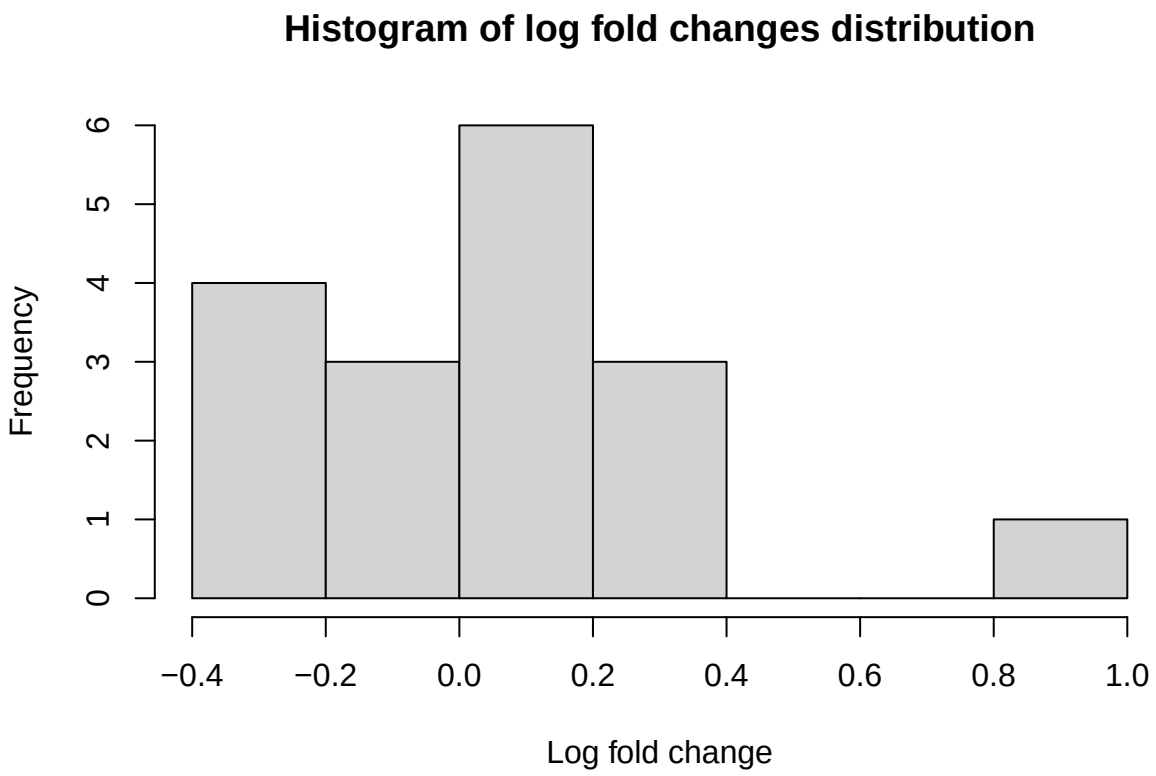



```
num_signifcant <- sum(top_tags_table_adjusted$FWER < 0.05)
```

- **After the Bonferroni correction there are 3 significant p-values at a threshold of** .05**.**

3. Show a histogram of the distribution of log fold changes. What is the most overexpressed gene in E relative to G? What is the most overexpressed gene in G relative to E?

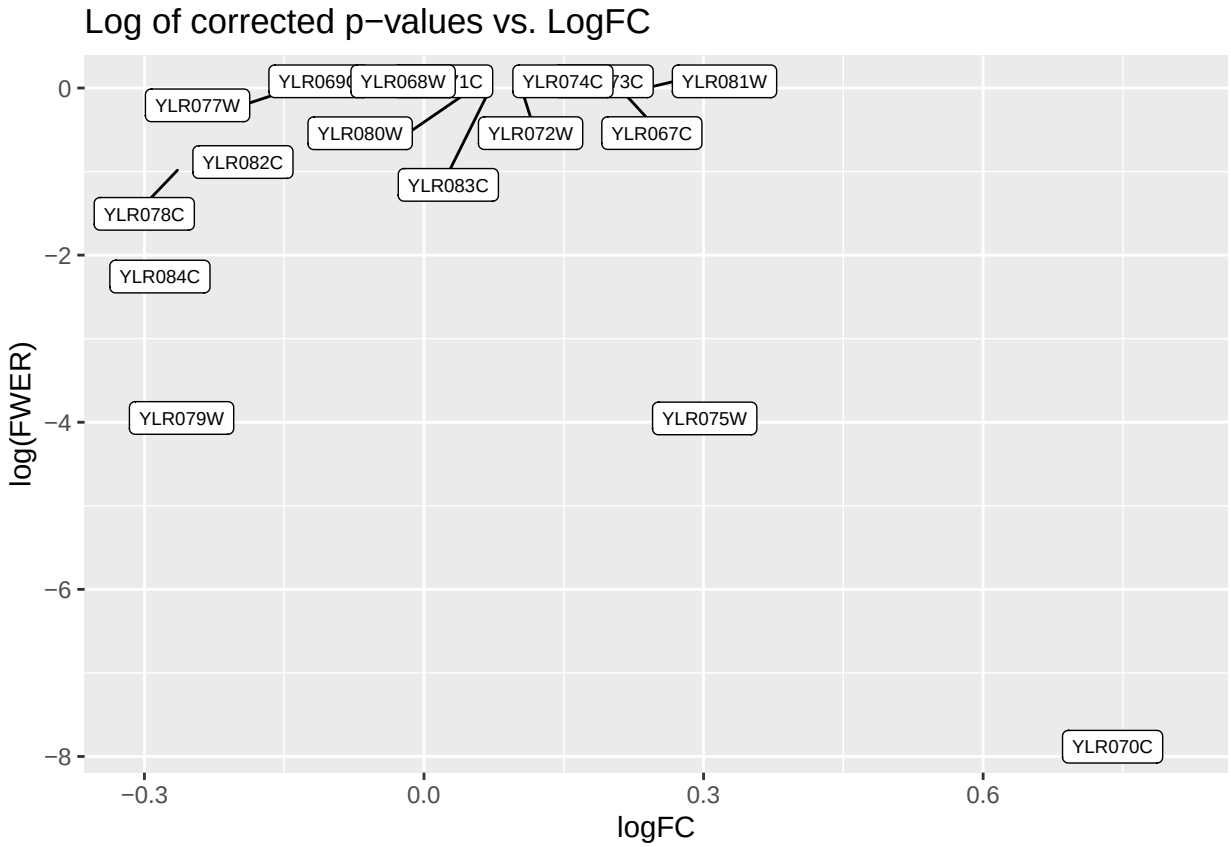
```
hist(top_tags_table_adjusted$logFC, main = "Histogram of log fold changes distribution",
    xlab = "Log fold change")
```



- **The gene with the most negative log fold change is YLR079W with a value of -0.30840408. The gene with the most positive log fold change is YLR070C with a value of 0.80713045. For YLR070C the expression level of E is higher than G and since the log value is positive E is overexpressed relative to G. For YLR079W the expression level of G is higher than E and since the log value is negative G is overexpressed relative to E. The sign of the log fold change determines whether the gene is overexpressed in E relative to G or vice , with the magnitude determining whether the gene is actually the most overexpressed.**

4. Use the R graphing package ggplot2 with option 'geom text' to create a plot with the logFC on the x-axis, the log of the Bonferroni-corrected p-value on the y-axis, and the names (not just dots) of each of the 18 genes on the graph. What does this plot suggest is the relationship between effect size (logFC) and significance?

```
plot_logfc_pval <- ggplot2::ggplot(data = top_tags_table_adjusted, mapping = aes(logFC,
    log(FWER), label = rownames(top_tags_table_adjusted)))
plot_logfc_pval + ggtitle("Log of corrected p-values vs. LogFC") + ggrepel::geom_label_repel(size = 2.5,
    max.overlaps = 12)
```



- **As logFC increases, the log of the corrected p-values also increases until it reaches the cap placed by the Bonferroni correction. This corresponds with an increase in significance.**

4 CRISPR Technologies

1. What is Cas9 and what does it do?

- **Cas9 is an RNA-programmable DNA nuclease. Although bacterial in origin, it has been repurposed to cut DNA at targeted sequences and ultimately mutate the human genome.**

2. How is Cas9 modified to achieve different effects in CRISPRko, CRISPRi, and CRISPRa systems?

- **In CRISPRi, Cas9 is used to turn gene expression down by targeting repressive domains to promoters. This is accomplished by using a modified Cas9 protein that has no nuclease activity and is paired with transcriptional repressors. In CRISPRa, gene expression is turned up by targeting activating domains to promoters. Like CRISPRi, the Cas9 protein used in CRISPRa is catalytically inactive but it is paired with transcriptional activators instead of repressors. CRISPko uses active Cas9 to introduce site-specific double strand breaks that when repaired cause a frameshift mutation leading to the loss of function of the targeted gene.**

3. What is a PAM? Why does it matter?

- **PAM stands for protospacer adjacent motif. The PAM is a short sequence directly following the target DNA sequence but not part of the bacterial genome. This is important because it allows Cas9 to recognize what is not its own bacterial host's DNA and thus actually cleave the target sequence and not attack itself.**

4. Use CRISPick to design 5 sgRNA sequences for the gene BRCA1 using reference genome GRCh38, the CRISPRi system, and SpyoCas9. What is the sgRNA sequence for the third result? What does the

"On-Target Efficacy Score" column describe and why is it important?

```
# Raw data can be accessed online at the CRISPick results page:
# https://portals.broadinstitute.org/gppx/crispick/public/results/e083760b-a746-4819-9586-9ee2968a810d
crispick_data <- read.delim("./hw2data/q4/sgrna-designs-e083760b-a746-4819-9586-9ee2968a810d.txt",
    sep = "\t")
knitr::kable(cbind(rownames(crispick_data), crispick_data$sgRNA.Sequence) %>%
    `colnames<-`(c("ranking", "sequence")), caption = "sgRNA sequences")
```

Table 3: sgRNA sequences

ranking	sequence
1	TGAAGGCCTCCTGAGCGCAG
2	TTACCCAGAGCAGAGGGTGA
3	ACTGGGCCCCTGCGCTCAGG
4	CTGGACGGGGGACAGGCTGT
5	GGTGAAGGCCTCCTGAGCGC

- **The sgRNA sequence for the third result is ACTGGGCCCCTGCGCTCAGG. The "On-Target Efficacy Score" column measures how well the designed sequence maximizes on-target (at the intended target sequence) activity. This is important because a low score could lead to Cas9 not reaching the actual target and perhaps even affecting off-target locations.**

```
sessionInfo(package = NULL)
```

```
## R version 4.1.1 (2021-08-10)
## Platform: x86_64-apple-darwin17.0 (64-bit)
## Running under: macOS Big Sur 10.16
##
## Matrix products: default
## BLAS:   /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRblas.0.dylib
## LAPACK: /Library/Frameworks/R.framework/Versions/4.1/Resources/lib/libRlapack.dylib
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods   base
##
## other attached packages:
## [1] formatR_1.11        knitr_1.36          ggrepel_0.9.1
## [4] ggplot2_3.3.5       edgeR_3.34.1        limma_3.48.3
## [7] BiocManager_1.30.16 dplyr_1.0.7
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_1.0.7       highr_0.9        pillar_1.6.3     compiler_4.1.1
##  [5] tools_4.1.1      digest_0.6.28    evaluate_0.14    lifecycle_1.0.1
##  [9] tibble_3.1.5     gtable_0.3.0     lattice_0.20-45  pkgconfig_2.0.3
## [13] rlang_0.4.11     yaml_2.2.1       xfun_0.26        fastmap_1.1.0
## [17] withr_2.4.2      stringr_1.4.0    generics_0.1.0   vctrs_0.3.8
## [21] locfit_1.5-9.4   grid_4.1.1       tidyselect_1.1.1 glue_1.4.2
## [25] R6_2.5.1         fansi_0.5.0      rmarkdown_2.11   farver_2.1.0
## [29] purrr_0.3.4      magrittr_2.0.1   splines_4.1.1    scales_1.1.1
## [33] ellipsis_0.3.2   htmltools_0.5.2  colorspace_2.0-2 labeling_0.4.2
```

```
## [37] utf8_1.2.2       stringi_1.7.4    munsell_0.5.0    crayon_1.4.1
```