Exam homework for BOHTA 2023

Name: Abdullah Faqih Al Mubarok

KU id: vpx267

Question 1- What transcription factors bind muscle- and liver-specific enhancers?

All data for this question is in the data_q1 folder, except enhancer regions which are defined by the slidebase tool - see below

Enhancers are regulatory regions far away from genes, but regulate genes through 3d looping to the gene promoter. Such enhancers are often tissue-specific, and are bound by specific transcription factors that are used in the specific tissue. Earlier work has established that liver-specific regulation is mediated by HNF1, HNF3, HNF4, HNF6, and CCAAT/enhancer binding protein (C/EBP) transcription factors while skeletal muscle specific regulation is mediated by MyoD and Myf- factors, MEF1, SREBP and TEF1.

In this question, we will explore whether we can confirm these earlier observations using two genomics data sets: the FANTOM5 enhancer database (predicted enhancer regions across nearly all tissues and cells -see https://www.nature.com/articles/nature12787) and ENCODE ChIP-seq peaks from the UCSC browser.

To find liver- and muscle-specific enhancers, we will use the SlideBase resource (https://slidebase.binf.ku.dk/) : this is a selection tool where one can select enhancer regions based on their activity in a given tissue. First, view the videos on the SlideBase web site to learn how it works. In the tutorial, enhancer selection is used as an example.

1: Use SlideBase to define two bed files corresponding to liver- and skeletal muscle- specific enhancers. Use the 'Organ expression', not the 'Cell expression' sliders, and obtain two bed files:

1: Enhancer regions where >=80% of expression comes from liver

2: Enhancer regions where >=80% of expression comes from skeletal muscle

How many enhancers are selected in each? How much do they overlap?

Here I downloaded the BED files for the liver and skeletal enhancers from the sliders, saved it on the current working directory and show the count enhancers via UNIX command:

```
wc -l liver_enhancers.bed
wc -l skeletal_enhancers.bed
```

```
##       87 liver_enhancers.bed
##      125 skeletal_enhancers.bed
```

From the above results, I can see that there are 87 enhancers from the liver and 125 enhancers from the skeletal muscle. Next, to know how much those enhancers overlap, we use the bedtools intersect command. Here we use the default overlap requirement which is 1bp (basepair)

```
bedtools intersect -a liver_enhancers.bed -b skeletal_enhancers.bed
```

The command doesn't return any overlap. Hence, I can conclude the two enhancers are not overlapped each other.

2: The chip_tfbs.bed file in the data_q1 folder is taken from the UCSC browser: it is the ChIP-seq peaks for a large number of cells from the ENCODE project. Each row is one ChIP

peak where the name corresponds to the ChIPed transcription factor. Find a way to overlap the enhancers we defined above with these ChIP results.

Overlap each enhancer set with the ChIP-seq regions. What are the 10 transcription factors whose ChIP-sites overlap the most with liver specific enhancers? What are the top 10 for muscle specific enhancers? Show the results in two tables using R. Do the transcription factor names match the previously established liver- and muscle-specific transcription factors that were discussed in the introduction?

First of all, I will find the overlap of each enhancers via the bedtools intersect command, again with the default overlap requirement, and output them into files.

```
# using cut command only to select the first four coloumns of chip_tfbs.bed files
# which are the chromosome, start, end, and the name of the TFs.

cut -f 1-4 data_q1/chip_tfbs.bed | \
  bedtools intersect -a stdin -b liver_enhancers.bed -c | \
  sort -k5,5rn > tfbs_liver.bed

cut -f 1-4 data_q1/chip_tfbs.bed | \
  bedtools intersect -a stdin -b skeletal_enhancers.bed -c | \
  sort -k5,5rn > tfbs_skeletal.bed
```

After that, I read the files to answer the top 10 overlapped CHIP-sites for the liver and the muscle enhancers.

```
tfbs_liver <- read_tsv("tfbs_liver.bed",
                       col_names = FALSE, show_col_types = FALSE)
colnames(tfbs_liver) <- c("chrom", "chromStart", "chromEnd",
                          "chipName", "countOverlap")

# Finding the top 10 from the liver
tfbs_liver <- tfbs_liver |>
  group_by(chipName) |>
  summarise(sumOverlap = sum(countOverlap)) |>
  arrange(desc(sumOverlap))

top_10_tfbs_liver <- tfbs_liver |>
  head(10)

print(top_10_tfbs_liver)
```

```
## # A tibble: 10 x 2
##    chipName        sumOverlap
##    <chr>                <dbl>
##  1 Pol2                    33
##  2 CEBPB                   28
##  3 HDAC2_(SC-6296)         27
##  4 HEY1                    27
##  5 p300                    27
##  6 FOXA2_(SC-6554)         26
##  7 SP1                     26
##  8 FOXA1_(C-20)            25
##  9 TBP                     24
## 10 HNF4A_(H-171)           23
```

From the liver results, I could see the sub-family of HNF3 proteins (which is also named FOX-protein[1]), HNF4, CCAAT or C/EBPB. However, I could not find any present of HNF1 and HNF6 transcription factors.

```
tfbs_liver |> filter(str_detect(chipName, "(?i)(HNF)"))
```

```
## # A tibble: 3 x 2
##   chipName        sumOverlap
##   <chr>                <dbl>
## 1 HNF4A_(H-171)           23
## 2 HNF4G_(SC-6558)         20
## 3 HNF4A                   14
```

However, when I checked the intersect results, there is only HNF4 chipName from the chip__tfbs.bed. Hence, it is expected if I could not find the HNF1 and HNF6 transcription factors.

Next, I checked for the skeletal muscle:

```
tfbs_skeletal <- read_tsv("tfbs_skeletal.bed",
                          col_names = FALSE, show_col_types = FALSE)
colnames(tfbs_skeletal) <- c("chrom", "chromStart", "chromEnd",
                          "chipName", "countOverlap")

# Finding the top 10 from the skeletal muscle
tfbs_skeletal <- tfbs_skeletal |>
  group_by(chipName) |>
  summarise(sumOverlap = sum(countOverlap)) |>
  arrange(desc(sumOverlap))

top_10_tfbs_skeletal <- tfbs_skeletal |>
  head(10)
print(top_10_tfbs_skeletal)
```

```
## # A tibble: 10 x 2
##    chipName    sumOverlap
##    <chr>            <dbl>
##  1 c-Fos               29
##  2 c-Jun               29
##  3 STAT3               28
##  4 p300                19
##  5 GATA-2              17
##  6 JunD                15
##  7 NFKB                15
##  8 Pol2                15
##  9 Rad21               14
## 10 p300_(N-15)         14
```

For the skeletal results, there is not any transcription factors which were mentioned from the earlier studies. Instead I could only see STAT3 as a transcription factor related to skeletal-muscle[2]. I also checked that only SREBP which presents on the ENCODE TFs data. However, it overlapped poorly on the provided enhancers.

```
tfbs_liver |> filter(str_detect(chipName, "(?i)(SREBP|TEF|MEF1|MyoD|Myf)"))
```

```
## # A tibble: 2 x 2
##   chipName sumOverlap
##   <chr>         <dbl>
## 1 SREBP1            2
## 2 SREBP2            1
```

3: To make a proper comparison of counts of transcription factors in each enhancer set, make an xy plot where y is the number of overlaps in liver-specific enhancers, x is he number of

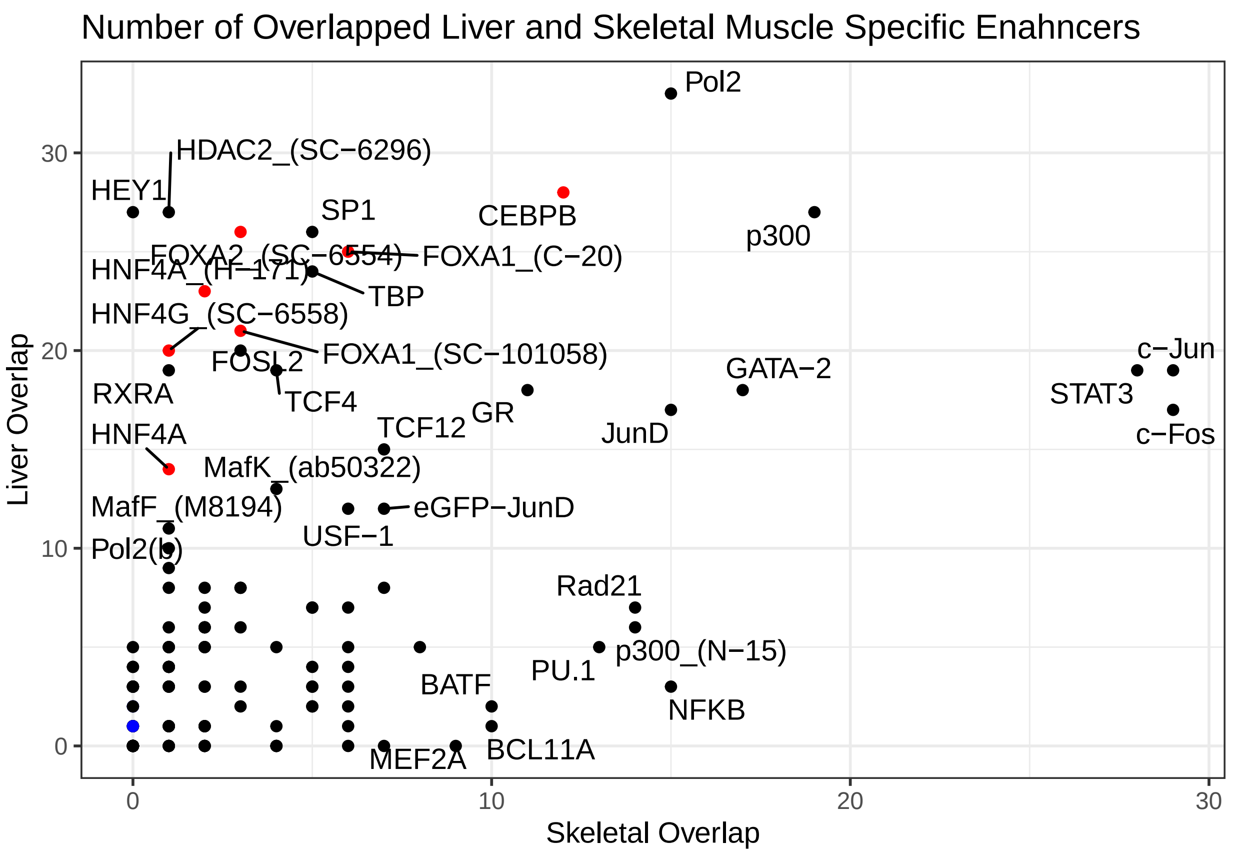
overlaps in muscle-specific enhancers, and each dot is one transcription factor. Make sure that you show if points are overlapping each other in a good way. The dots should also show the name of the factor, at least for the most interesting dots - geom_text_repel() is recommended from the ggrepel package.

Tip: pivot_wider() may be useful.

Comment on your plot: what is shared and what are those transcription factors: what factors are mostly binding muscle- or liver-specific enhancers only?

```r
tfbs_liver_skeletal <- bind_rows(tfbs_liver
                                  |> mutate(type = "tfbs_liver"),
                                 tfbs_skeletal
                                  |> mutate(type = "tfbs_skeletal"))




tfbs_liver_skeletal <- tfbs_liver_skeletal |>
  pivot_wider(names_from = type,
              values_from = sumOverlap)

tfbs_liver_skeletal <- tfbs_liver_skeletal |>
  mutate(colour = ifelse(
    str_detect(chipName, "(?i)(HNF|FOX|CEBP|CCAT)"), "red",
               ifelse(str_detect(chipName, "(?i)(SREBP|TEF|MEF1|MyoD|Myf)"),
                      "blue", "black")
    ))

ggplot(data = tfbs_liver_skeletal,
       mapping = aes(x = tfbs_skeletal, y = tfbs_liver)) +
  geom_point(color = tfbs_liver_skeletal$colour) +
  geom_text_repel(mapping = aes(label = chipName)) +
  labs(title =
         "Number of Overlapped Liver and Skeletal Muscle Specific Enahncers",
       x = "Skeletal Overlap",
       y = "Liver Overlap") +
  theme_bw()
```

```
## Warning: ggrepel: 113 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



On the above plot, HNF3 sub-family (FOXA TFs) and HNF4 subfamily (HNF4A and HNF4G) transcription factors are mainly binding into liver enhancers which is expected and it confirms the early studies. In addition, there are also other TFs which are mostly binding into liver enhancers: HEY1 and HDAC2. Meanwhile, STAT3, c-Jun and c-Fos are highly binding into skeletal enhancers despite their absence on previous studies. As mentioned earlier, it is expected that STAT3 is mostly attaching to the skeletal enhancers[2] while c-Fos and c-Jun usualy form a dimmer (AP-1) which involves in proliferation, apoptosis, and survival[3]. In addition, Pol2, p300 and GATA-2 are shared between the two.

Q2: Is DNA accessibility associated to cancer sub types?

All data for this question is in the data_q2 folder

In the data_q2 folder, there is a large file called atac_brca.tsv. This shows ATAC-seq peak intensities from samples from 74 breast cancer patients. Specifically:

Each row shows one patient (sample).

Column 1 shows patient ID

Column 2 shows a PAM50 type - a classification for cancer sub types - in this data, there are six PAM50 types

Columns 3. . . N shows all ATAC-seq peaks and their intensity

The ATAC-seq peak intensity is measuring how accessible that part of the DNA is. The reason this experiment was made was that there was a hypothesis that DNA accessibility can tell us something about breast cancer and in particular breast cancer sub type.

Let's find out if this is true.

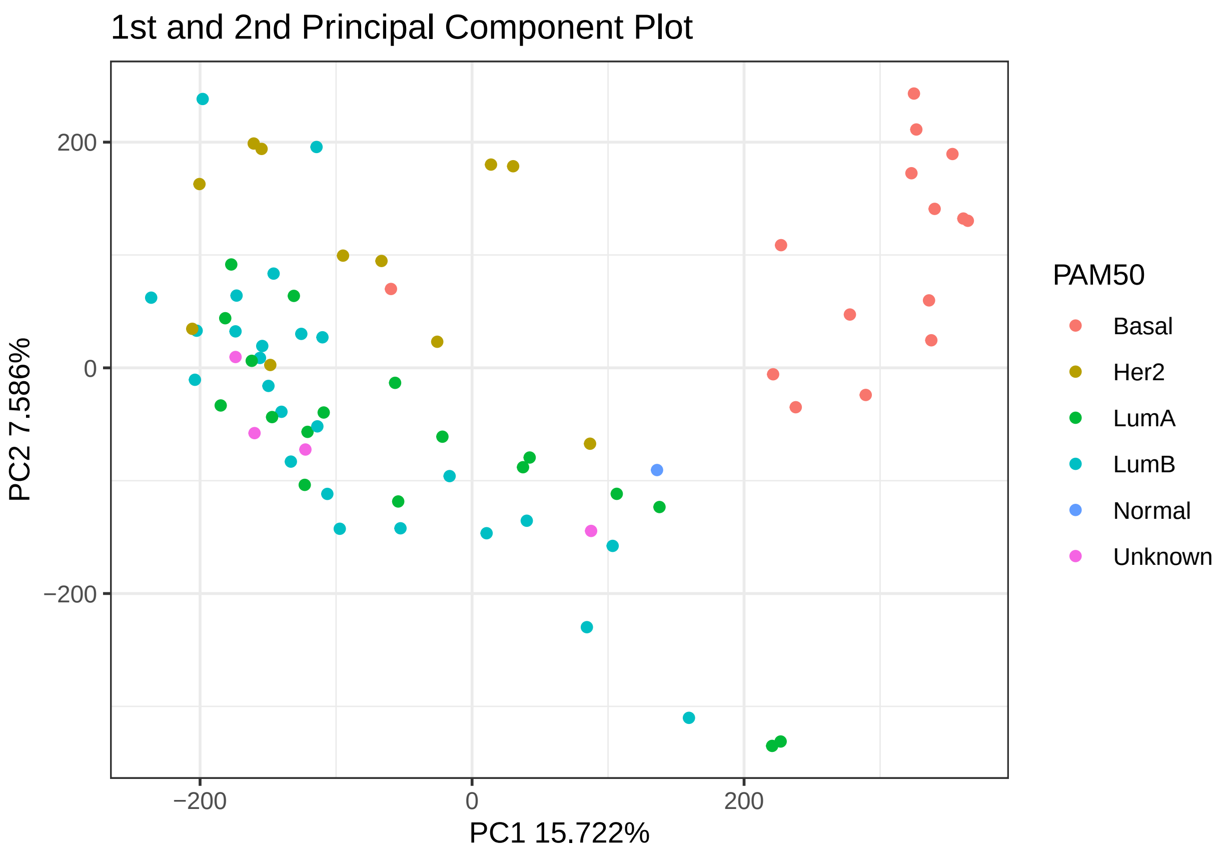
1: Make a PCA of the ATAC-seq data where dots indicate patients. You should scale and

center the data, and show percent explained variance on axes. Do note this is a big file: it make take a bit more time to analyze than in class exercises. Comment on your plot

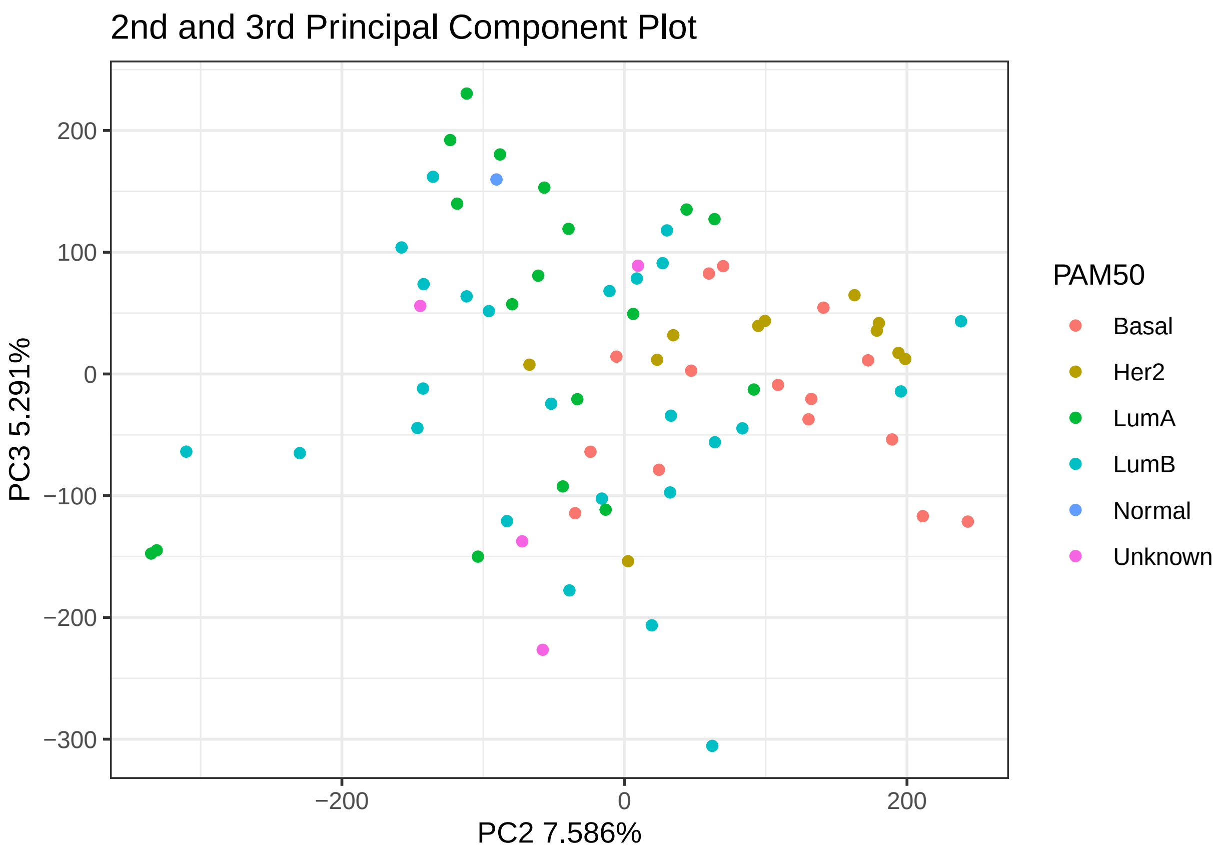
```r
data_q2 <- read_tsv("./data_q2/atac_brca.tsv", show_col_types = FALSE)

# Scaled and centered PCA
data_q2_numeric <- data_q2 |>  select(!c(sample,PAM50))
data_q2_pca <- prcomp(data_q2_numeric,
                      center = TRUE, scale = TRUE)

summary_data_q2_pca <- summary(data_q2_pca)

# Get the variance proportion of each principal components
var_prop <- summary_data_q2_pca$importance["Proportion of Variance",]

# Make a tibble
annot_pca_result <- as.tibble(data_q2_pca$x) |>
  add_column(data_q2 |>
               select(sample, PAM50),
             .before = "PC1"
             )
```

```
## Warning: `as.tibble()` was deprecated in tibble 2.0.0.
## i Please use `as_tibble()` instead.
## i The signature and semantics have changed, see `?as_tibble`.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
# Plotting
ggplot(data = annot_pca_result,
       mapping = aes(x = PC1, y = PC2, colour = PAM50)) +
  geom_point() +
  labs(title = "1st and 2nd Principal Component Plot",
       x = paste0("PC1 ", var_prop[1]*100, "%"),
       y = paste0("PC2 ", var_prop[2]*100, "%")) +
  theme_bw()
```



```
ggplot(data = annot_pca_result,
       mapping = aes(x = PC2, y = PC3, colour = PAM50)) +
  geom_point() +
  labs(title = "2nd and 3rd Principal Component Plot",
       x = paste0("PC2 ", var_prop[2]*100, "%"),
       y = paste0("PC3 ", var_prop[3]*100, "%")) +
  theme_bw()
```



Using the 1st and 2nd PCs, I can see that Basal PAM50 class is the most distinct breast cancer type. Other types tend to be mixed and can not clearly separated by the two PCs. "Unknown" class samples are located between those not clearly separated classes. Furthermore, plotting the 2nd and 3rd PCs is not so informative.

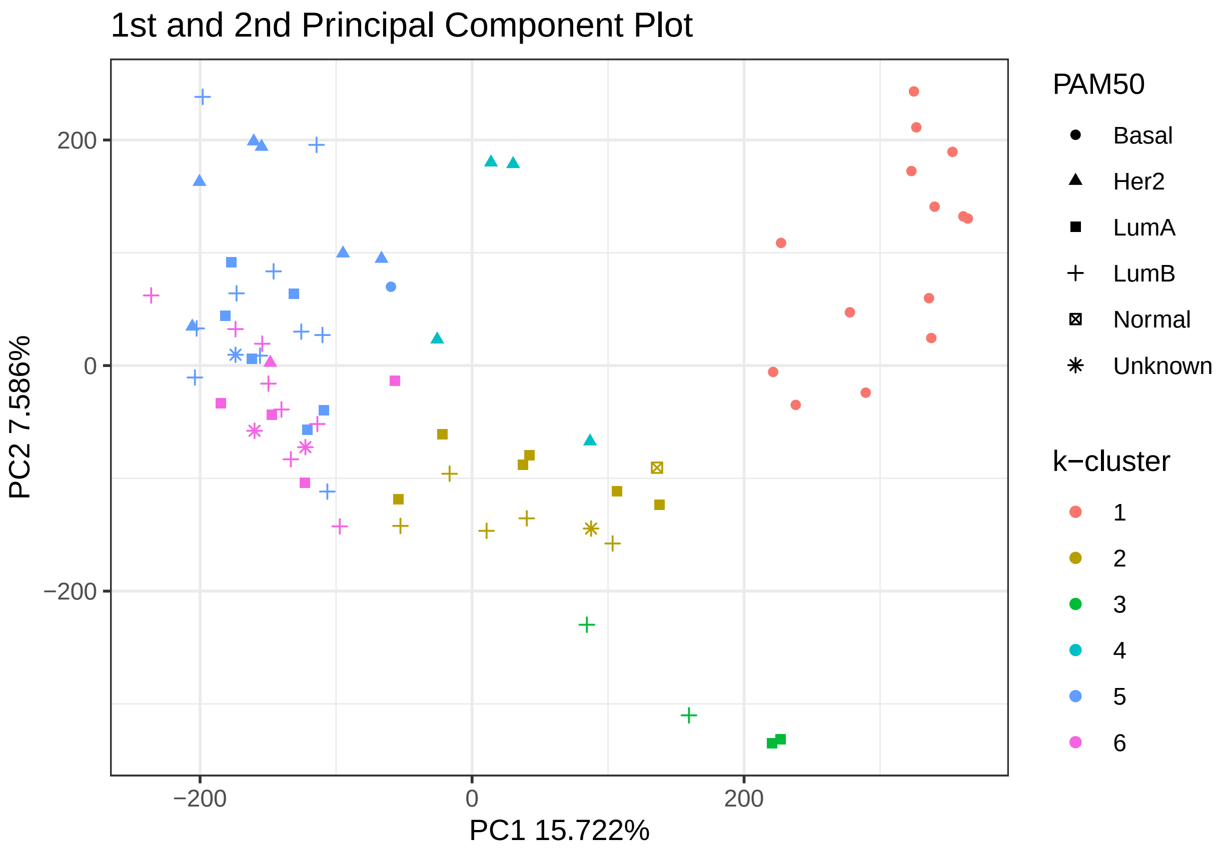
2: The PAM50 classification has six classes. We would hope those classes correspond to clusters in the data, and in particular one the PCA that we made. Use k means where k =6 on the atac-seq matrix. Then:

i) Visualize both PAM50 classification and what k means cluster each patient belong to in the PCA plot (with % variance indicated on axes) and

ii) Make a table that shows the overlaps between k means and PAM50 classifications (e.g. how many patients are in k means group 1 and PAM50 classification LumA, etc etc)

In the ideal scenario, these k means and PAM50 classifications would agree almost 100%. Do they? Discuss and interpret the outcome.

```r
# Set seed for reproducible result
set.seed(15)
data_q2_cluster <- kmeans(as.matrix(data_q2_numeric), 6)


# Plotting
ggplot(data = annot_pca_result,
       mapping = aes(x = PC1, y = PC2)) +
  geom_point(mapping = aes(colour = as.factor(data_q2_cluster$cluster),
                           shape = PAM50)) +
  labs(title = "1st and 2nd Principal Component Plot",
       x = paste0("PC1 ", var_prop[1]*100, "%"),
       y = paste0("PC2 ", var_prop[2]*100, "%"),
       colour = "k-cluster") +
  theme_bw()
```



```
# Creating the tables to compare k-means result
class_table <- table(data_q2_cluster$cluster, data_q2$PAM50)
print(class_table)
```

```
##
##     Basal Her2 LumA LumB Normal Unknown
##   1    14    0    0    0      0       0
##   2     0    0    6    5      1       1
##   3     0    0    2    2      0       0
##   4     0    4    0    0      0       0
##   5     1    6    6   10      0       1
##   6     0    1    4    8      0       2
```

The kmeans classifications could only give "almost agree 100%" clustering for the Basal cancer sub-type as can be seen on the frequency table of the k-clusters and sub-type breast cancer types. This result is also reflected from the PCA plot where the Basal breast cancer sub-type class tends to be well separated from the other classes. However, the other groups tend to not agree. It can be due those groups are mixed and close to each other.

Question 3: Helping Novo Nordisk

All data for this question is in the data_q3 folder.

You have been hired by the Danish pharmaceutical giant Novo Nordisk to analyze an RNA-Seq study they have recently conducted. The study involves treatment of pancreatic islet cells with

new experimental drugs for treatment of type 2 diabetes. Novo Nordisk wants to investigate how the drugs affects cellular mRNA levels in general, and whether the expression of key groups of genes are affected.

As the patent for the new experimental drug is still pending, Novo Nordisk has censored the names of genes. The experiment uses four treatments, A,B,C,D - we are not really told what these are, except that C and D are two different types of controls, but there is a suspicion these two controls are so similar that should be merged into one group. This is one of your jobs to find out.

You have been supplied with 4 files in the question 3 data folder:

- `studyDesign.tsv`: File describing treatment of the 30 samples included in the study.
- `countMatrix.tsv`: Number of RNA-Seq reads mapping to each of the genes.
- `normalizedMatrix.tsv`: Normalized expression to each of the genes.
- `diabetesGene.tsv`: Collection of IDs of genes known to be involved in type 2 diabetes.

1: Read all data sets into R, and make sure the top three files have matching numbers and names of both samples and genes

```r
groups <- read_tsv("./data_q3/studyDesign.tsv",
                   show_col_types = FALSE)
measurements <- read.delim("./data_q3/countMatrix.tsv", sep = "\t",
                           header = TRUE, row.names = 1) |> as.matrix()
normalized_measurements <- read.delim("./data_q3/normalizedMatrix.tsv",
                                      sep = "\t",
                                      header = TRUE,
                                      row.names = 1) |> as.matrix()
diabetes_genes <- read_tsv("./data_q3/diabetesGenes.tsv",
                           show_col_types = FALSE)

# Checking samples
#Check whether the number of samples between groups and measurement are the same
dim(groups)[1] == dim(measurements)[2]
```

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

```r
#Check whether the name of samples between groups and measurement are the same
!any((groups$Sample == colnames(measurements))==FALSE)
```

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

```r
#Check whether the number of samples between groups and normalized
#measurement are the same
dim(groups)[1] == dim(normalized_measurements)[2]
```

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

```r
#Check whether the name of samples between groups and normalized
#measurement are the same
!any((groups[1] == colnames(normalized_measurements))==FALSE)
```

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

```r
# Checking genes length
#Check whether genes length between measurement and normalized measurement are the same:",
dim(measurements)[1] == dim(normalized_measurements)[1]
```

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

```
#"Check whether the name of genes between normalized and unnormalized measurement are the same:",
!any((rownames(measurements) == rownames(normalized_measurements))==FALSE)
```

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

```
# Print the sample and gene size
cat("Sample size:", dim(measurements)[2], " |  Gene size:", dim(measurements)[1])
```

```
## Sample size: 24  |  Gene size: 6789
```

```
# Print the number of samples for each treatments
groups |> dplyr::count(Condition)
```

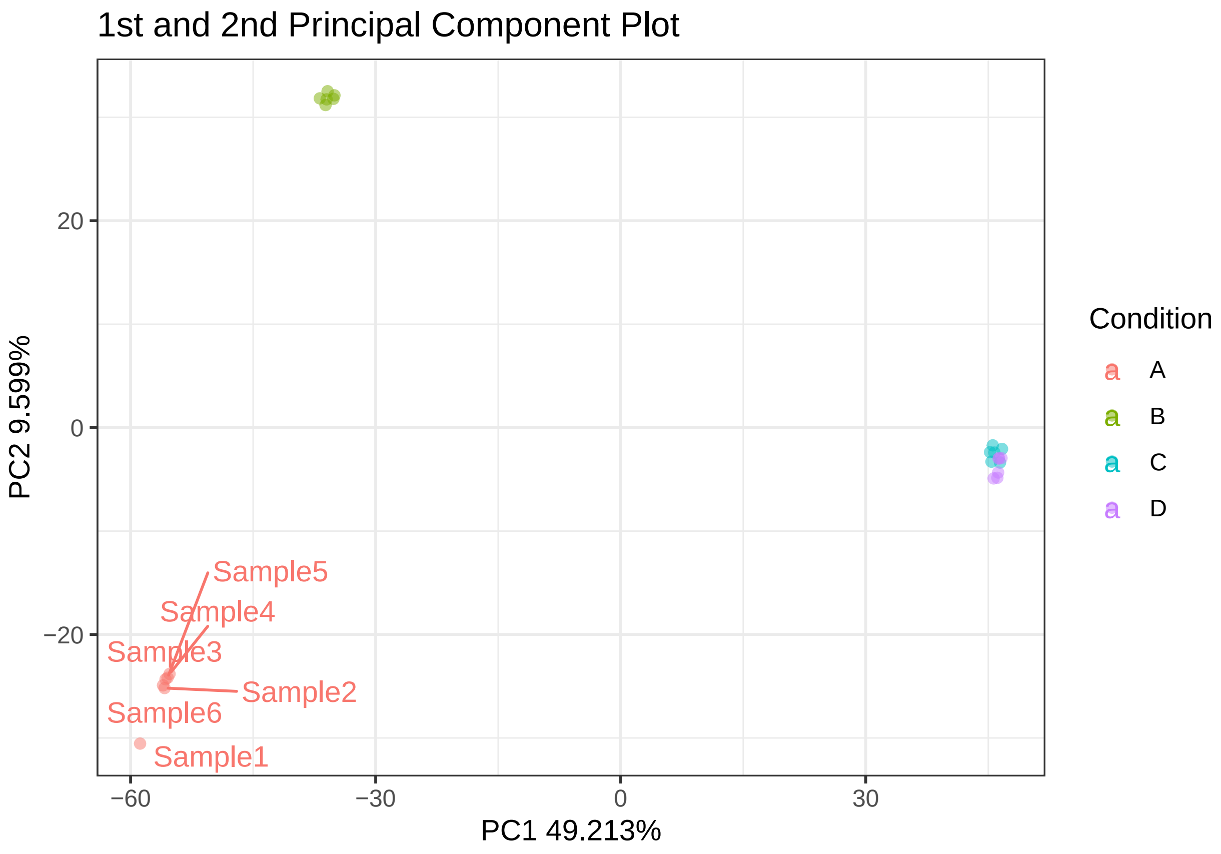
```
## # A tibble: 4 x 2
##   Condition     n
##   <chr>     <int>
## 1 A             6
## 2 B             6
## 3 C             6
## 4 D             6
```

From the result above, The number and the name of the samples and genes are validated (the same between top three data). In addition, there are 24 samples and 6789 genes. Furthermore, the number of the samples for each treatments are the same (6).

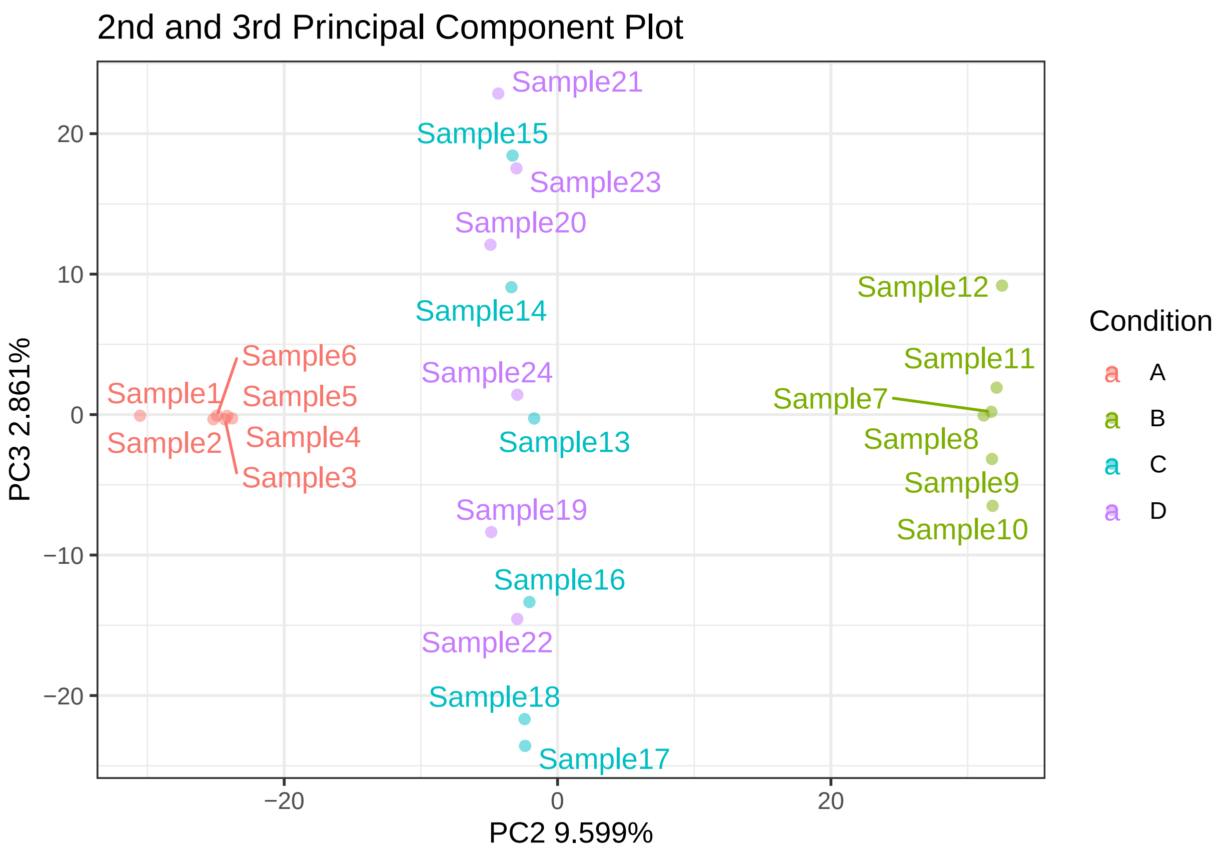
2: Use a PCA to see how well separated the groups are, if there are any outliers and whether it makes sense to make C and D into a single control group. You should center but not scale the data before PCA. As above, indicate % variance on axes.

```
# Need to transpose since I want the observations (samples) as rows
annot_measurements <- as_tibble(t(measurements),
                                rownames = "Sample") |>
  mutate(Condition = groups$Condition, .after = Sample)

annot_normalized_measurements <- as_tibble(t(normalized_measurements),
                                rownames = "Sample") |>
  mutate(Condition = groups$Condition, .after = Sample)

measurements_pca <- prcomp(annot_normalized_measurements |> select(!c(Sample, Condition)),
                           center = TRUE)

summary_measurements_pca<- summary(measurements_pca)

# Get the variance proportion of each principal components
var_prop_measurements <- summary_measurements_pca$importance["Proportion of Variance",]

# Make a tibble
annot_measurements_pca_result <- as.tibble(measurements_pca$x) |>
  add_column(annot_measurements |>
               select(Sample, Condition),
             .before = "PC1"
             )

# Plotting
ggplot(data = annot_measurements_pca_result,
       mapping = aes(x = PC1, y = PC2, colour = Condition)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(mapping = aes(label = Sample)) +
  labs(title = "1st and 2nd Principal Component Plot",
       x = paste0("PC1 ", var_prop_measurements[1]*100, "%"),
       y = paste0("PC2 ", var_prop_measurements[2]*100, "%")) +
  theme_bw()
```

```
## Warning: ggrepel: 18 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```
ggplot(data = annot_measurements_pca_result,
       mapping = aes(x = PC2, y = PC3, colour = Condition)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(mapping = aes(label = Sample)) +
  labs(title = "2nd and 3rd Principal Component Plot",
       x = paste0("PC2 ", var_prop_measurements[2]*100, "%"),
       y = paste0("PC3 ", var_prop_measurements[3]*100, "%")) +
  theme_bw()
```



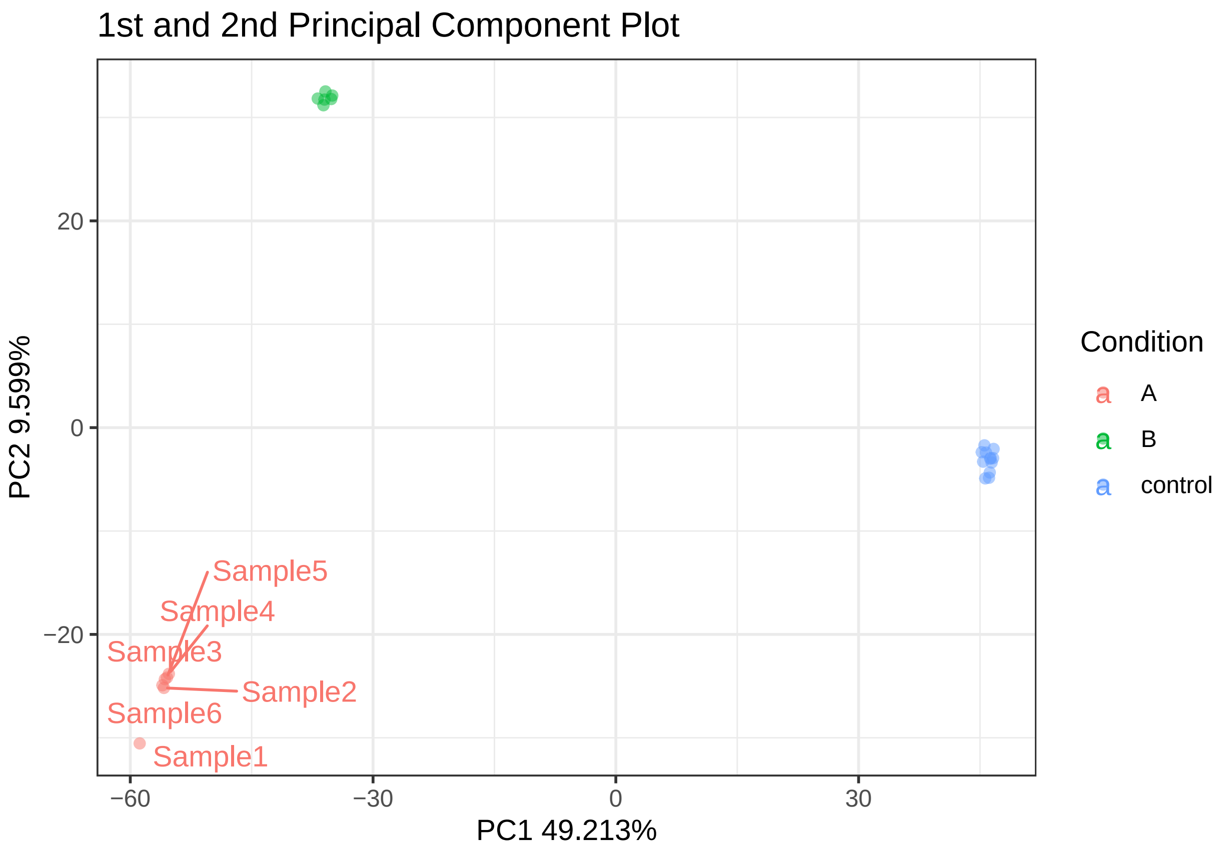
To see whether C and D can be merged, I did principal analysis for the normalized measurement data and plotted two PCs plots (PC1 vs PC2 and PC2 vs PC3). It can be seen that the C and D tend to be close each other and well-separated from A and B. Therefore, It might be good idea to merge the two groups. In addition, It seems there is not any outliers based on the plots.

3: Discuss the PCA and make the appropriate changes - remove potential outliers if any, decide if C+D samples should be merged into one control group and if so, do that without editing the files. If you decide not to merge, select one of C or D to be the control group that we will use below. Regardless, we will compare A and B to this control group below and we will call it 'control'. Make a new PCA plot after making the changes and comment on that. As above, indicate % variance on axes.

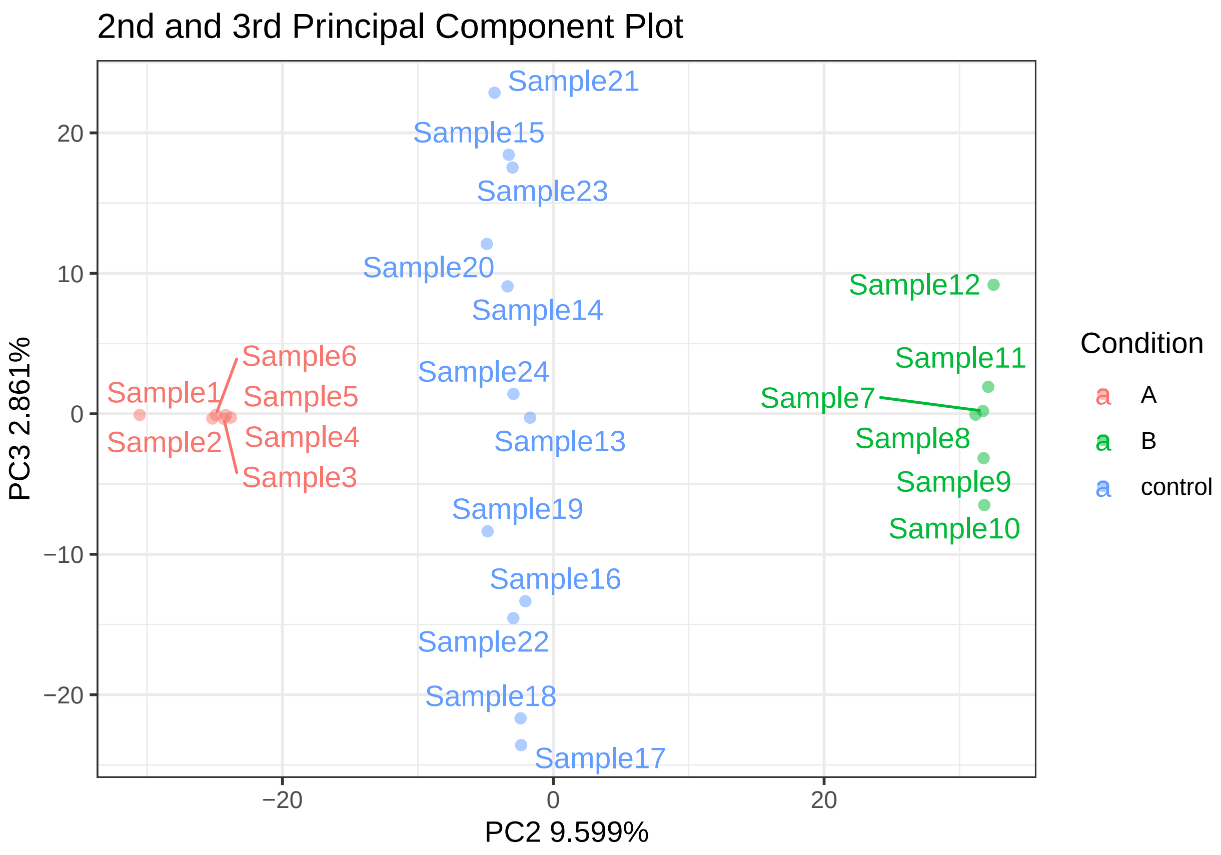
```
merge_annot_measurements <- annot_measurements |>
  mutate(Condition = ifelse(Condition == "D" | Condition == "C",
                            "control", Condition))

merge_annot_normalized_measurements <- annot_normalized_measurements |>
  mutate(Condition = ifelse(Condition == "D" | Condition == "C",
                            "control", Condition))

merge_groups <- groups |>
  mutate(Condition = ifelse(Condition == "D" | Condition == "C",
                            "control", Condition))

measurements_pca <- prcomp(annot_normalized_measurements |> select(!c(Sample, Condition)),
                           center = TRUE)

summary_measurements_pca<- summary(measurements_pca)
```

```r
# Get the variance proportion of each principal components
var_prop_measurements <- summary_measurements_pca$importance["Proportion of Variance",]

# Make a tibble
annot_measurements_pca_result <- as.tibble(measurements_pca$x) |>
  add_column(merge_annot_measurements |>
               select(Sample, Condition),
             .before = "PC1"
             )

# Plotting
ggplot(data = annot_measurements_pca_result,
       mapping = aes(x = PC1, y = PC2, colour = Condition)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(mapping = aes(label = Sample)) +
  labs(title = "1st and 2nd Principal Component Plot",
       x = paste0("PC1 ", var_prop_measurements[1]*100, "%"),
       y = paste0("PC2 ", var_prop_measurements[2]*100, "%")) +
  theme_bw()
```

```
## Warning: ggrepel: 18 unlabeled data points (too many overlaps). Consider
## increasing max.overlaps
```



```r
ggplot(data = annot_measurements_pca_result,
       mapping = aes(x = PC2, y = PC3, colour = Condition)) +
  geom_point(alpha = 0.5) +
  geom_text_repel(mapping = aes(label = Sample)) +
  labs(title = "2nd and 3rd Principal Component Plot",
       x = paste0("PC2 ", var_prop_measurements[2]*100, "%"),
       y = paste0("PC3 ", var_prop_measurements[3]*100, "%")) +
  theme_bw()
```



Again, the above PCA plots are the same with the previous plots but differ only on the Condition categories where C and D are merged into "control".

4: Use DESeq2 to obtain differentially expressed (DE) genes comparing i) A vs control and ii) B vs control. Use default parameters, except use a logFC threshold of 0.25 and an adjusted P-value threshold of 0.05. How many genes are up- and down-regulated in respective comparison?

```
dds <- DESeqDataSetFromMatrix(measurements, colData =  merge_groups,
                              design = ~Condition)
```

```
## Warning in DESeqDataSet(se, design = design, ignoreRank): some variables in
## design formula are characters, converting to factors
```

```
dds <- DESeq(dds)
```

```
## estimating size factors

## estimating dispersions

## gene-wise dispersion estimates

## mean-dispersion relationship

## final dispersion estimates

## fitting model and testing
## -- replacing outliers and refitting for 43 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)

## estimating dispersions

## fitting model and testing
```

```
ALPHA = 0.05
LFCTHRESHOLD = 0.25

# The low reads on the genes will be filtered out by the results() leaving the
# padj to "NA", default lfcThreshold is zero.

A_control <- results(dds, contrast = c("Condition", "A", "control"),
                 lfcThreshold = LFCTHRESHOLD, alpha = ALPHA)
tibble_A_control <- results(dds, contrast =  c("Condition", "A", "control"),
                        lfcThreshold = LFCTHRESHOLD, alpha = ALPHA,
                        tidy = TRUE)

B_control <- results(dds, contrast = c("Condition", "B", "control"),
                 lfcThreshold = LFCTHRESHOLD, alpha = ALPHA)
tibble_B_control <- results(dds, contrast =  c("Condition", "B", "control"),
                        lfcThreshold = LFCTHRESHOLD, alpha = ALPHA,
                        tidy = TRUE)

summary(A_control)
```

```
##
## out of 6789 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0.25 (up)     : 1521, 22%
## LFC < -0.25 (down) : 2201, 32%
## outliers [1]        : 20, 0.29%
## low counts [2]      : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```
summary(B_control)
```

```
##
## out of 6789 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0.25 (up)     : 1516, 22%
## LFC < -0.25 (down) : 1442, 21%
## outliers [1]        : 20, 0.29%
## low counts [2]      : 0, 0%
## (mean count < 0)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

From the summary results, Both the A and B treatments up-regulated around 22% of the genes while A treatment has higher percentage of affecting the genes to be down-regulated (32%) compared to B treatment (21%).

5: To check if our data is fine, make the following 'diagnosis' plots and comment on them for

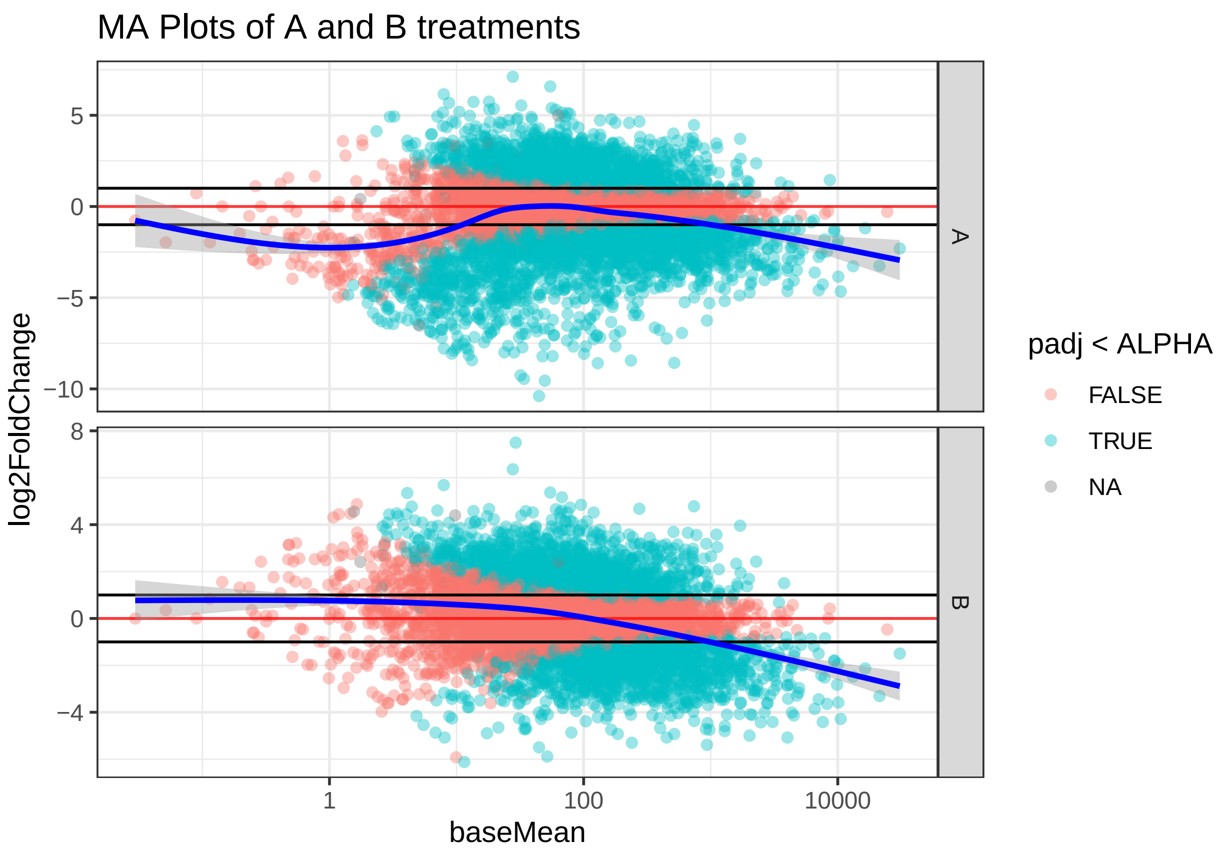
each comparison

For each of these questions, make a single plot with one facet for each comparison

5.1: MA plots. Discuss whether the MA-plots indicate appropriate normalization (max 70 words discussion)

```
A_B_control <- bind_rows(tibble_A_control |> mutate(type = "A"),
                         tibble_B_control |> mutate(type = "B"))
A_B_control <-A_B_control |>
  dplyr::rename(Gene = row)


ggplot(A_B_control, mapping = aes(x = baseMean, y = log2FoldChange,
                             color=padj < ALPHA)) +
  geom_point(alpha=0.4) +
  geom_hline(yintercept = 0, alpha = 0.75, color = "red") +
  geom_hline(yintercept = -1) +
  geom_hline(yintercept = 1) +
  geom_smooth(colour = "blue") +
  scale_x_log10() +
  facet_grid(rows = vars(type), scales = "free") +
  labs(title = "MA Plots of A and B treatments") +
  theme_bw()
```

```
## `geom_smooth()` using method = 'gam' and formula = 'y ~ s(x, bs = "cs")'
```

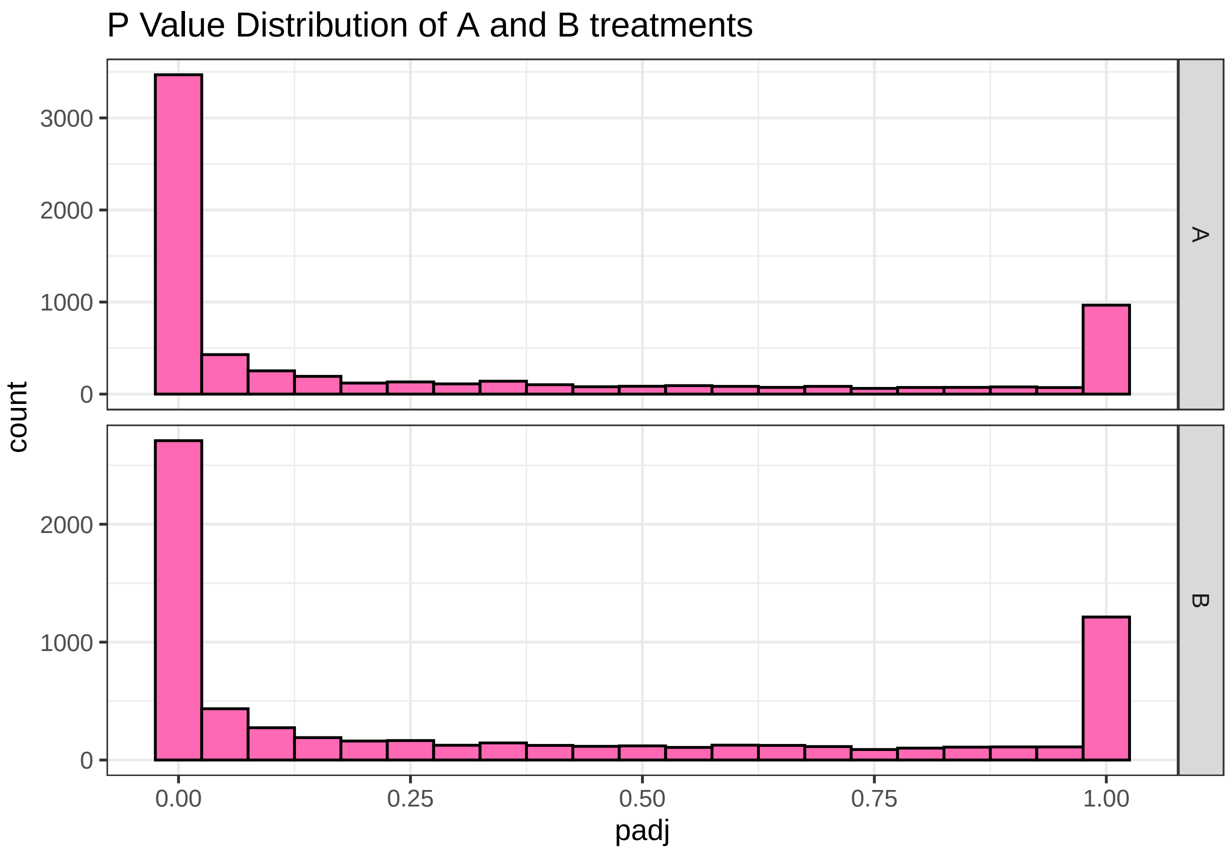


Data is normalized if the Y-axis of MA plots are centered around zero. However,the observed trend line (blue) for both MA plots tend to deviate from the reference horizontal (red). This trends may be the consequence of the majority of the genes are being down-regulated rather than up-regulated in response to the both treatments.

5.2: P value distributions.

```
ggplot(A_B_control, aes(x=padj)) +
  geom_histogram(binwidth = 0.05,
    fill="hotpink", color="black") +
  facet_grid(rows = vars(type), scales = "free") +
  labs(title = "P Value Distribution of A and B treatments") +
  theme_bw()
```

```
## Warning: Removed 40 rows containing non-finite values (`stat_bin()`).
```

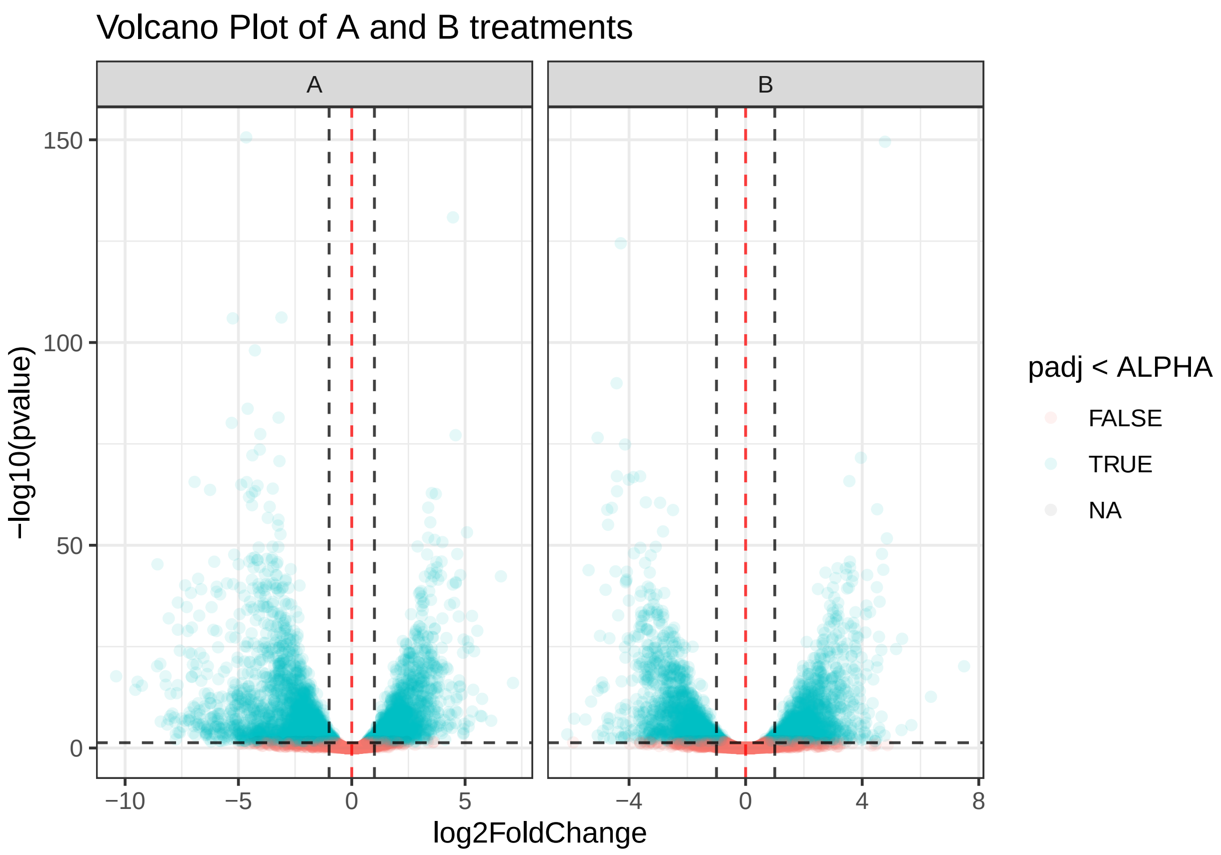


Since the lfcthreshold is set to not zero, It is expected that there is a shift on p-values near 1 since setting the lfcThreshold will impact the statistical test.

6.3 Volcano plots:

```
ggplot(A_B_control, aes(x=log2FoldChange,
                   y=-log10(pvalue), color=padj < ALPHA)) +
  geom_point(alpha=0.1) +
  geom_hline(yintercept = -log10(ALPHA), alpha = 0.75, linetype="dashed") +
  geom_vline(xintercept = -1, alpha = 0.75, linetype="dashed") +
  geom_vline(xintercept = 1, alpha = 0.75, linetype="dashed") +
  geom_vline(xintercept = 0, alpha = 0.75, linetype="dashed", color = "red") +
  facet_grid(cols = vars(type), scales = "free") +
  labs(title = "Volcano Plot of A and B treatments") +
  theme_bw()
```

```
## Warning: Removed 40 rows containing missing values (`geom_point()`).
```



Here I plotted volcano plots with additional dashed line for the p-value threshold (alpha) and two lines which indicate doubled up or down-regulation of LFC

7: Is any of treatment preferentially acting on diabetes-related genes?

-Novo Nordisk claims that treatments A and B affects up-regulated genes related to diabetes. Your task is to investigate whether this is true. They have supplied you with a long list of genes that are diabetes-related - diabetesGenes.tsv

Principally there are two ways of doing this, by answering two questions:

7.1: Are significantly up-regulated genes (FDR <0.05 and log2FC >0) in a given comparison (defined as above) significantly more likely to be diabetes related compared non-differentially expressed genes?

7.2: For each comparison, do diabetes-related genes have a significantly different change in expression vs control than other genes?

Answer both of these questions using the appropriate visualizations and statistical tests. If you use 2x2 contingency tables, show the tables. Comment your results and compare the answers of the two questions - if they are the same, why is that? If not, why is that?

To answer the first question, I chose the Fisher Exact test. The reason for choosing that test are follows:

- I want to know whether there is significant association between two categories: the regulation effect (up-regulated and non-differentially expressed) and the gene type (diabetes genes or not).
- The data might be unbalanced (due to diabetes genes selection)

For this test I have the following hypotheses:

- *H0*: Odds ration is equal to 1. There is no association between two variables (row and column).

- *Alternative*: Odds ratio is not equal to 1. There is association between the two variables (row and column).

At first, I added two columns to indicate whether certain gene is related into diabetes and the effect of the regulation.

```
# Adding diabetes genes and signficantly up-regulated flags
A_B_control <- A_B_control |>
  mutate(gene_flag = ifelse(Gene %in% diabetes_genes$Gene,
                                  "diabetes_gene", "non_diabetes_gene"),
         regulation_effect = ifelse(log2FoldChange>0 & padj<0.05,
                                 "up_regulated",
                                 ifelse(log2FoldChange<0 & padj<0.05,
                                        "down_regulated",
                                        "non_differentially_expressed"))
         )
```

After that I created contingency tables for each comparison

Treatement A:

```
# Creating the contigency table for A
A_contingency <- A_B_control |>
  filter(type == "A" & (regulation_effect %in% c("up_regulated",
                                                 "non_differentially_expressed")
                        )
         ) |>
  select(gene_flag, regulation_effect) |>
  table() |> t()

A_contingency <- A_contingency[2:1,]

# Transpose
print(A_contingency)
```

```
##                               gene_flag
## regulation_effect              diabetes_gene non_diabetes_gene
##   up_regulated                            54              1467
##   non_differentially_expressed            45              3002
```

Treatement B:

```
# Creating the contigency table for B
B_contingency <- A_B_control |>
  filter(type == "B" & (regulation_effect %in% c("up_regulated",
                                                 "non_differentially_expressed")
                        )
         ) |>
  select(gene_flag, regulation_effect) |>
  table() |> t()

B_contingency <- B_contingency[2:1,]

# Transpose
print(B_contingency)
```

```
##                               gene_flag
## regulation_effect              diabetes_gene non_diabetes_gene
##   up_regulated                           80              1436
##   non_differentially_expressed           20              3791
```

Doing the Fisher test for each treatment:

```
fisher.test(A_contingency)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  A_contingency
## p-value = 1.221e-05
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.613629 3.750874
## sample estimates:
## odds ratio
##   2.455111
```

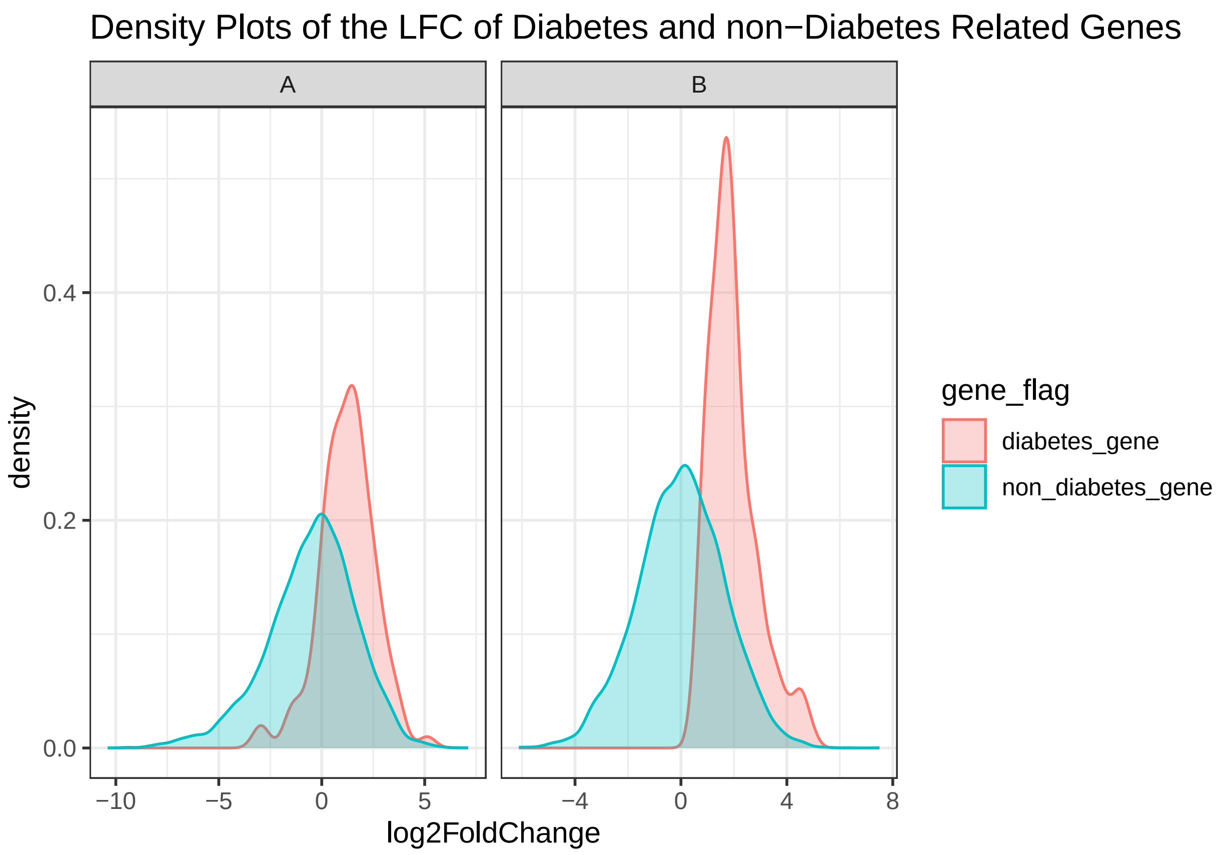
```
fisher.test(B_contingency)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  B_contingency
## p-value < 2.2e-16
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##   6.374133 18.262606
## sample estimates:
## odds ratio
##   10.55437
```

From the test results, A and B treatments have p-value less than my threshold (0.05) which are 1.221e-05 and < 2.2e-16. Therefore I can reject the null hypothesis. **In conclusion, there is a association between the type of genes (diabetes related or not) and the regulation effects (up-regulated and non-differentially expressed) for both A and B treatments. In addition, for A treatment, there is chance of up-regulated for the diabetes genes about two times more while the B treatment is ten times more.**

To answer the second question, I need to plot the distribution of the LFC for diabetes genes and non-diabetes genes of each treatment to choose the test.

```
ggplot(data = A_B_control,
       mapping = aes(x = log2FoldChange, colour = gene_flag, fill = gene_flag,
                     )) +
  geom_density(alpha = 0.3) +
  facet_grid(cols = vars(type), scales = "free") +
  labs(title = "Density Plots of the LFC of Diabetes and non-Diabetes Related Genes") +
  theme_bw()
```



From the density plots above, I can see that the distributions are not normally distributed for the diabetes genes. Thus, I choose the non-parametric test. Since I only have two groups for each treatment (diabetes and non-diabetes genes) I choose the wilcoxon test.

The test has the following hypotheses:

- *H0*: The dispersion or the location shift of the two expression distributions (diabetes and non-diabetes genes) is 0.
- *Alternative*: The dispersion or location shift is not equal to 0.

Next, I did the test for each treatment

Treatment A:

```
wilcox.test(A_B_control |>
              filter(type == "A", gene_flag == "diabetes_gene") |>
              pull(log2FoldChange),
            A_B_control |>
              filter(type == "A", gene_flag == "non_diabetes_gene") |>
              pull(log2FoldChange))
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pull(filter(A_B_control, type == "A", gene_flag == "diabetes_gene"), log2FoldChange) and pull
## W = 504551, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Treatment B:

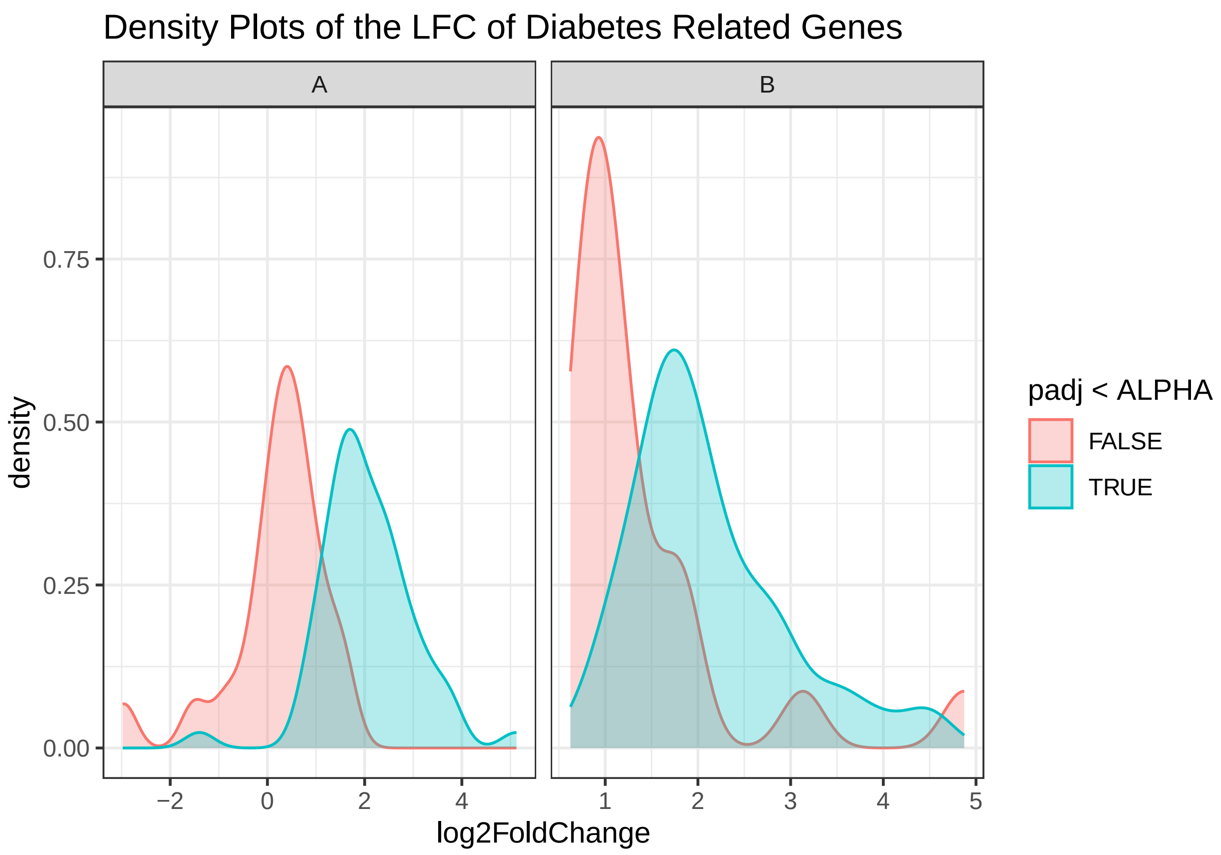
```r
wilcox.test(A_B_control |>
              filter(type == "B", gene_flag == "diabetes_gene") |>
              pull(log2FoldChange),
            A_B_control |>
              filter(type == "B", gene_flag == "non_diabetes_gene") |>
              pull(log2FoldChange))
```

```
##
##  Wilcoxon rank sum test with continuity correction
##
## data:  pull(filter(A_B_control, type == "B", gene_flag == "diabetes_gene"), log2FoldChange) and pull
## W = 571635, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Based on the test results, A and B treatments have p-value less than my threshold (0.05) which are < 2.2e-16. Thus, I can reject the null hypothesis. **In conclusion, there is a significant different change in expression vs control for diabetes genes compared to non-diabetes genes in A and B treatments.**

The two tests give statistically significant results for the two treatments. Hence, I can confidently conclude that the treatments (A and B) can potentially up-regulated diabetes genes. In addition, the reason of why the two tests give the same results might be due to most of the genes have LFC>0.

```r
ggplot(data = A_B_control |> filter(gene_flag == "diabetes_gene"),
       mapping = aes(x = log2FoldChange, colour = padj<ALPHA, fill = padj<ALPHA,
                     )) +
  geom_density(alpha = 0.3) +
  facet_grid(cols = vars(type), scales = "free") +
  labs(title = "Density Plots of the LFC of Diabetes Related Genes") +
  theme_bw()
```



References:

1. Costa RH, Kalinichenko VV, Holterman AX, Wang X. Transcription factors in liver development, differentiation, and regeneration. Hepatology. 2003 Dec;38(6):1331-47. doi: 10.1016/j.hep.2003.09.034. PMID: 14647040.
2. Guadagnin E, Mázala D, Chen YW. STAT3 in Skeletal Muscle Function and Disorders. Int J Mol Sci. 2018 Aug 2;19(8):2265. doi: 10.3390/ijms19082265. PMID: 30072615; PMCID: PMC6121875.
3. Meng Q, Xia Y. c-Jun, at the crossroad of the signaling network. Protein Cell. 2011 Nov;2(11):889-98. doi: 10.1007/s13238-011-1113-3. Epub 2011 Dec 17. PMID: 22180088; PMCID: PMC4875184.