Individual homework BOHTA 2020

Name: Stefano Pellegrini

KU id: MLQ211

Instructions to run the RMD:

- Follow the instructions in exercise 1.1 to download the files from the Genome Browser, use the bedtools intersect to generate `reverse_overlap_forward_count.bed` and `forward_overlap_reverse_count.bed` files, move the files to the same folder where the RMD file is located (or change the path in the RMD).
- Remove the code to print the picture (Figure 1) in exercise 1.3.
- `question2_data` and `question3_data` must be in the same folder where the RMD file is located (or change the path in the RMD).

Question 0

Load all the libraries that you use in the rest of the analyses.

Answer goes here

```
# Load libraries
library(tidyverse)
```

```
## Warning: package 'ggplot2' was built under R version 4.0.2
```

```
## Warning: package 'tidyr' was built under R version 4.0.2
```

```
## Warning: package 'dplyr' was built under R version 4.0.2
```

```
library(DESeq2)
library(pheatmap)
library(viridis)
library(ggrepel)
library(ggforce)
```

```
## Warning: package 'ggforce' was built under R version 4.0.2
```

```
# Set working directory to source file location
setwd(".")
```

Question 1

Introduction

Antisense transcription is a current hot topic in genomics: when a certain genomic region is transcribed on both strands. In one model, the two RNAs produced may hybridize with each other since they are complementray, and then be degraded (a type of negative regulators). In another, deregulation happens becasue of RNA polymerase collisions.

We will now try to answer the question : How common is this in the human genome?

Use the refseq track from the hg19 assembly.

Using R and bedtools (and possibly a text editor to add headers to bed files if needed), find out:

- 1: What fraction of refseq transcripts overlap another transcript on the opposite strand, covering at least 20% of the transcript?

Answer goes here

I started by downloading all the RefSeq transcripts of the hg19 assembly using the Table Browser program of the UCSC Genome Browser. I used the filter function of the Table Browser to download, as two separate files, the RefSeq transcripts on the forward and reverse strands. As output, I chose the BED output format.

```
# Use the bedtools intersect function to find the number of transcript that do not overlap
# another transript (20% threshold)
nice bedtools intersect -a refseq_hg19_forward.bed -b refseq_hg19_reverse.bed -f 0.2 -v | wc -l
nice bedtools intersect -a refseq_hg19_reverse.bed -b refseq_hg19_forward.bed -f 0.2 -v | wc -l
```

Then, I loaded the two files in the Ricco server and I used the bedtools toolset to extract the overlapping transcripts. I first used the bedtools intersect function to find the number of transcripts, on the forward strand, that do not overlap any other transcript on the opposite strand (coverage threshold of 20%). I repeated the command for the reverse strand, and I found that the number of transcripts that do not overlaps any other transcript on the opposite strand is, 36543 for the forward strand, and 35010 for the reverse strand.

```
# Find the total number of RefSeq transcripts on each strands
wc -l refseq_h19_forward.bed
wc -l refseq_h19_reverse.bed
```

```
# Save the results into R
transcripts_forward_NOT_overlap_reverse <- 36543
transcripts_reverse_NOT_overlap_forward <- 35010
tot_transcripts_forward <- 39593
tot_transcripts_reverse <- 38217

# Calculate overlapped transcripts for each strand
transcripts_forward_overlap_reverse <-
  tot_transcripts_forward - transcripts_forward_NOT_overlap_reverse
transcripts_reverse_overlap_forward <-
  tot_transcripts_reverse - transcripts_reverse_NOT_overlap_forward
paste("Forward transcripts overlapped:", transcripts_forward_overlap_reverse)
```

```
## [1] "Forward transcripts overlapped: 3050"
```

```
paste("Reverse transcripts overlapped:", transcripts_reverse_overlap_forward)
```

```
## [1] "Reverse transcripts overlapped: 3207"
```

Afterwards, I used the wc Unix command to find the total number of transcripts on each strand (39593 on the forward amd 38217 on the reverse strand). Finally, I simply subtracted the number of transcripts, that do not overlap any transcript on the opposite strand, to the total number of transcript on that strand. Therefore, I obtained that the number of transcripts that overlap (at least) another transcript on the opposite strand is 3050, for the forward strand, and 3207 for the reverse strand.

```
# Calculate the percentage of overlapped transcripts using each strand as "query"
# (not necessary for the exercise)
fraction_forward_overlap_reverse <-
  transcripts_forward_overlap_reverse / tot_transcripts_forward
fraction_reverse_overlap_forward <-
  transcripts_reverse_overlap_forward / tot_transcripts_reverse
paste("Fraction of transcript in the forward strand that averlap is ",
      round(fraction_forward_overlap_reverse, 4), " (", round(fraction_forward_overlap_reverse, 4)
      * 100, " %)", sep = "")
```

```
## [1] "Fraction of transcript in the forward strand that averlap is 0.077 (7.7 %)"
```

```
paste("Fraction of transcript in the reverse strand that averlap is ",
      round(fraction_reverse_overlap_forward, 4), " (", round(fraction_reverse_overlap_forward, 4)
      * 100, " %)", sep = "")
```

```
## [1] "Fraction of transcript in the reverse strand that averlap is 0.0839 (8.39 %)"
```

```
# Calculate the (total) fraction of overlapped transcripts
total_fraction_overlap <-
  (transcripts_forward_overlap_reverse + transcripts_reverse_overlap_forward) /
                          (tot_transcripts_forward + tot_transcripts_reverse)
paste("The total fraction of transcripts that overlap is " ,
      round(total_fraction_overlap, 4), " (", round(total_fraction_overlap, 4)
      * 100, " %)", sep = "")
```

```
## [1] "The total fraction of transcripts that overlap is 0.0804 (8.04 %)"
```

In order to obtain the fraction of the overlapped transcripts (considering both strands), I summed the number of overlapped transcripts obtained above (on both strands), then I divided this number for the total number of RefSeq transcripts on both strands. The resulting fraction of transcripts that overlap another transcript on the opposite strand is 0.0804 (8.04%).

- 2: What is the distribution of of overlaps on opposite strand? E.g: plot an appropriate histogram how many refseq transcripts have 1,2,3...N opposite strand overlapping transcripts?

Answer goes here

```
# Use intersect with the -c flag to ouput the overlaps distribution using each strand as "query"
nice bedtools intersect -a refseq_hg19_forward.bed -b refseq_hg19_reverse.bed -f 0.2 -c >
forward_overlap_reverse_count.bed
nice bedtools intersect -a refseq_hg19_reverse.bed -b refseq_hg19_forward.bed -f 0.2 -c >
reverse_overlap_forward_count.bed
```

Using each strand as "query", I used the -c flag of the intersect function to output the distribution of the overlaps number on opposite strand. Then, I loaded them into R to plot an appropriate histogram.

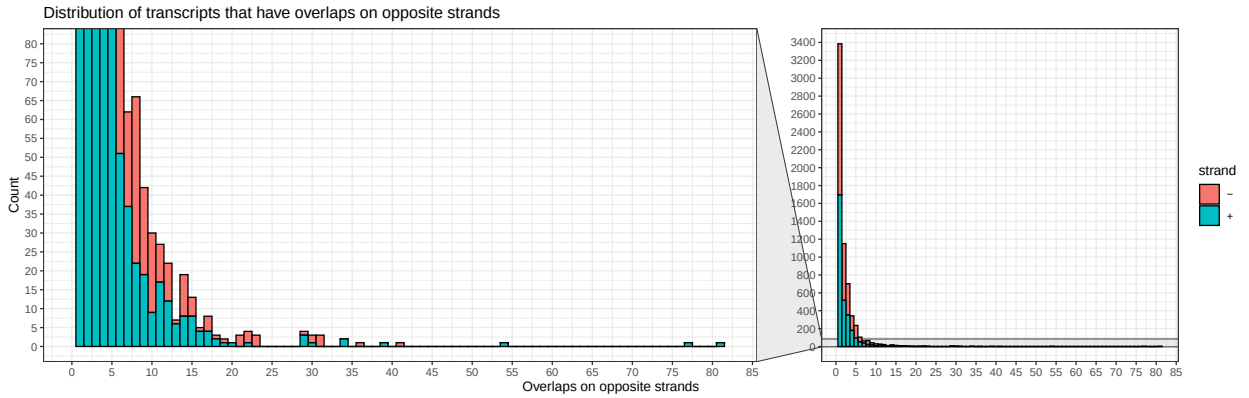
```
# Read opposite strand overlaps counts for each strand
col_names <- c(X1 = "Chr",            X2 = "chromStart",
               X3 = "chromEnd",       X4 = "transcript_ID",
               X5 = "score",          X6 = "strand",
               X7 = "cdsStart",       X8 = "cdsEnd",
               X9 = "itemRgb",        X10 = "exonsCount",
               X11 = "exonsSize",     X12 = "exonsStart",
               X13 = "overlaps_count")

forward_overlap_reverse_count <- read_tsv("forward_overlap_reverse_count.bed",
                                          col_names = col_names)

reverse_overlap_forward_count <- read_tsv("reverse_overlap_forward_count.bed",
                                          col_names = col_names)

# Plot overlaps distribution
bind_rows(forward_overlap_reverse_count, reverse_overlap_forward_count) %>%
  filter(overlaps_count > 0) %>%
  ggplot(aes(x = overlaps_count, fill = strand)) +
  geom_histogram(binwidth = 1, color = "black") +
    theme(legend.position = "none") +
```

```
  labs(title = "Distribution of transcripts that have overlaps on opposite strands",
       color = "Strand") +
  ylab("Count") +
  xlab("Overlaps on opposite strands") +
  facet_zoom(ylim=c(0, 80), shrink = FALSE) +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 20)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 20)) +
  theme_bw()
```



- 3: What is the Refseq transcript with the largest number of overlapping transcripts on the other strand (same thresholds as above)? Show this in the genome browser, disccuss the image and suggest how we can improve the anlaysis based on this (max 200 words).

Answer goes here

```
# Arrange by descending order, and select most overlapped transcript
bind_rows(forward_overlap_reverse_count, reverse_overlap_forward_count) %>%
  arrange(desc(overlaps_count)) %>%
  head(1) -> most_overlapped_transcript

# Show most osverlapped transcript
most_overlapped_transcript %>%
  select(Chr, chromStart, chromEnd, transcript_ID, strand, overlaps_count)
```

```
## # A tibble: 1 x 6
##   Chr   chromStart chromEnd transcript_ID strand overlaps_count
##   <chr>      <dbl>    <dbl> <chr>         <chr>           <dbl>
## 1 chr18   34854422 34856363 NR_134588.1   +                  81
```

```
# Select columns needed for the Genome Browser upload and save file
most_overlapped_transcript %>%
  select(Chr, chromStart, chromEnd) %>%
  write_tsv("most_overlapped_transcript.bed", col_names = FALSE)
```

In Figure 1, we can see the transcript NR_134588.1 (forward strand) and the 81 overlapping transcripts (reverse strand) found on the 18 chromosome. The added custom track (shown in red), with the coordinate of the most overlapped transcript obtained by the analysis, is located at the top of the picture. The corresponding RefSeq gene is the LOC105372068 gene and is shown at the bottom. In between there are 81 overlapping transcripts found on the opposite strand, they are 81 transcript variants of the CELF4 gene. They are members of the CELF/BRUNOL protein family, which is involved in pre-mRNA alternative splicing regulation [1]. NR_134588.1 transcript is a long non-coding RNA (lncRNA) [2], lncRNAs can be classified into different subtypes (Antisense and others) depending on the position and the direction of transcription in relation to other genes [3].

So, I hypothesize that NR_134588.1 is an Antisense RNA (asRNA) involved in the post-transcriptional regulation of the CELF4 transcrip variants.
One problem of the performed analysis is that, in our "query", we included both mRNAs that are overlapped by an asRNA on the opposite strand, and asRNAs that are overlapped by mRNAs. One way to improve it, and make it more specific, is to look for transcripts that are overlapped by an mRNA (or a coding region) on the opposite strand.

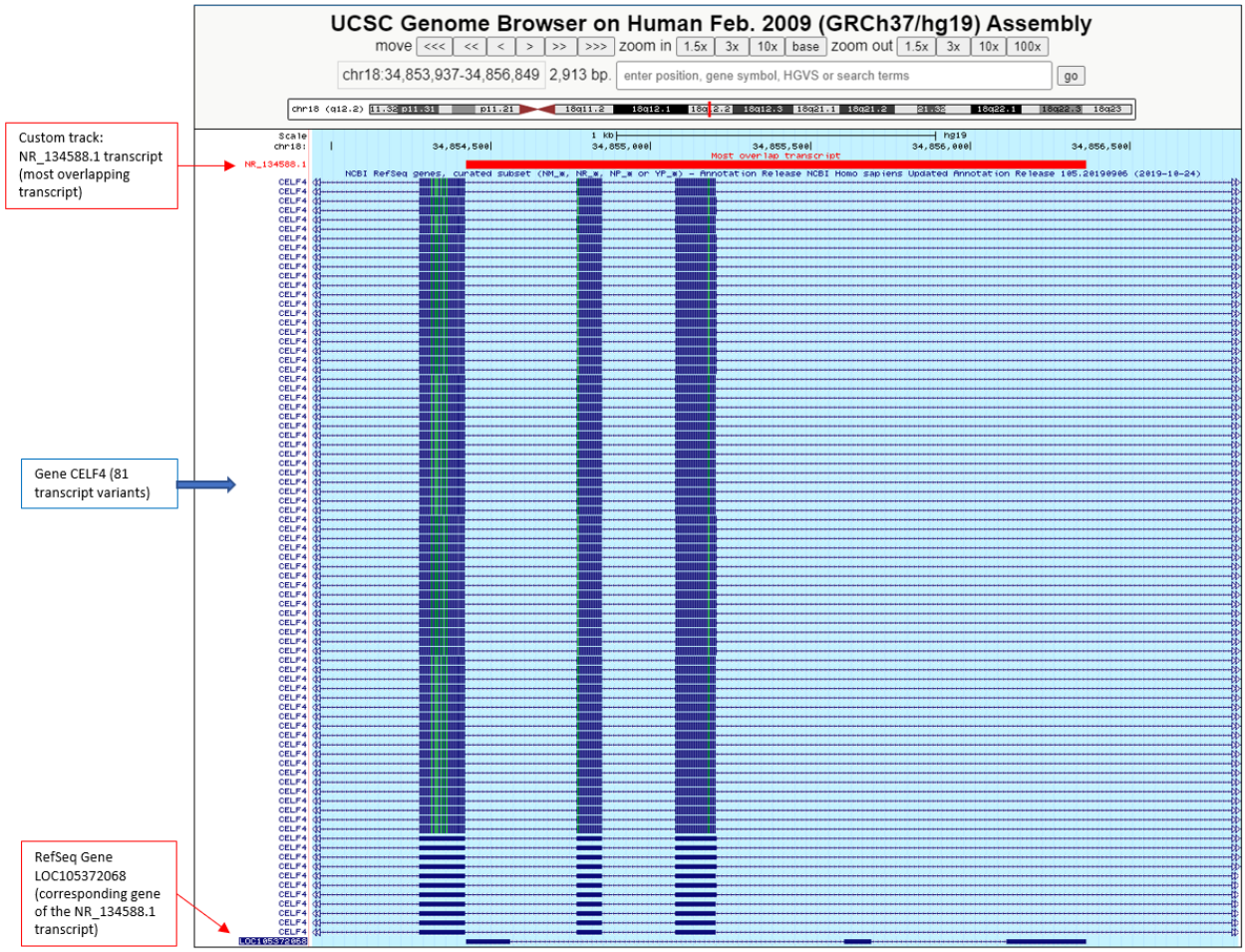


Figure 1: *Figure 1: Refseq transcript with the largest number of overlapping transcripts on the other strand, shown on the UCSC Genome Browser.*

Question 2

Data is available in the folder `question2_data`

Introduction

The RRP40 gene is part of the exosome complex, a molecular machine that degrades RNAs in the cells from the 3' end. Your collaborator has just made a CAGE experiment in cells in which RRP40 was depleted using a RRP40-specific siRNA. For comparison, he/she also made a control experiment where a random siRNA was used. The hope is to be able to identify what RNAs that are degraded by RRP40, becasue we should observe higher levels of these if RRP40 is depleted. They have an hypothesis that there may be RNAs transcription initiation close to known gene transcription start sites (TSSs) that we never observe in normal cells becasue the RNAs are eaten up fast.

The CAGE reads are already mapped to the genome and another collaborator has already made some files for you that shows where they fall around annotated TSSs. Specifically, you are given 4 files where the rows are the -600 to +400 region around ~12.000 annotated TSSs, and the columns are the positions (so, the first is position -600 etc for respective TSS). Values are TPM-normalized CAGE counts plus a small amount of artificially added noise to avoid model overfitting, aside from the first column which is just the genomic position we are looking at, eg chr4:10000-11000+.

Because you have two experiments and two strands, you have four files in total. For example, the `Hw4_CAGE_rrp40_senseStrand` file has CAGE data on the plus strand and from the RRP40 depletion experiment. "Strand" is here always relative to the annotated TSS, which is always defined to be on the plus strand.

These files are quite big (around 12 million data points, or 60-70 megabyte each), and your collaborators belatedly realized they could not plot these using Excel. Panic ensued. This is why they hired you: you know how to use R and your job is now to analyze the data and visualizing the results.

Specifically, they want you to:

Using tidyverse, make a plot where the Y axis is average fold change (rrp40/ctrl) and X axis is position relative to TSS (-600 to +400). Calculate fold changes for each strand so in the end, you will have a plot with two fold change 'lines', one for each strand. Interpret the results: What are we seeing and does this agree with the text-book decription of promoters and transcription start sites? (max 100 words)

Tip: If numeric columns become characters during your tidyverse manipulations and you want them to be numeric, `as.numeric()` is a good function.

Answer goes here

Data loading and inspection

```
# Load data
cage_control_sense <- read_tsv("question2_data/Hw4_CAGE_Ctrl_senseStrand.txt")
cage_control_antisense <- read_tsv("question2_data/Hw4_CAGE_Ctrl_antisenseStrand.txt")
cage_rrp40_sense <- read_tsv("question2_data/Hw4_CAGE_rrp40_senseStrand.txt")
cage_rrp40_antisense <- read_tsv("question2_data/Hw4_CAGE_rrp40_antisenseStrand.txt")

# Inspect data (not showed in the HTML for clarity)
head(cage_control_sense)
head(cage_control_antisense)
head(cage_rrp40_sense)
head(cage_rrp40_antisense)
```

```
dim(cage_control_sense)
```

```
## [1] 12213  1001
```

```
dim(cage_control_antisense)
```

```
## [1] 12213  1001
```

```
dim(cage_rrp40_sense)
```

```
## [1] 12213  1001
```

```
dim(cage_rrp40_antisense)
```

```
## [1] 12213  1001
```

```
# Remove information about chromosome
cage_control_sense <- cage_control_sense[-1]
```

```
cage_control_antisense <- cage_control_antisense[-1]
cage_rrp40_sense <- cage_rrp40_sense[-1]
cage_rrp40_antisense <- cage_rrp40_antisense[-1]
```

Sense strand

```
## Create merged dataset with FC for the sense strand
#  (log2 is applied to the average FC for more resolution)

# Convert data to long format
cage_control_sense %>%
  gather(key = "Position", value = "TPM_control") -> cage_control_sense_long
cage_rrp40_sense %>%
  gather(key = "Position", value = "TPM_RRP40") -> cage_rrp40_sense_long

# Merge control and treatment data
bind_cols(cage_control_sense_long, cage_rrp40_sense_long) %>%
  mutate(Fold_change = (TPM_RRP40 / TPM_control)) %>%
  mutate(Position = as.numeric(gsub("pos.", "", Position...1))) %>%
  mutate(Position = Position - 601) %>%
  group_by(Position) %>%
  summarize(log2Mean_FC = log2(mean(Fold_change))) %>%
  mutate(Strand = "Sense") -> log2mean_FC_sense
```

```
## New names:
## * Position -> Position...1
## * Position -> Position...3
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(log2mean_FC_sense)
```

```
## # A tibble: 6 x 3
##   Position log2Mean_FC Strand
##      <dbl>       <dbl> <chr>
## 1     -600     0.00924 Sense
## 2     -599     0.0126  Sense
## 3     -598     0.00719 Sense
## 4     -597     0.00729 Sense
## 5     -596     0.0118  Sense
## 6     -595     0.0126  Sense
```

Antisense strand

```
## Create merged dataset with FC for the antisense strand
#  (log2 is applied to the average FC for more resolution)

# Convert data to long format
cage_control_antisense %>%
  gather(key = "Position", value = "TPM_control") -> cage_control_antisense_long
cage_rrp40_antisense %>%
  gather(key = "Position", value = "TPM_RRP40") -> cage_rrp40_antisense_long

# Merge control and treatment data
bind_cols(cage_control_antisense_long, cage_rrp40_antisense_long) %>%
```

```
  mutate(Fold_change = (TPM_RRP40 / TPM_control)) %>%
  mutate(Position = as.numeric(gsub("pos.", "", Position...1))) %>%
  mutate(Position = Position - 601) %>%
  group_by(Position) %>%
  summarize(log2Mean_FC = log2(mean(Fold_change))) %>%
  mutate(Strand = "Antisense") -> log2mean_FC_antisense
```

```
## New names:
## * Position -> Position...1
## * Position -> Position...3
```

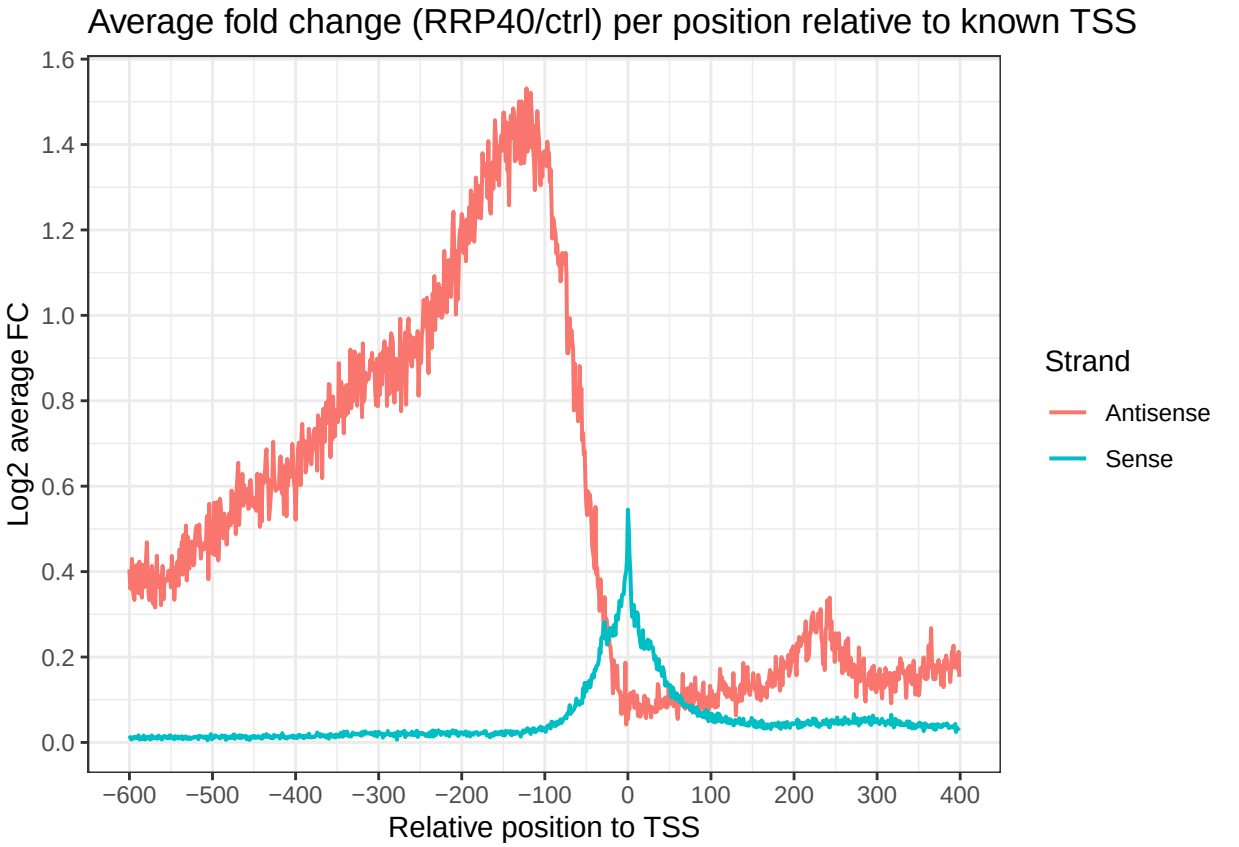
```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
head(log2mean_FC_antisense)
```

```
## # A tibble: 6 x 3
##   Position log2Mean_FC Strand
##      <dbl>       <dbl> <chr>
## 1     -600       0.405 Antisense
## 2     -599       0.360 Antisense
## 3     -598       0.366 Antisense
## 4     -597       0.430 Antisense
## 5     -596       0.359 Antisense
## 6     -595       0.343 Antisense
```

Plot both sense and antisense (log2) average fold change for each position relative to TSS

```
# Plot
bind_rows(log2mean_FC_sense, log2mean_FC_antisense) %>%
  ggplot(aes(x=Position, y=log2Mean_FC, col=Strand)) +
  geom_line(size=0.7) +
  labs(title = "Average fold change (RRP40/ctrl) per position relative to known TSS") +
  ylab("Log2 average FC") +
  xlab("Relative position to TSS") +
  scale_y_continuous(breaks = scales::pretty_breaks(n = 10)) +
  scale_x_continuous(breaks = scales::pretty_breaks(n = 10)) +
  theme_bw()
```



We can see that the greatest increase in transcription initiation activity occurs upstream the known TSSs, and in reverse direction respect to the transcription of the known genes. One interpretation is that we are detecting promoter-upstream transcripts (PROMPTs), and that RRP40 is involved in their degradation. In fact, the RRP40 gene, or EXOSC3, is part of the exosome complex RNA-degradation machine [4]. In the nucleus, it is involved in the degradation of non-coding transcripts, such as Antisense RNA and PROMPTs [4]. PROMPTs are transcribed in reverse orientation to most active protein coding genes [5] and are produced only upstream of their promoters, also, they are only detectable when RNA degradation factors are compromised [6].

Question 3

Data is available in the folder `question3_data`

Introduction

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 a new experimental drug for treatment of type 2 diabetes. Novo Nordisk wants to investigate how the drug 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.

You have been supplied with 4 files:

- `studyDesign.tsv`: File describing treatment of the 18 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.
- `diabetesGenes.tsv`: Collection of genes known to be involved in type 2 diabetes.

Part 1: Exploratory Data Analysis

Question 3.1.1: Read all dataset into R, and make sure all three files have matching numbers and names of both samples and genes.

Answer goes here

Data loading and inspection

```
# Load data
design_matrix <- read_tsv("question3_data/studyDesign.tsv")
count_matrix <- read.table("question3_data/countMatrix.tsv")
normalized_matrix <- read.table("question3_data/normalizedMatrix.tsv")
known_genes <- read_tsv("question3_data/diabetesGenes.tsv")

## Inspect data and check dimensions
# Deisgn matrix
dim(design_matrix)
```

```
## [1] 18  2
```

```
head(design_matrix)
```

```
## # A tibble: 6 x 2
##   Sample  Condition
##   <chr>   <chr>
## 1 Sample1 Ctrl
## 2 Sample2 Ctrl
## 3 Sample3 Ctrl
## 4 Sample4 Ctrl
## 5 Sample5 Ctrl
## 6 Sample6 Ctrl
```

```
# Count_matrix
dim(count_matrix)
```

```
## [1] 5669   18
```

```
head(count_matrix)
```

```
##       Sample1 Sample2 Sample3 Sample4 Sample5 Sample6 Sample7 Sample8 Sample9
## Gene1      42      43      37      16      23      11      25      29       3
## Gene2       6      15      28      65      54       5      41      40      39
## Gene3      25      35       4      22      31     284      60      49      25
## Gene4     108     154     129     121       2     599      81      80     141
## Gene5      44      17      64      29      84      34       4       7      60
## Gene6      31       2      16       9      11       3      29       5       2
##       Sample10 Sample11 Sample12 Sample13 Sample14 Sample15 Sample16 Sample17
## Gene1       36       43       54       47       40        5       42       26
## Gene2        1        1        1        0        2        3        2        1
## Gene3      468      248      173       30      160      357      302      435
## Gene4      384      254      245      646      149       58      296      308
## Gene5       46       44       35       43       41       40       67        1
## Gene6       53       18       20       42       48        2       15       15
##       Sample18
## Gene1       23
## Gene2       76
## Gene3       98
```

```
## Gene4       69
## Gene5       47
## Gene6       15
```

```
# Normalized matrix
dim(normalized_matrix)
```

```
## [1] 5669   18
```

```
head(normalized_matrix)
```

```
##        Sample1  Sample2  Sample3  Sample4  Sample5  Sample6  Sample7  Sample8
## Gene1 5.166482 5.024020 5.006239 3.979371 4.504762 3.495344 4.582357 4.612975
## Gene2 2.711949 3.491176 4.330378 5.157526 5.105973 2.289754 4.771890 4.606759
## Gene3 5.029787 5.219791 3.412445 4.744161 5.249882 7.458622 5.957103 5.600569
## Gene4 6.801304 7.030640 6.982561 6.761835 3.367099 8.485103 6.462118 6.315323
## Gene5 5.284018 4.098349 5.688550 4.662541 6.016682 4.704378 2.934431 3.275561
## Gene6 4.567927 1.957353 3.835834 3.130305 3.466407 2.120249 4.477463 2.618506
##        Sample9 Sample10 Sample11 Sample12  Sample13 Sample14 Sample15 Sample16
## Gene1 2.663626 4.679403 5.037823 5.104823 5.1070199 4.789645 2.935055 4.826393
## Gene2 4.704789 1.219430 1.328357 1.206473 0.7881749 1.574962 1.986891 1.562762
## Gene3 5.005122 8.029113 7.450092 6.830384 5.0426678 6.756100 7.886965 7.483198
## Gene4 7.077614 7.931380 7.622766 7.385417 8.6977386 6.832118 5.962909 7.603868
## Gene5 5.608467 5.011376 5.128183 4.692784 5.0730859 4.880430 5.025056 5.410576
## Gene6 2.056138 4.846484 3.832054 3.767546 4.7226053 4.731024 1.968603 3.471533
##       Sample17 Sample18
## Gene1 4.469343 4.379619
## Gene2 1.313364 5.371754
## Gene3 8.096946 6.389199
## Gene4 7.824356 6.166998
## Gene5 2.080504 5.220734
## Gene6 3.622297 3.660755
```

```
# Check if genes names match between count and normalized matrix
sum(rownames(count_matrix) != rownames(normalized_matrix))
```

```
## [1] 0
```

```
# Check if samples names match between count and normalized matrix
sum(colnames(count_matrix) != colnames(normalized_matrix))
```

```
## [1] 0
```

```
# Check if the samples name of the count/normalized matrix match the design matrix
sum(colnames(count_matrix) != design_matrix$Sample)
```

```
## [1] 0
```

I can see that the design matrix contains 18 samples, 9 treatment and 9 control. Both counts and normalized matrixes contain 18 samples and 5669 genes. Both samples and genes names match between the three files.

Question 3.1.2: Heat map: For heat maps, it makes no sense to include all genes - instead, we will only look at genes that vary substantially across the samples. Specifically, select the genes top 10% of genes based on their variance across all samples, and make a heat map of those using the pheatmap library (standard settings). Rows in the heat mpa should be genes, columns shoudl be samples. Make an annotation row that shows whether each sample is treatment or control. Comment on your plot

Answer goes here

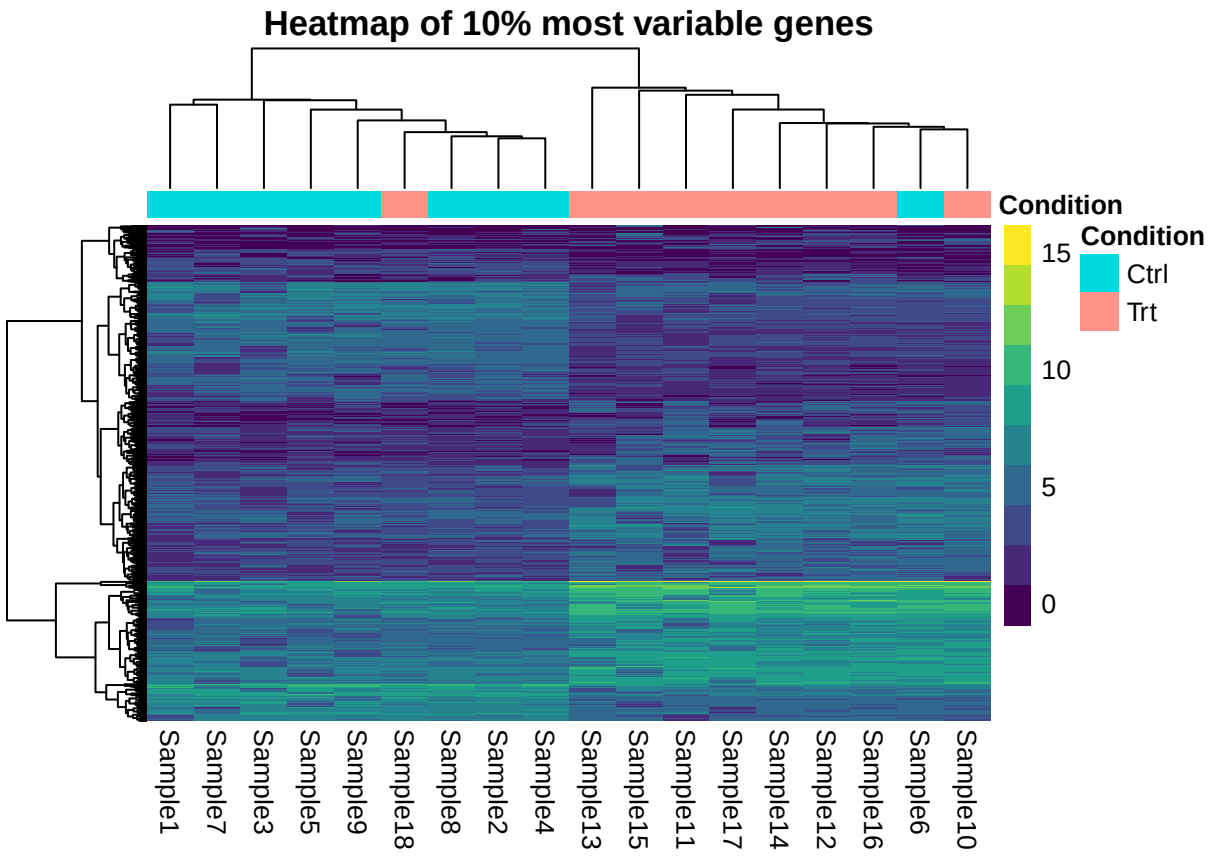
Heatmap

```
# Compute the 10% of the genes number
ten_percent <- round(dim(normalized_matrix)[1] * (10/100))

# Select top 10% of the genes based on their variance
normalized_matrix %>%
  mutate(Variance = apply(normalized_matrix, 1, var)) %>%
  arrange(desc(Variance)) %>%
  head(ten_percent) %>%
  select(-Variance) -> ten_percent_by_var

# Convert design matrix to the format for the heatmap annotation
condition_df <- data.frame("Condition" = design_matrix$Condition)
rownames(condition_df) <- design_matrix$Sample

# Generate an heatmap with an annotation row (sample condition)
pheatmap(ten_percent_by_var,
         annotation_col=condition_df,
         show_rownames = FALSE,
         color=viridis(10),
         main = "Heatmap of 10% most variable genes")
```



I generated an heatmap with the top 10% of the genes based on their variance. The heatmap is generated without the rows normalization, so it shows the "absolute" transcripts abundance (TPM) for each gene in each sample. The columns rapresent the samples and the rows rapresent the genes. The clusters division showed by the trees and the colors of the heatmap is performed by hierarchical clustering (Euclidean distance).

It is possible to see that there are two main clusters of genes. On the top of the heatmap, there is a larger gene cluster that contains genes with a lower expression and, on the bottom there is a smaller cluster with greater expressed genes. Eventually, we can further divide the two main clusters. The larger gene cluster at the top can be divided in three subclusters and the smaller cluster at the bottom can be divided in two subclusters. Also, as expected, the samples are divided in two main clusters corresponding to the two experimental conditions. An annotation row indicates which condition corresponds to each sample. In this regard we can see that there is something unexpected. In fact, a treated sample is in the control cluster, and a control sample is in the treatment one. This may be the result of a labeling error.

Question 3.1.3: PCA: Using the normalized matrix (all genes, not the top 10% of genes as in the heat map), perform a Principal Components Analysis (PCA) on the samples and produce a PCA-plot of the two first components, where the axis labels show the amount of variance explained by each component and samples are colored by their experimental group. Find a way to label the samples, so the identity (the sample name) of each point can easily be seen (hint: look at `geom_text()` or the ggrepel package!). Note, you should center but not scale the data. Comment on your plot

Answer goes here

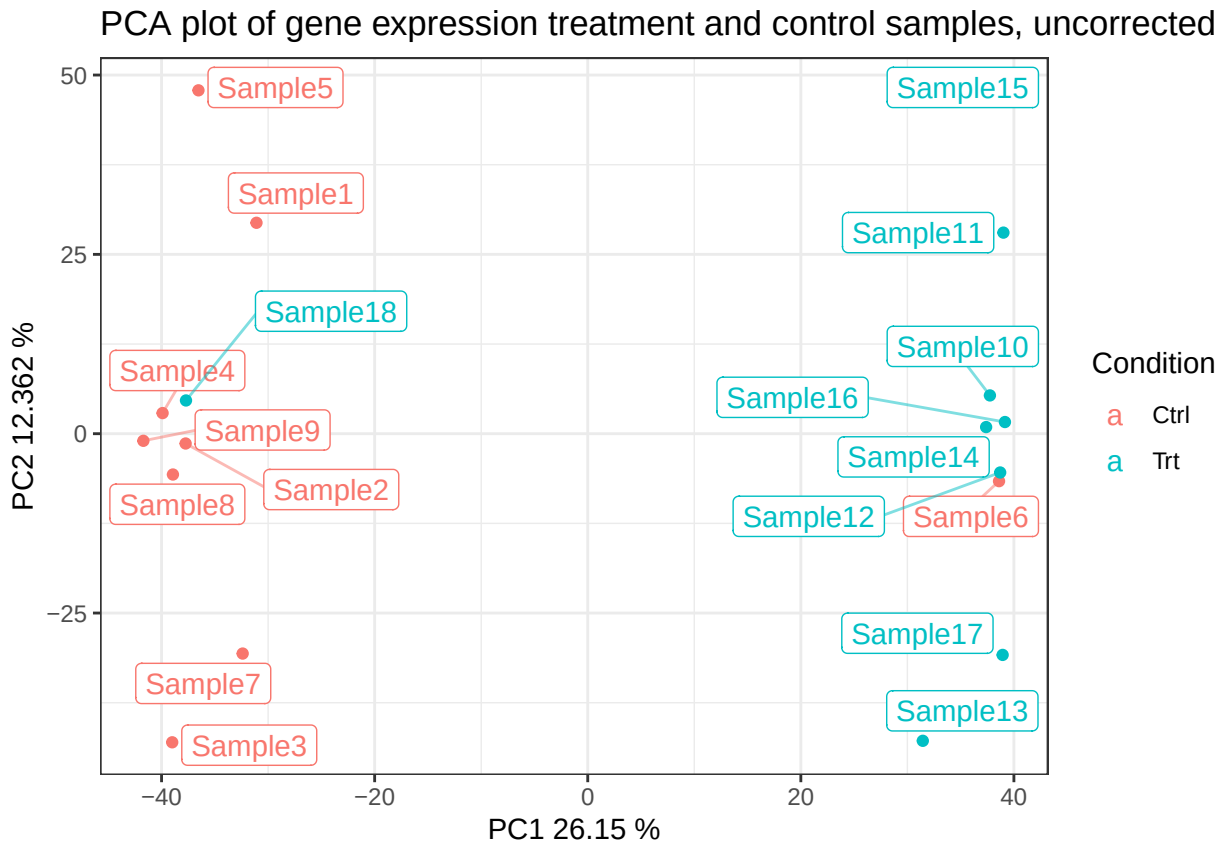
PCA un-corrected

```
# Convert the normalized matrix (from data frame to matrix) and transpose it
normalized_matrix %>%
  as.matrix %>% t -> normalized_matrix_t

# Center the data and erform PCA
pca_norm_matrix <- prcomp(normalized_matrix_t, center=TRUE)

# Extract prportion of variance captured by each principal component
percent_variance <- summary(pca_norm_matrix)$importance["Proportion of Variance",] * 100

# Plot the data projected in the new dimensions defined by the two PCs
as_tibble(pca_norm_matrix$x) %>%
  ggplot(aes(x = PC1, y = PC2,
             label = design_matrix$Sample,
             col = design_matrix$Condition)) +
  geom_point() +
  geom_label_repel(segment.alpha = 0.5) +
  labs(title = "PCA plot of gene expression treatment and control samples, uncorrected",
       color = "Condition") +
  xlab(label = paste("PC1", percent_variance[1], "%")) +
  ylab(label = paste("PC2", percent_variance[2], "%")) +
  theme_bw()
```



I performed a PCA and I plotted the 18 samples projected in the space defined by the two principal components capturing the largest amount of variance in the data (~26% and ~12%). In the PCA plot we can see that all treated samples, except one, are located on the right side of the plot, while all control samples, except one, are located on the left side. Basically, it confirms that there is a problem in the data, as previously observed in the annotation row of the heatmap. In fact, sample 18 is a treated sample and is located in a cluster of control samples, while sample 6 is a control sample and is located in a cluster of trated samples.

Also, it seems that there is a simmetric pattern in the distributions of the samples. There are two sub-groups along the middle horizontal line, each containing 5 samples. While other 4 samples, at each side of the plot, are simmetrically distributed below and above the sub-groups.

Question 3.1.4: Based on the two previous questions, discuss (max 50 words) whether your observations indicate that there are any problems with the data - e.g. outliers, mix ups, sub-groups. If you identified problems try to fix them (e.g. remove clear outliers if you find them, fix mix-ups, etc). If you make a correction, make a PCA plot with your corrected data to check that the correction is doing the right thing

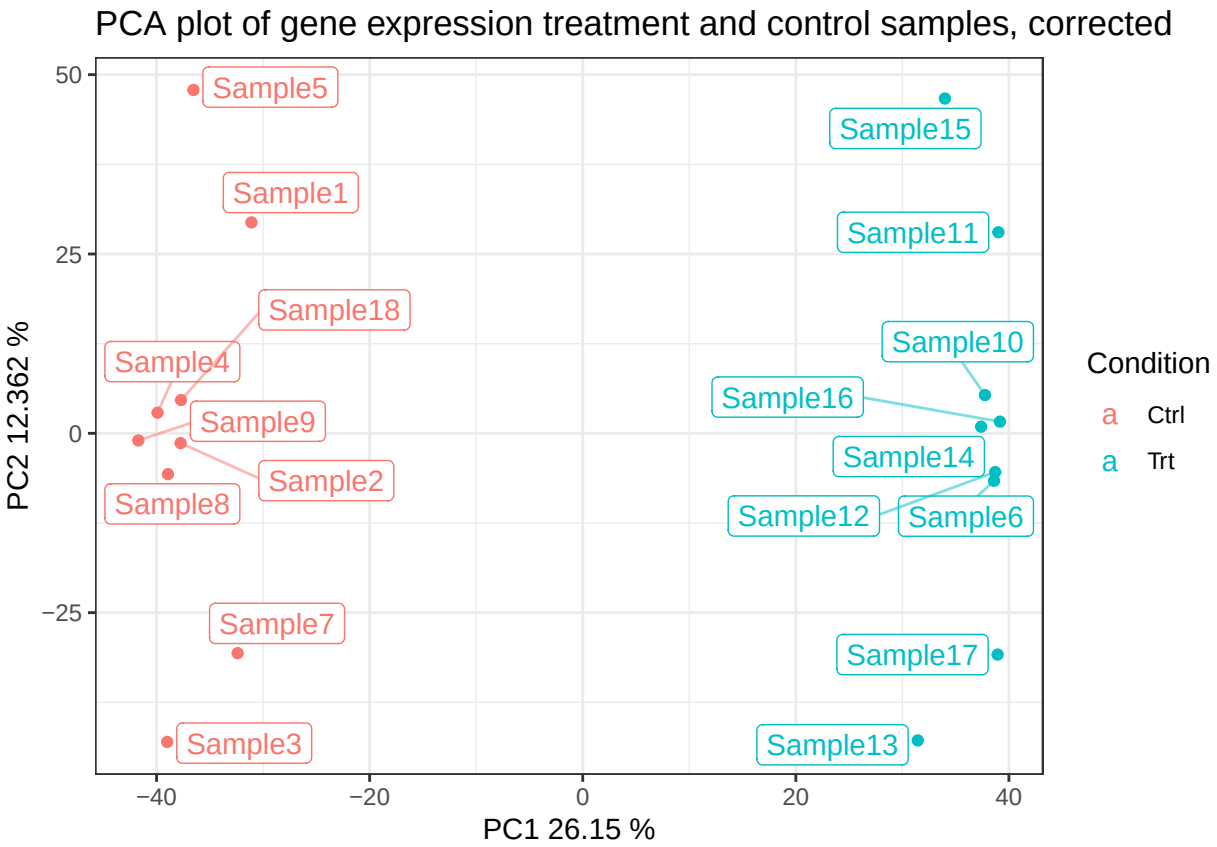
Answer goes here

PCA corrected

```
# Make a corrected copy of the design matrix
design_matrix_corrected <- design_matrix
design_matrix_corrected[c(6,18),2] <- design_matrix_corrected[c(18,6),2]

# PCA plot with corrected data
as_tibble(pca_norm_matrix$x) %>%
  ggplot(aes(x=PC1, y=PC2,
```

```
            label=design_matrix_corrected$Sample,
            col=design_matrix_corrected$Condition)) +
  geom_point() +
  geom_label_repel(segment.alpha = 0.5) +
  labs(title = "PCA plot of gene expression treatment and control samples, corrected",
       color = "Condition") +
  xlab(label = paste("PC1", percent_variance[1], "%")) +
  ylab(label = paste("PC2", percent_variance[2], "%")) +
  theme_bw()
```



Both the heatmap's annotation row and the previous PCA plot indicated that sample 6 was erroneously labelled as control and sample 18 was erroneously labeleed as treated. So, I swapped the two conditions of the two samples and now the PCA plot doesn't seem to show any problem in the data.

Part 2: Differential Expression (DE)

Question 3.2.1: Use DESeq2 to obtain differentially expressed (DE) genes between the two experimental conditions. Use default parameter, except use a logFC threshold of 0.25 and an adjusted P-value threshold of 0.05. How many up-and down regulated genes are there on your corrected data compared to if you do the Deseq2 analysis on un-corrected data?

Answer goes here

DESeq2

```
# Convert the count matrix data frame to a matrix
count_matrix %>% as.matrix() -> count_matrix
```

```r
## Differetential expression un-corrected data
# Save the data as DESeqDatSet-object
dds <- DESeqDataSetFromMatrix(countData = count_matrix,
                              colData = design_matrix,
                              design = ~ Condition)
# Running DEseq2
dds <- DESeq(dds)
# Look at the results
res <- results(dds,
               lfcThreshold = 0.25,
               alpha = 0.05)
paste("DESeq un-corrected data:")
```

```
## [1] "DESeq un-corrected data:"
```

```r
summary(res)
```

```
##
## out of 5669 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0.25 (up)    : 160, 2.8%
## LFC < -0.25 (down) : 194, 3.4%
## outliers [1]       : 0, 0%
## low counts [2]     : 550, 9.7%
## (mean count < 2)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```r
## Repeat for differetential expression corrected data
dds_corrected <- DESeqDataSetFromMatrix(countData = count_matrix,
                                        colData = design_matrix_corrected,
                                        design = ~ Condition)

dds_corrected <- DESeq(dds_corrected)
res_corrected <- results(dds_corrected,
                         lfcThreshold = 0.25,
                         alpha = 0.05)
paste("DESeq corrected data:")
```

```
## [1] "DESeq corrected data:"
```

```r
summary(res_corrected)
```

```
##
## out of 5669 with nonzero total read count
## adjusted p-value < 0.05
## LFC > 0.25 (up)    : 653, 12%
## LFC < -0.25 (down) : 873, 15%
## outliers [1]       : 0, 0%
## low counts [2]     : 110, 1.9%
## (mean count < 1)
## [1] see 'cooksCutoff' argument of ?results
## [2] see 'independentFiltering' argument of ?results
```

```r
## Compute ratio of down and up regulated genes between corrected and uncorrected data
Corrected_Uncorrected_UP_ratio <- 653 / 160
```

```
Corrected_Uncorrected_UP_ratio
```

```
## [1] 4.08125
```

```
Corrected_Uncorrected_DOWN_ratio <- 873 / 194
Corrected_Uncorrected_DOWN_ratio
```

```
## [1] 4.5
```

I performed a differnetial expression analysis using DESeq2 on both corrected and un-corrected data. The results are filtered using a log2 fold change threshold of 0.25 and an adjusted P-value threshold of 0.05. In the corrected data there are 653 (12%) up regulated genes and 873 (15%) down regulated genes, while in un-corrected data there are 160 (2.8%) up regulated genes and 194 (3.4%) down regulated genes. So, in the corrected data there are approximately 4.1 times more up regulate genes and 4.5 times more down regulated genes than the uncorrected data.

Question 3.2.2: From now on, we will only analyze the corrected data. Convert the output of DESeq2 (corrected data) to a tibble, and make an MA-plot using ggplot2. The MA-plot should show the overall trend using a trend line and genes should colored according to their DE status. Discuss whether the MA-plot indicates an approriate DESeq2 analysis (max 70 words discussion).

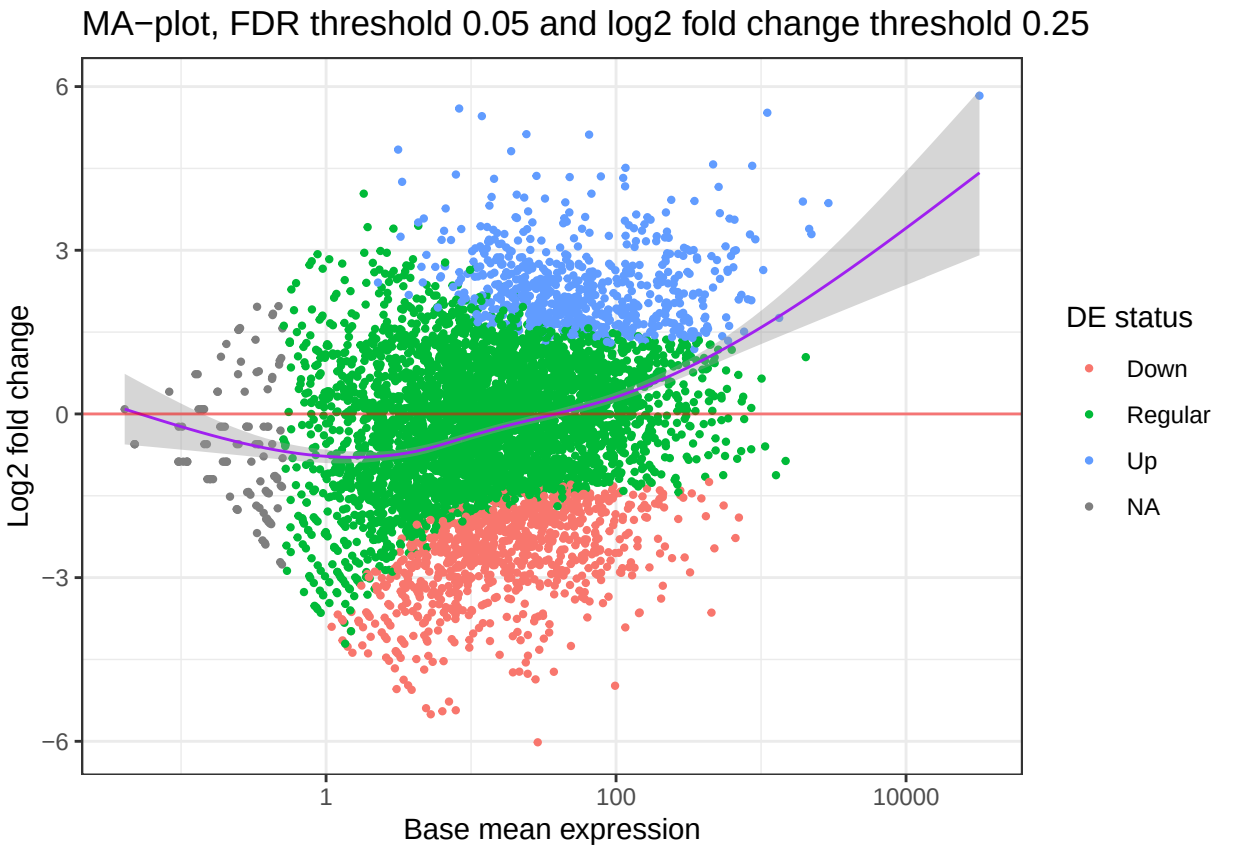
Answer goes here

MA-plot

```
# Convert the result of DESeq2 to a tibble
res_corrected %>% as.data.frame %>%
                  rownames_to_column("gene") %>%
                  as_tibble -> res_corrected_tibble
head(res_corrected_tibble)
```

```
## # A tibble: 6 x 7
##   gene  baseMean log2FoldChange lfcSE    stat   pvalue     padj
##   <chr>    <dbl>          <dbl> <dbl>   <dbl>    <dbl>    <dbl>
## 1 Gene1     25.8         0.0887 0.451  0      1.00e+ 0 1.00e+ 0
## 2 Gene2     19.4        -4.74   0.585 -7.66   1.79e-14 3.56e-12
## 3 Gene3    127.          2.60   0.465  5.06   4.25e- 7 1.04e- 5
## 4 Gene4    176.          1.51   0.478  2.63   8.56e- 3 3.44e- 2
## 5 Gene5     34.3        -0.299  0.528 -0.0937 9.25e- 1 1.00e+ 0
## 6 Gene6     15.8         0.558  0.595  0.517  6.05e- 1 8.37e- 1
```

```
# MA-Plot, color the data points according to the DE status (up, down, regular, NA)
res_corrected_tibble %>%
  mutate(DE = ifelse(padj < 0.05 & log2FoldChange > 0.25, "Up",
              ifelse(padj < 0.05 & log2FoldChange < 0.25, "Down", "Regular"))) %>%
ggplot(aes(x=baseMean, y=log2FoldChange, col=DE)) +
  geom_point(size=0.8) + geom_smooth(color="purple", size=0.5) + scale_x_log10() +
  geom_hline(yintercept = 0, alpha = 0.5, color="red") +
  labs(title = "MA-plot, FDR threshold 0.05 and log2 fold change threshold 0.25",
       color = "DE status") +
  ylab("Log2 fold change") +
  xlab("Base mean expression") +
  theme_bw()
```



The MA-plot can be used as diagnostic plot to check the results of the DESeq2 analysis. Here we can see that our data are not correctly normalized. In fact, the trend line, which indicates the actual average log2 fold change between the genes, should approximately correspond to zero and match the diagonal red line. In our case this is not true, in fact it is slightly "U-shaped" and it bends up on the right side of the plot.

Question 3.2.3: Sort the DE statistics table that you get from DESeq2 to report the top 10 genes sorted by

a) positive logFC (highest on top)

Answer goes here

Up regulated genes

```
# Filter by P-value threshold (for significantly expressed genes) and sort by descending FC
res_corrected_tibble %>%
  filter(padj < 0.05) %>%
  arrange(desc(log2FoldChange)) %>%
  head(10)
```

```
## # A tibble: 10 x 7
##    gene     baseMean log2FoldChange lfcSE  stat   pvalue     padj
##    <chr>       <dbl>          <dbl> <dbl> <dbl>    <dbl>    <dbl>
##  1 Gene861  32094.             5.83 0.455 12.3  1.18e-34 3.94e-31
##  2 Gene4975     8.28           5.60 1.48   3.62 2.96e- 4 2.32e- 3
##  3 Gene4582  1105.             5.52 0.430 12.3  1.42e-34 3.94e-31
##  4 Gene5510    11.9            5.46 1.35   3.86 1.16e- 4 1.08e- 3
##  5 Gene541     24.1            5.13 0.741  6.58 4.75e-11 3.89e- 9
```

```
##  6 Gene2038    65.2            5.12 0.902  5.40 6.75e- 8 2.19e- 6
##  7 Gene1950     3.14           4.84 1.14   4.04 5.26e- 5 5.62e- 4
##  8 Gene4259    18.9            4.82 1.20   3.80 1.44e- 4 1.28e- 3
##  9 Gene5215   467.             4.57 0.447  9.67 4.21e-22 5.85e-19
## 10 Gene1910   869.             4.55 0.496  8.66 4.83e-18 2.68e-15
```

b) negative logFC (lowest on top)

Answer goes here

Down regulated genes

```
# Filter for adjusted P-value threshold and sort by ascending FC
res_corrected_tibble %>%
  filter(padj < 0.05) %>%
  arrange(log2FoldChange) %>%
  head(10)
```

```
## # A tibble: 10 x 7
##    gene     baseMean log2FoldChange lfcSE   stat   pvalue     padj
##    <chr>       <dbl>          <dbl> <dbl>  <dbl>    <dbl>    <dbl>
##  1 Gene2502    28.8           -6.02 0.668  -8.63 6.21e-18 3.14e-15
##  2 Gene5604     5.27          -5.50 0.971  -5.41 6.14e- 8 2.06e- 6
##  3 Gene5099     6.34          -5.45 0.873  -5.96 2.59e- 9 1.28e- 7
##  4 Gene1613     7.85          -5.43 0.944  -5.49 4.07e- 8 1.50e- 6
##  5 Gene3160     4.89          -5.39 0.907  -5.67 1.40e- 8 5.99e- 7
##  6 Gene3593     7.04          -5.27 1.03   -4.87 1.14e- 6 2.38e- 5
##  7 Gene77       3.89          -5.06 1.08   -4.47 7.85e- 6 1.23e- 4
##  8 Gene4597     3.07          -5.04 1.06   -4.51 6.59e- 6 1.06e- 4
##  9 Gene1056    98.5           -4.98 0.449 -10.5  5.27e-26 9.77e-23
## 10 Gene5282     3.68          -4.97 1.11   -4.26 2.09e- 5 2.64e- 4
```

only looking at significantly differentially expressed genes

Part 3: Is the drug any good?

Question 3.3.1: Novo Nordisk claims their treatment affects expression of 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. Are these genes more up/down regulated than expected by chance, by looking at log2FC values from above ?

Answer goes here

Diabete related genes

```
# Up regulated genes, sorted by log2 fold change
res_corrected_tibble %>%
  filter(gene %in% known_genes$Gene, padj < 0.05, log2FoldChange > 0.25) %>%
  arrange(desc(log2FoldChange))
```

```
## # A tibble: 18 x 7
##    gene     baseMean log2FoldChange lfcSE  stat   pvalue         padj
##    <chr>       <dbl>          <dbl> <dbl> <dbl>    <dbl>        <dbl>
##  1 Gene5284    20.3            3.61 0.999  3.36 7.84e- 4 0.00495
##  2 Gene5258   157.             3.48 0.523  6.18 6.59e-10 0.0000000370
##  3 Gene2103    25.6            3.01 0.610  4.52 6.28e- 6 0.000102
##  4 Gene5554    32.9            2.96 0.603  4.50 6.70e- 6 0.000108
##  5 Gene1241   496.             2.82 0.475  5.41 6.33e- 8 0.00000209
```

```
##  6 Gene1013     25.6           2.67 0.876  2.76 5.75e- 3 0.0250
##  7 Gene1940     74.5           2.57 0.523  4.44 9.11e- 6 0.000137
##  8 Gene5271     11.2           2.52 0.761  2.98 2.86e- 3 0.0143
##  9 Gene1187     33.2           2.34 0.517  4.05 5.20e- 5 0.000559
## 10 Gene1585     34.5           2.23 0.545  3.63 2.81e- 4 0.00222
## 11 Gene1114     30.0           2.08 0.585  3.14 1.71e- 3 0.00928
## 12 Gene4542    221.            2.04 0.419  4.27 1.95e- 5 0.000252
## 13 Gene4040     42.4           1.99 0.547  3.18 1.48e- 3 0.00828
## 14 Gene896      19.4           1.95 0.640  2.65 8.05e- 3 0.0328
## 15 Gene1779     50.6           1.89 0.442  3.70 2.15e- 4 0.00179
## 16 Gene4329    189.            1.80 0.467  3.32 9.09e- 4 0.00556
## 17 Gene4060     11.1           1.68 0.518  2.76 5.72e- 3 0.0249
## 18 Gene3804     69.5           1.66 0.475  2.97 2.99e- 3 0.0148
```

```
# Down regulated genes, sorted by log2 fold change
res_corrected_tibble %>%
  filter(gene %in% known_genes$Gene, padj < 0.05, log2FoldChange < 0.25) %>%
  arrange(log2FoldChange)
```

```
## # A tibble: 13 x 7
##    gene     baseMean log2FoldChange lfcSE  stat       pvalue        padj
##    <chr>       <dbl>          <dbl> <dbl> <dbl>        <dbl>       <dbl>
##  1 Gene836      7.81          -3.76 0.773 -4.54 0.00000561   0.0000934
##  2 Gene4703     1.89          -3.64 1.19  -2.84 0.00457      0.0209
##  3 Gene2984     3.83          -3.13 0.847 -3.40 0.000678     0.00443
##  4 Gene4720    74.6           -2.95 0.471 -5.73 0.0000000101 0.000000438
##  5 Gene2692     4.28          -2.70 0.896 -2.73 0.00630      0.0270
##  6 Gene4691    12.7           -2.39 0.526 -4.08 0.0000456    0.000508
##  7 Gene318     72.2           -2.04 0.429 -4.18 0.0000294    0.000355
##  8 Gene5651     5.25          -1.94 0.655 -2.58 0.00974      0.0382
##  9 Gene1391    14.2           -1.90 0.589 -2.80 0.00509      0.0226
## 10 Gene102     22.2           -1.88 0.499 -3.27 0.00106      0.00628
## 11 Gene5458    26.2           -1.63 0.521 -2.65 0.00805      0.0328
## 12 Gene2285    18.0           -1.61 0.543 -2.51 0.0120       0.0450
## 13 Gene1448    25.9           -1.59 0.499 -2.68 0.00735      0.0307
```

The differential expression analysis perform the Wald test to see if the genes are differentially expressed between the two conditions. Indeed, the null hypothesis is that a gene is not differentially expressed. The P-value indicates how likely we can observe our result, under the assumption that the null hypothesis is true. Since we are testing a larger number of genes, we use an adjusted P-value (false discovery rate) that is corrected for multiple testing.

I filtered the results of the differential expression analysis using an adjusted P-value threshold of 0.05 and a log2 fold change threshold of 0.25. Also, I further filtered the significant hits in order to show only the genes that are known to be related to diabetes. From the resulting tables it is possible to see that there are 18 more up regulated genes and 13 more down regulated genes, than what we would expect by chance.

One way to correctly answer this question is to plot the log2 fold change of diabete and non diabete related genes to see if they follow a normal distribution. Then, I could have performed a 2-sample T-test to test if the mean of the distributions of the log 2 fold change between the two groups are different.

References

[1] Homo sapiens CUGBP Elav-like family member 4 (CELF4), transcript variant 65, mRNA, NCBI.
[2] Homo sapiens uncharacterized LOC105372068 (LOC105372068), long non-coding RNA, NCBI.
[3] Mattick JS, Rinn JL. Discovery and annotation of long noncoding RNAs. Nat Struct Mol Biol. 2015;

22(1):5-7.
[4] Exosome complex component RRP40, UniProt.
[5] P. Preker et Al. PROMoter uPstream Transcripts share characteristics with mRNAs and are produced upstream of all three major types of mammalian promoters. Nucleic Acids Res. 2011; 39(16):7179–7193.
[6] M. Lloret-Llinares et Al. Relationships between PROMPT and gene expression. RNA Biolv. 2016; 13(1).