Individual Homework BOHTA 2017

William Paul Bullock
ZHN775
Exam #49

13 June 2017

Preliminaries

All libraries I use in this assignment are stated below.

Please load these libraries

```
library(ggplot2)
library(tidyverse)
library(plyr)
library(reshape2)
library(data.table)
```

Part 1: 10 super-transcription factor clusters to rule them all?(total 11p)

Question 1: In the ENCODE project, a large number of labs have pooled resources to make many ChIP experiments for a large number of transcription factors (TFs) in different cell lines. Each such experiment will give a large number of ChIP peaks, corresponding to transcription factors binding DNA. ENCODE made a "master" bed track which shows ChIP peaks for all the transcription factors that they made ChIP experiments for. This can be found in the UCSC browser at Regulation->ENCODE regulation->Txn Factor ChIP. The below image (omitted in answer submission) shows an example of the track around the RXRA TSS. Each block is the ChIP peak of one TF.

As the picture (omitted in answer submission) show, there seem to be hotspots in the genome where many different TFs bind. At /home/bohta/HW4/part1/txnChIP.bed on the ricco.popgen.dk server there is a link to a bed file corresponding to the master track above, from the hg19 assembly.

A: Using BEDtools in Linux, merge the ChIP peaks that overlap over 1bp or more, and produce a BED file with these regions. How many merged regions are produced compared to how many peaks you started with? 2p

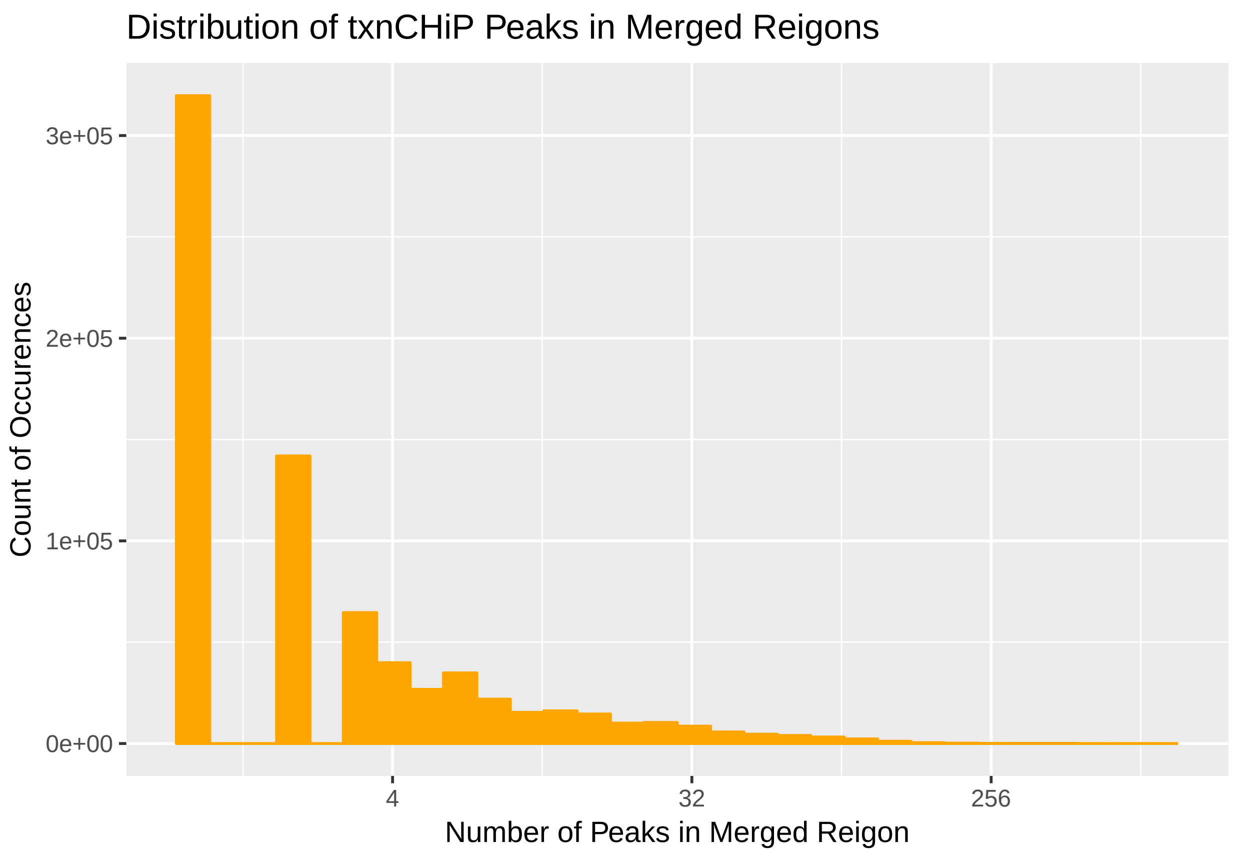
```
# First I sorted the tnxChIP bedfile...
sort -k1,1V -k2,2n txnChIP.bed > txnChIP_sorted.bed

#Then I merged the overlapping ChiP peaks and produced, as the 4th column;
#a count of the number of overlapping intervals in the chromosome that comprised
#that merged reigon.
bedtools merge -i txnChIP_sorted.bed -c 1 -o count > merged_txnChIP_sorted.bed

#Counting the number of merged reigons in comparison to the number of start peaks:
```

```
wc -l txnChIP.bed
#Output
4380444 txnChIP.bed

wc -l merged_txnChIP_sorted.bed
#Output
746610 merged_txnChIP_sorted.bed
```

Comment: As such we can see there are 746610 merged reigons out of 4380444 total peaks.

B: Using R, plot the distribution of ChIP peaks in each merged region using ggplot. Use log2 scaling for the x-axis, not logged values (in the example below, the first plot is a raw distribution, the other is with log2'd values and the third is with log2 scale). Briefly comment your plot (30 words max). 2p

```
#load the bed file into R
txnCHiP <- read.table("merged_txnChIP_sorted.bed", header = F)

#plotting in ggplot
ggplot(txnCHiP, aes(x = V4)) +
  #bin for a histogram
  geom_bar(stat = "bin", color = "orange", fill = "orange") +
  #Log2 Scaling for x
  scale_x_continuous(trans='log2') +
  xlab("Number of Peaks in Merged Reigon") + ylab("Count of Occurences") +
  ggtitle("Distribution of txnCHiP Peaks in Merged Reigons")
```



Comment: The majority of regions are not merged, (only 1 peak In the region). As the number of peaks in each region increases, the count of occurrences decreases (becomes less likely).

C: Using R only (no pasting or editing allowed in external programs), produce a new BED file that contains the 10 merged regions having the highest number of ChIP peaks. 1p

```
#sorting txnCHiP data frame by count column (descending).
txnCHiP_sorted <- txnCHiP[order(-txnCHiP$V4),]

#extracting top 10 rows
txnCHiP_best <- txnCHiP_sorted[1:10,]

#exporting as a BED file, tab delimited, no column headers or row numbers,
#strings output without "" notation.
write.table(txnCHiP_best, "txnCHiP_best.bed",
            sep="\t", col.names = F, row.names = F, quote = F)
```

D: Upload this into the UCSC browser. Look at each merged region and try to interpret it, also taking the peaks it contains into account. What do the merged regions typically overlap? Is there a particular factor that is responsible for the clusters? 4p

Amongst all 10 of the genes; the peaks correspond mostly with upstream regions of the gene, it is intuitive that promoter regions should lie here, so as to be able to alter gene expression down stream. Furthermore at many sites of the transcription factors (TFs) align almost into columns, suggesting that specific regions of the gene bind with promiscuity to a great number of TFs, rather than individual promoters for each TF being continuous expressed throughout the genes and themselves overlapping each other.

Alot of TFs appear in multiple genes, and many of them seem to appear together, (e.g. MAX and YY1), this is to be expected, as TFs are not all mutually exclusive, and may be linked in some way. For example; some may be functionally dependant on others. It could be worthwhile investigating this further to determine how they might interact with each other and affect gene expression together.

“POLR2A”, shown as a transcription factor on genome browser stands out especially, as the only TF that encompasses the entire length of all 10 genes. This is in actuality showing where RNA polymerase II can bind. As this is the main mediator or the transcription process, it is unsurprising it can bind to any region that any of the TFs can bind do, which in these examples comprises all of each gene.

A typical instance of POLR2A, along with the aforementioned distribution of TFs can be seen in the screen-capture shown below

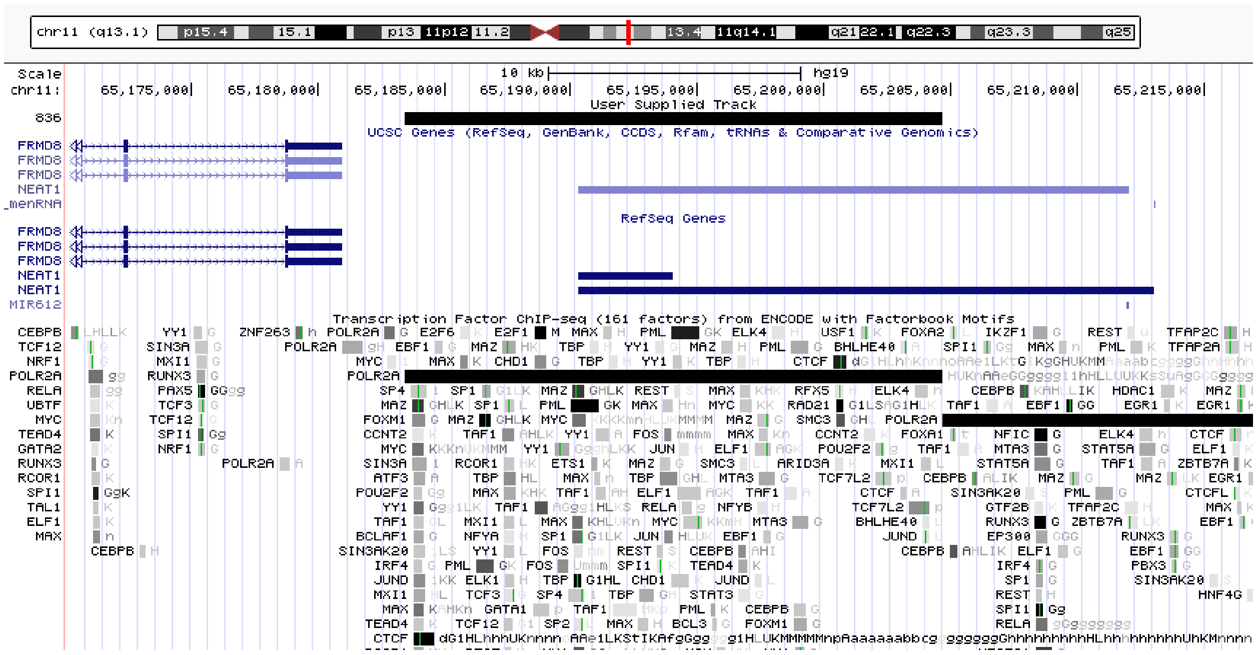


Figure 1: UCSC browser showing custom user track and ENCODE Regulation Txn Factor ChiP data, chr14 positon 69250548-69263605

The Dubious inclusion ofPOLR2A as a TF in the analysis will affect clustering by causing all the TFs to gain +1 to the number of overlaps, even those that would not have overlapped with anything otherwise. Furthermore POLR2A itself will always overlap with a huge amount of TFs and likely continually cause clusters containing it to have disproportionately high levels of overlapping and rsie to prominence in the analysis.

E: In hindsight, given the result in D, what could we have done to improve the analysis? 2p

It would be beneficial to repeat this analysis omitting POLR2A as a TF; so as to remove the aforementioned biases and mis-representation of clustering caused by the inclusion of this as a TF.

Potentailly removing those with only 1 overlap, or filtering for an arbitrary minimum number of overlaps could present more meaningful data.

Combining TFs into profiles, based on structural or functional similarities so as to determine if TFs with these similarities also prefer to bind in similar regions.

Part 2: Selection of tools – server part (total 18p)

Intro: The professor you are doing your master thesis for wants you to analyze an RNA-seq dataset. Furthermore, she is very keen on using some of these so-called pseudo-aligner tools that she has heard so much about – more specifically Salmon and Kallisto (https://www.ncbi.nlm.nih.gov/pubmed/28263959 and https://www.ncbi.nlm.nih.gov/pubmed/27043002 respectively).

For the final analysis you can naturally only use one tool, which means you have to select either Kallisto or Salmon. Since you are a budding data scientist, you will select the best tool based on performance on real data instead of flipping a coin. More specifically, you want to analyze 3 paired end RNA-seq libraries (biological replicates) using both tools and see which one performs better. There is no need to quality trim the sequences.

Part 2 is about doing the actual quantification with the two tools and the next question (part 3) is about doing the comparison (using the data we provide!).

Part 2 is going to be done exclusively on the Ricco server. Note that we provide you with real-sized dataset so some waiting time might occur.

To run Kallisto and Salmon you naturally need the data. In your own directory (/home//) use the “ln -s” function to make a symbolic link (shortcut) to all 6 fastq files located in the /home/bohta/HW4/part2/” folder. This is done instead of copying the file to your directory and saves a lot of space (please don’t copy – we will run out of hard disk space if you do!). Remember you can always get information about a command you don’t know using ‘man ’.

Note that the files with the “R1” and “R2” suffix correspond to each end of the paired end data (so the first sequence in each file is a pair, the second sequence in each file is a pair etc).

Question 2.1: Provide one line of code which will make symbolic links only to the 6 fastq files (hint you can write “man ln” to get the manual page on the server or look here: http://man7.org/linux/man-pages/man1/ln.1.html)1p

```
#run from inside destination folder,

# -s makes symbolic
# ? is a wildcard that denotes any single character
# . denotes the link should be made in current location.
ln -s ../../../bohta/HW4/part2/WT?_R?.fastq .
```

Next, you want to check how well the sequencing run went.

Question 2.2: use the “wc” function to calculate the number of reads in all fastq files and comment on the results using max 50 words. 2p

```
#grep captures all lines that begin with (denoted by ^) '+'.
#This is then pipelined to a wc which counts the lines.
#each read includes only one line that begins iwth the plus symbol.
#as such this gives us the number of reads for each sample.

grep '^+$' WT1_R1.fastq | wc -l
#Output
45626717
```

```
grep '^+$' WT1_R2.fastq | wc -l
#Output
45626717


grep '^+$' WT2_R1.fastq | wc -l
#Output
41428670

grep '^+$' WT2_R2.fastq | wc -l
#Output
41428670


grep '^+$' WT3_R1.fastq | wc -l
#Output
19326183

grep '^+$' WT3_R2.fastq | wc -l
#Output
19326183
```

Comment: The first two libraries are larger than the third, with the third being roughly half the size. Typically, R1 is in forward orientation and R2 is reverse-complement orientation. Here R1 & R2 all have the same number of reads.; trimming data achieves this by removing promoters, adaptors, and single reads.

Question 2.3:

Now you are ready to quantify the data. To save time and computational resources, you will in this exam just quantify the WT1 library. You start with Kallisto which can be run on the server with the following command: "nice /home/bohta/bin/kallisto quant". It is important that you include the "nice" part of the command as that will enable multiple users to use the server simultaneously! Note that you can open the Kallisto documentation by typing: "nice /home/bohta/bin/kallisto quant"

You want to: 1) Tell Kallisto what isoforms to quantify (as stored in a precompiled index) by specifying "–index /home/bohta/HW4/part2/kallistoIndex". 2) Tell Kallisto that it is a first stranded RNA-seq library by adding "–fr-stranded" to the command. 3) Tell Kallisto to use 6 cores for the quantification by adding "-t 6" to the command (Only use 6 so there are also cores for your classmates!). 4) Tell Kallisto to output a plain text document by adding "–plaintext" to the command. 5) Enable bias correction. 6) Output result to an appropriately named directory.

Assignment:

A) Report the command for running Kallisto on the WT1 RNA-seq data. 1p

B) Report the number of pseudo-aligned reads. 1p

C*) Report the estimated average fragment length. Based on this result, what is then the distance between the 3'end of an average read pair? Comment on the result using max 50 words. 4p

```
#A)
nice /home/bohta/bin/kallisto quant --index
/home/bohta/HW4/part2/kallistoIndex
--fr-stranded --bias -t 6  -o ./kallisto_ans
--plaintext WT1_R1.fastq WT1_R2.fastq

#Kallisto Output:
[quant] fragment length distribution will be estimated from the data
[index] k-mer length: 31
[index] number of targets: 143,335
[index] number of k-mers: 108,532,752
[index] number of equivalence classes: 465,217
[quant] running in paired-end mode
[quant] will process pair 1: WT1_R1.fastq
                             WT1_R2.fastq
[quant] finding pseudoalignments for the reads ... done
[quant] learning parameters for sequence specific bias
[quant] processed 45,626,717 reads, 23,720,001 reads pseudoaligned
[quant] estimated average fragment length: 178.432
[   em] quantifying the abundances ... done
[   em] the Expectation-Maximization algorithm ran for 1,699 rounds

#B)
#from the above output we can see
"23,720,001 reads were pseudoaligned"

#C)
# i determine that:
#Avg 3' distance = avg fragment length - (2 * avg read length.)

#sum of all fragment lengths
grep -F "TCONS" abundance.tsv |cut -f2 | paste -sd+ | bc
#Output
325570317

#number of lines
grep -F "TCONS" abundance.tsv |wc -l
#Output
143335

#sum of all fragment lengths / number of lines gives...
325570317 / 143335 = 2271.394404716
average transcript length = 2271.394404716

#From Kallisto Output:
"estimated average fragment length was 178.432"

#Now I Looked in the fasta file for read lengths...
#it's not specified but they seem to all be ~100nt in length.
#We can check this as below:

#This command prints the length of every 4th line starting from the 2nd
#(this is the read), the length output only returns unique values.
```

```
awk '{if(NR%4 == 2) print length($1)}' WT1_R1.fastq | uniq
#Output
98

awk '{if(NR%4 == 2) print length($1)}' WT1_R2.fastq | uniq
#Output
98
#this shows all reads are 98 nucleotides in length.

#I Completed the following calculations using a calculator...
# avg fragment length - (2 * avg read length.) = Avg 3' distance
178.432 - (98 * 2) = -17.568

#Overlap! but average distance is an absolute value so...
17.568
#is the average number of nucleotides distance between the two 3'prime'
#ends on a pair of reads.
```

Comment: This distance is the 'no-mans-land' (unmapped region) or overlap between the 2 reads mapped one each to the two fragment strands. It initially was a minus number, thus suggesting that the majority of each fragment has reads which overlap and are mapped.

Question 2.4

Next, use Salmon for the quantification which can be run on the server with the following command: "nice /home/bohta/bin/salmon quant". Note that you are quantifying reads (not alignments) and only need to consider "basic options" and that you can open the Salmon documentation by typing: "nice /home/bohta/bin/salmon quant -h"

You want to:

1) Quantify the same isoforms as you did with Kallisto by adding "–index /home/bohta/HW4/part2/salmonIndex" to the command.
2) Tell Salmon to use 6 cores for the quantification by adding "-p 6" (Only use 6 so there are also cores for your classmates!).
3) Tell Salmon to automatically detect library type by adding "–libType A".
4) Turn on the bias correction algorithms.
5) Output the result to an appropriate directory.

Assignment:

A*) Report the command for running Salmon on the WT1 RNA-seq data. 2p.

B) Report the most likely library type as identify by Salmon. 1p.

C) Report mapping rate. 1p.

```
#A)
nice /home/bohta/bin/salmon quant --index
/home/bohta/HW4/part2/salmonIndex -p 6 --libType A --gcBias --seqBias
-1 WT1_R1.fastq -2 WT1_R2.fastq -o ./salmon_ans

#I Recieved this warning: due to specifying --libType A:
[2017-06-14 19:59:56.547] [jointLog] [warning] NOTE: Read Lib
[( WT1_R1.fastq, WT1_R2.fastq )] :
Greater than 5% of the fragments disagreed with the provided library type;
check the file: ./salmon_ans/lib_format_counts.json for details

#B)
#When opening the lib_format_counts.json file (using nano or similar)
#one can see a few things regarding library type.

#Of interest here is the statement:
"expected_format": "ISF",
# which seems to suggest that salmon was expecting an ISF format library.

#furthermore, later in the file...
#when looking at the number of fragments found for each Library format;
#ISF scores the highest with:
"ISF": 79884864,

#furthermore opening /logs/salmon_quant.log (with nano or similar)...
#one can see stated:
Automatically detected most likely library type as ISF

#C)
#Opening /logs/salmon_quant.log with nano or similar)...
#one can see the mapping rate is specifed as:
Mapping rate = 60.9567%
```

Question 2.5:

Which tool aligned more reads? Comment on the result using max 50 words. 1p

Salmon reads 27,812,554 and Kallisto 23,720,001.

Potentially due to differences in methodology. Kallisto pseudo-aligns reads as overlapping K-mers, and indexes them to a hash table representing the transcriptome . Salmon compares unedited reads with reference transcripts, it's slower but more accurate. It seems likely that Salmon gives greater read depth for some transcripts.

Question 2.6*: The effective length estimate (estimated by both tools) incorporates the bias corrections performed by each tool respectively into the length of the isoform. This means you can calculate RPKM/FPKM values less biased by sequence specific effects (such as length and GC content). The file with the quantification is called 'abundance.tsv' and 'quant.sf' for Kallisto and Salmon respectively. (hint: you might need to look at the actual files previously used in this assignment).

A) Compare the estimated "effective length" of the isoform 'TCONS_00000020' from Kallisto and Salmon to each other and the reference length. This must be done either in R or with the command line tool "grep". 1p
B) *What could explain the difference in the effective length? Which estimate do you trust more? Answer using max 75 words. 3p

```
#bash command used for kallisto:
grep -F TCONS_00000020 abundance.tsv | cat
#Output
TCONS_00000020  4456    4830.89 0   0

#bash command used for Salmon:
grep -F TCONS_00000020 quant.sf | cat
#Output
TCONS_00000020  4456    3835.53 0   0
```

A) In each case the 2nd column is Reference length and 3rd column is the effective length. As such the effecitve length for this isoform is… for kallisto - 4830.89 for Salmon - 3835.5 whereas the reference length is the same for both and is 4456.

B) The effective length is a probabilistic representation of the expected length a sequence would have to generate the observed number of reads.

I trust Salmon more, as it is more concerned with accuracy, and has deeper levels of bias correction, such as GC content, which kallisto doesn't. Furthermore it's expected that effective sequence length is usually smaller than the real length (unlike kallisto) because of the untranslated regions of the sequence and GC content limitations.

Part 3: Selection of tools – analysis part (total 19p)

In the attached data in the part 3 folder in http://people.binf.ku.dk/albin/teaching/bohta2017/data.zip you will find results from Kallisto and Salmon runs for all three biological replicas in the file "part3.Rdata". This file (created with the save() function) can be loaded into R with the load() function and contains 3 R objects:
-A replicate count data.frame ("countDF"). This data.frame will be referred to as the "count data".

-An annotation matrix ("annotationDF"). This data.frame will be referred to as "annotation".

-A filtered RPKM replicate expression matrix that has been log10 transformed with a pseudo count of 1 ("logRpkmDF"). This data.frame with be referred to as logRpkm data/values.

Samples from Kallisto have the prefix "K__" and ones from Salmon the prefix "S__". Use these files for the rest of the part 3! Note that countDF and logRpkmDF data.frames contain different isoforms and should not be compared.

Question 3.1: Load the data into R. How many isoforms are quantified in the count data? 1p

```r
#loading file
load(file = "part3.Rdata")
#query number of rows in file
nrow(countDF)
```

```
## [1] 5787
```

There are 5787 isoforms

Question 3.2:

You naturally want to normalize the data – more specifically you want to calculate RPKM values from the count data using the length from the annotation data. Use vectorised analysis to calculate RPKM values for all samples. A vectorised approach means using R code that calculates the operation for the whole vector (aka row or columns) simultaneously instead of one at the time (this means without using loops, the apply family of functions (apply, sapply etc.) or similar functions).

Assignment:

- Report the R code for how to calculate RPKM values. 3p

```r
#makes a data frame of sum for each column in countDF
sumDF <- data.frame(colSums(countDF))
#Renames the column in sumDF
colnames(sumDF) <- "Total"
#Extracts the gene length for each isoform from annotationDF
#and attaches it as a new column to countDF
countDF$Length <- annotationDF[match(rownames(countDF), annotationDF$isoform),"length"]
#takes the rowname to be a column in the dataframe
#(necessary for melting)
countDF <- cbind(Isoform = rownames(countDF), countDF)
#reformats the database to wide format,
#keeping Length and isoform as id's
meltDF <- melt(countDF, id=c("Length", "Isoform"))
#adds the total transcript length as a new column
```

```
#to the melted data frame
meltDF$Total <- sumDF[match(meltDF$variable, rownames(sumDF)),"Total"]
#new datafram to accept RPKM values
meltRPKM_DF <- data.frame(meltDF$Isoform, meltDF$variable, meltDF$value)
#renaming columns in new dataframe
colnames(meltRPKM_DF) <-c("Isoform", "variable", "value")
#calculating RPKM as
#RPKM = count / (gene length/1000) * (transcript length/1000000)
meltRPKM_DF$value <- (meltDF$value / ((meltDF$Length/1000) * (meltDF$Total/1000000)))

#reformatting melted datafram into standard format
RPKM_DF <- dcast(setDT(meltRPKM_DF), Isoform~variable, value.var=c('value'))

#change first row to row names (uses tidyverse package)
#mostly for visual readability
RPKM_DF <- RPKM_DF %>% remove_rownames %>% column_to_rownames(var="Isoform")

#output first few rows
head(RPKM_DF)
```

```
##    K_WT1_Counts K_WT2_Counts K_WT3_Counts S_WT1_Counts S_WT2_Counts
## 1:    25.209247    28.097248    20.992555 1.895475e+01    33.231942
## 2:     4.622839     2.198117     3.485759 4.855156e-08     1.755663
## 3:   116.401887   152.102984   155.407072 9.022047e+01   125.337756
## 4:     4.124546     7.118653     2.135348 2.513001e+00     5.405496
## 5:    13.374675    16.348447    15.870074 2.349930e+00     1.973684
## 6:   471.402701   618.990604   549.925260 3.773847e+02   529.216320
##    S_WT3_Counts
## 1:    14.371779
## 2:     3.713318
## 3:   127.355821
## 4:     1.647925
## 5:     3.720151
## 6:   461.352183
```

Question 3.3: Make a one-liner (without the use of ";") that outputs the mean RPKM value of each sample. The restrictions from the previous question no longer apply. 2p

```
#applies mean function over all columns of database from Q3.2
#and prints to outputinstead of assigning to vector
apply(RPKM_DF, 2, mean)
```

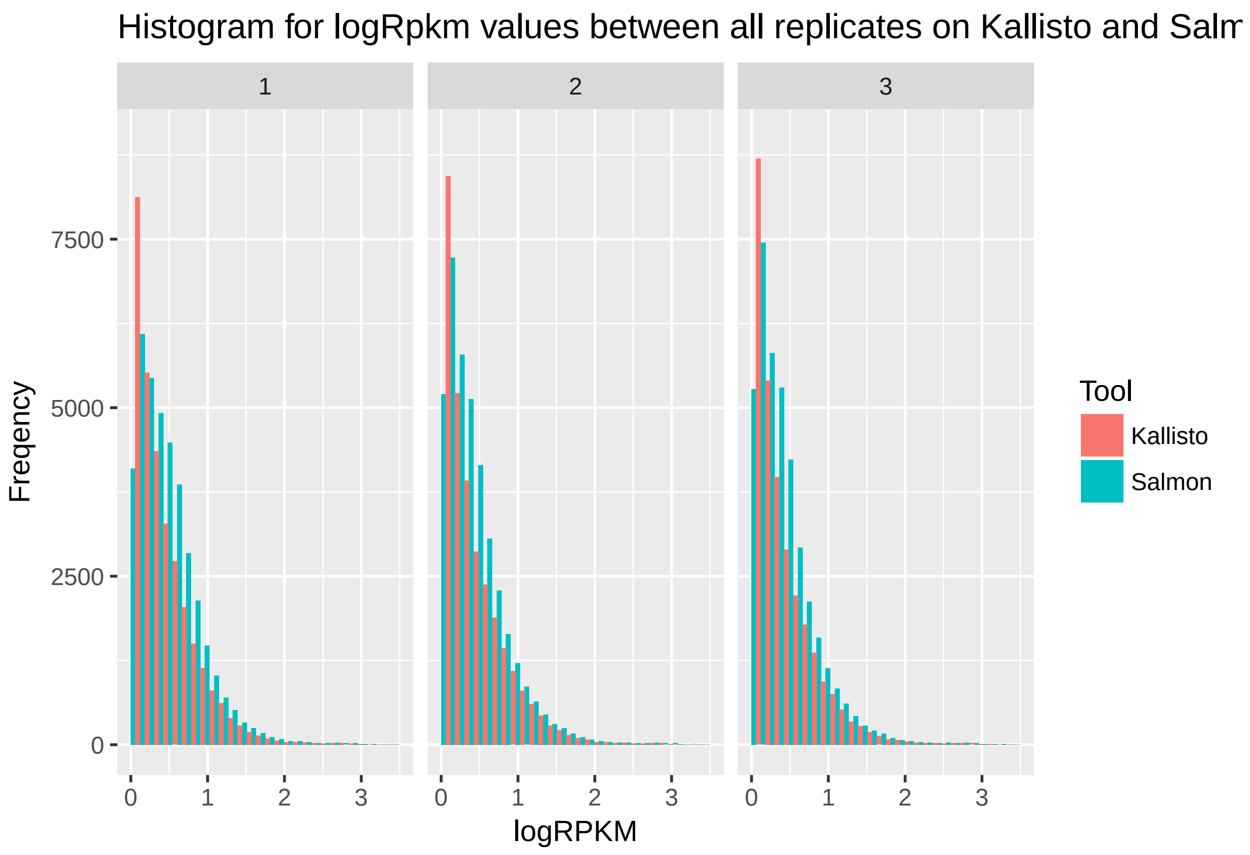
```
## K_WT1_Counts K_WT2_Counts K_WT3_Counts S_WT1_Counts S_WT2_Counts
##     122.3387     129.5154     121.6085     157.9253     185.6539
## S_WT3_Counts
##     186.2984
```

Now that you have the normalized data: time to make some summary plots

Question 3.4: Make histograms of the distribution of the logRpkm expression values. For this plot use colour to indicate the tool and use ggplot2 facets to make subplots that show each replicate (meaning you have 3 subplots in a larger plot). Provide all the R code necessary to produce the plots and comment on the distributions with max 50 words. 3p

```r
#Creates copy of logRpkmDF dataframe
log_copy  <- logRpkmDF
#takes the rowname to be a column in the dataframe
#(necessary for melting)
log_copy <- cbind(Isoform = rownames(log_copy), log_copy)
#reformats the database to wide format,
#keeping isoform as id's
log_melt <- melt(log_copy, "Isoform")
#removes unnecessary column of data
log_melt <- log_melt[,-1]
#creates a copy of the database
log_trim <- log_melt
#Creates a new column called "set" and assigns all fields to 1.
log_trim <- cbind(set = 1, log_melt)
#removes another uneeded column
log_trim <- log_trim[,-3]
#creates a dataframe of unique entries from log_trim,
#(also forces all unspecifed values to not be factors)
log_u <- data.frame(unique(log_trim), stringsAsFactors = F)
#to the unique dataframe, adds a column called "tool",
#all values are 0
log_u <- cbind(tool = 0, log_u)

#assigns tool to be "Kallisto"" or "salmon" for appropriate entries in #the unique data frame
log_u[1:3,1] = as.character("Kallisto")
log_u[4:6,1] = as.character("Salmon")
#assigns set to be 2/3 for the 2nd and 3rdreplicates for a appropriate
#entries in the unique data frame.
log_u[c(2,5),2] = 2
log_u[c(3,6),2] = 3

#Assigns all values in the first melted data frame an appropriate
#tool and set varaible
#uses match() to lookup the relevant entry in teh unique dataframe
log_melt$Tool<- log_u[match(log_melt$variable, log_u$variable),"tool"]
log_melt$replicate <- log_u[match(log_melt$variable, log_u$variable),"set"]

#Plots the melted data frame in ggplot
ggplot(data=log_melt, aes(x=value, fill=Tool)) +
  #subplot by replicate
  facet_grid(. ~ replicate) +
  labs(title="Histogram for logRpkm values between all replicates on Kallisto and Salmon") +
  labs(x="logRPKM", y="Freqency")+
  #truncate x-axis for readability
  xlim(0,3.5)+
  #alternate bars by tool
```

```
geom_histogram(position="dodge")
```



Comment: For each replicate the distribution is very similar; it seems Kallisto favours lower logRPKM values (0-0.5) , while Salmon is slightly more 'right shifted', and seems to have a greater proportion of its frequency 0.5+ than Kallisto does. The significance of this difference is questionable, and should statistically quantified.

Question 3.5: For each tool use logRpkm to calculate all pairwise replicate Pearson correlations of the replicate expression values and report the numbers in a table (one table per tool). 2p

```r
#subsetting data based on tool used
logRpkmDF_K <- logRpkmDF[1:3]
logRpkmDF_S <- logRpkmDF[4:6]

#Pearsons r on Kallisto data
K_pearson <-cor(logRpkmDF_K, use = "pairwise.complete.obs", method = "pearson")
K_pearson
```

```
##             K_WT1_RPKM K_WT2_RPKM K_WT3_RPKM
## K_WT1_RPKM  1.0000000  0.9458276  0.9465259
## K_WT2_RPKM  0.9458276  1.0000000  0.9637109
## K_WT3_RPKM  0.9465259  0.9637109  1.0000000
```

```r
#Pearsons r on Salmon data
S_pearson<-cor(logRpkmDF_S, use = "pairwise.complete.obs", method = "pearson")
S_pearson
```

```
##             S_WT1_RPKM S_WT2_RPKM S_WT3_RPKM
## S_WT1_RPKM  1.0000000  0.9365968  0.9402787
## S_WT2_RPKM  0.9365968  1.0000000  0.9614738
## S_WT3_RPKM  0.9402787  0.9614738  1.0000000
```

Comment: In general terms, correlation shows any statistical relatedness (not necessarily causality) between bivariate data (or two random variables).

Here Pearson's correlation coefficient is a numerical representation from -1 to +1 of linear correlation. In each instance of the pairwise testing, we can see, on the diagonal, each replicate has positive correlation of 1, this is to be expected, as the two sets of values being compared are of course the same.

In all other instances of comparison between replicates, for both tools, give a very strong positive correlation. This suggests that if one observes high or low expression values for an isoform in one replicate, you would expect similar expression levels for that isoform in other replicates. (If the correlation were 0, no inference could be made (no correlation), if it were toward -1 , a negative correlation, then low levels in one replicate would suggest high level in others, or vice-versa.)

This difference may be by the differing methods used by each tool; in Kallisto this will occur depending upon how the tool decided to create fragments and read k_mers in each iteration. In Salmon this variance arises based on how it forms chains of maximal exact matches (MEMs) and super maximal exact matches (SMEMs) in each iteration.

Question 3.6: One of the best ways of evaluating replicate agreement is to calculate the cross replicate variance (often abbreviated CV) so of course you want to do that.

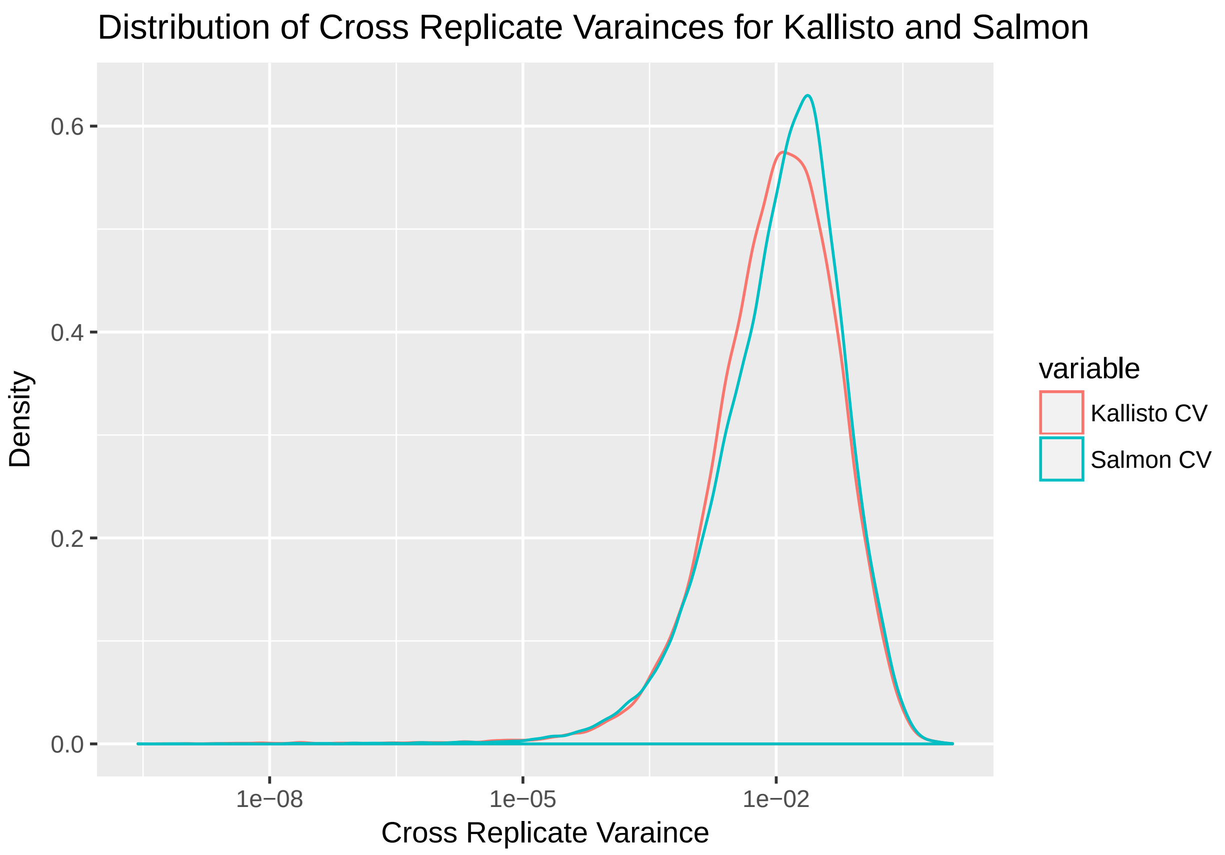
A CV value can be calculated for isoform i as follows: cvi = variance(xi) / average(xi), where xi is a vector with the replicate expression values for isoform i.

A) Construct a function which calculates the CV and use the apply() function to calculate the CV based on logRpkm values for Kallisto and Salmon (separately). 1p

B) Plot the distribution of CV values as density lines (in one single plot) using color to indicate the tool and log transform the x-axis (using log10). 1p

C) Comment on the CV plots using max 75 words. 2p

```r
#A)

#funciton to calculate CV as given
cv <- function (x) {
  return(var(x)/mean(x))
}

#Apply function on previously subsetted data frames (from Q3.5)
#(separating by tool) and store as new data frames...

#Kallisto
K_cv <- apply(logRpkmDF_K, 1, cv)
K_cv <- data.frame(K_cv)
#Salmon
S_cv <- apply(logRpkmDF_S, 1, cv)
S_cv <- data.frame(S_cv)

#B)
#combine the above into one datframe for the CVs for both tools
all_cv <- merge(K_cv, S_cv, by=0, all=TRUE)
#rename columns for readability
colnames(all_cv) <-c("Isoform", "Kallisto CV", "Salmon CV")
#melt the merged dataframe to wide format, keeping isoform as id.
cv_melt <- melt(all_cv, "Isoform")
#ggplot the melted dataframe
qplot(data = cv_melt, x = value, geom = "density", colour=variable) +
  #transform x axis
  scale_x_continuous(trans='log10')+
  xlab("Cross Replicate Varaince") + ylab("Density") +
  ggtitle("Distribution of Cross Replicate Varainces for Kallisto and Salmon")
```



Comment: Both tools follow the same distribution, and are very similar it's uncertain if they are significantly different. It seems that Salmon, has greater cross replicate variance slightly more frequently.

This suggests that Salmon is lightly less accurate over its replicas. However, because the CV is weighted by the mean, we would have to ascertain if these means are significantly different before declaring this.

Question 3.7*: Based on all the results you have collected here (all of part 2 and all of part 3), discuss in max 100 words which tool would you choose to continue with if you wanted to make a differential expression analysis. 4p

First, some statistics tests.

Statistics motivation:

In Q3.3 we can see Salmon has greater RPKM than Kallisto, and in Q3.6, the CV distributions differ slightly but it is not clear if either are significant.

To ascertain this I performed two paired Wilcox tests, one for RPKM values (across all replicates), Kallisto vs Salmon and one for CV values, Kallisto vs Salmon.

I used a wilcox test to validate if the mean values in both cases were significantly different, n0 being that the data sets do indeed, have significantly different means.

I used this instead of a t-test as it cannot be assumed that the data follows a normal distribution. This is a paired test, because the two tools share the same origin data, they are two different "samples" of this pool.

```
#Comparing RPKM means from Q3.3
#Subsetting based on tool
RPKM_DF_K <- RPKM_DF[,1:3]
RPKM_DF_S <- RPKM_DF[,4:6]
```

```r
#Calculating the mean for each tool and
#concatenating into a single datframe
RPKM_MEANS <- data.frame(apply(RPKM_DF_K[,1:3], 1, mean), apply(RPKM_DF_S[,1:3], 1, mean))
#renaming columns for readability
colnames(RPKM_MEANS) <- c("Kallisto", "Salmon")
#wilcox test
wilcox.test(RPKM_MEANS$Kallisto, RPKM_MEANS$Salmon, paired = T)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  RPKM_MEANS$Kallisto and RPKM_MEANS$Salmon
## V = 10224000, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

```r
#Comparing CV distributions from Q3.6
#wilcox test on already subsetted CVs by tool.
wilcox.test(K_cv$K_cv, S_cv$S_cv, paired = T)
```

```
##
##  Wilcoxon signed rank test with continuity correction
##
## data:  K_cv$K_cv and S_cv$S_cv
## V = 321950000, p-value < 2.2e-16
## alternative hypothesis: true location shift is not equal to 0
```

Statistics interpretation:

In both cases the P-value was less than 2.2e-16. On a significace level of 95%, this is highly significant, and for both the RPKM value comparison, and the CV value comparison. we can reject the null hypothesis. The conclusion is that there is a significant difference in the attained RPKM values and CV between Kallisto and Salmon.

Comment (this is 100 words):

I would use Salmon for a differential expression analysis (DE).

Although it has a greater spread across its replicates, (Q.3.6), it has a higher number of reads than Kalisto, and significantly higher RPKM values (Q3.3 & Q3.4) this suggests to me Salmon is capable of giving a greater read depth, and the loss of accuracy is due to an increase in sensitivity. With multiple replicates, Salmon will give more robust results closer to the true value.

Salmon's more direct method and superior bias calculation (Q2.5 & Q2.6), along with this increased read depth will give more accurate expression values.

Part 4: Exploratory Data Analysis (15p total)

Two postdocs in your lab, Bob and Alice, have been collaborating on an experiment. Using your labs favorite cell line, they have generated 25 knockdowns of two transcription factors along with 25 control samples. For each knock down sample, they have measured the knock down efficiency using qPCR and quantified expression of 10000 genes using RNA-Seq.

Last week the two postdocs got into an argument: Bob is accusing Alice of being sloppy in the lab and thereby ruining the experiment. Upon hearing these allegations, Alice responded by accusing Bob of sneezing into one of the samples.

Your professor, who knows you have been attending a bioinformatics course, has charged you with analyzing the data to solve the dispute between Bob and Alice and determine whether the dataset is ruined.

You have been supplied with an expression matrix containing normalized and log-transformed expression values, as well as the study design describing the content of each sample.

Question 0: Before you start, set your seed by set.seed to 2017 and load all packages you need for your analysis (0p – but you will get a minus point if you miss this).

```
#setting seed as directed
set.seed(2017)
```

Question 1: Read both the expression matrix ("ExpressionMatrix.tab") and study design ("StudyDesign.tab") into R. These are both found in http://people.binf.ku.dk/albin/teaching/bohta2017/data.zip, part 4. Report the number of samples and the number of genes. 1p

```
#reading in data sets
exp <- read.table("ExpressionMatrix.tab", header = TRUE)

St_de <- read.table("StudyDesign.tab", header = TRUE)

#number of samples
ncol(exp)
```

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

```
#number of genes
nrow(exp)
```

```
## [1] 10000
```

Question 2: Perform PCA on the samples and report the amount of variance contained in the first 5 principle components. Note: You should scale the expression values before the PCA. 2p

```
#call principle componenet analysis on the transpose
#of the given expression values
#scaling to have unit varaince as directed.
pca_exp <- prcomp(t(exp), scale = TRUE)

#call object summary and save as table
```

```r
pca_exp_abst <- summary(pca_exp)

#output the cumulative proportion of the variance by PC
pca_exp_abst$importance[3,]
```

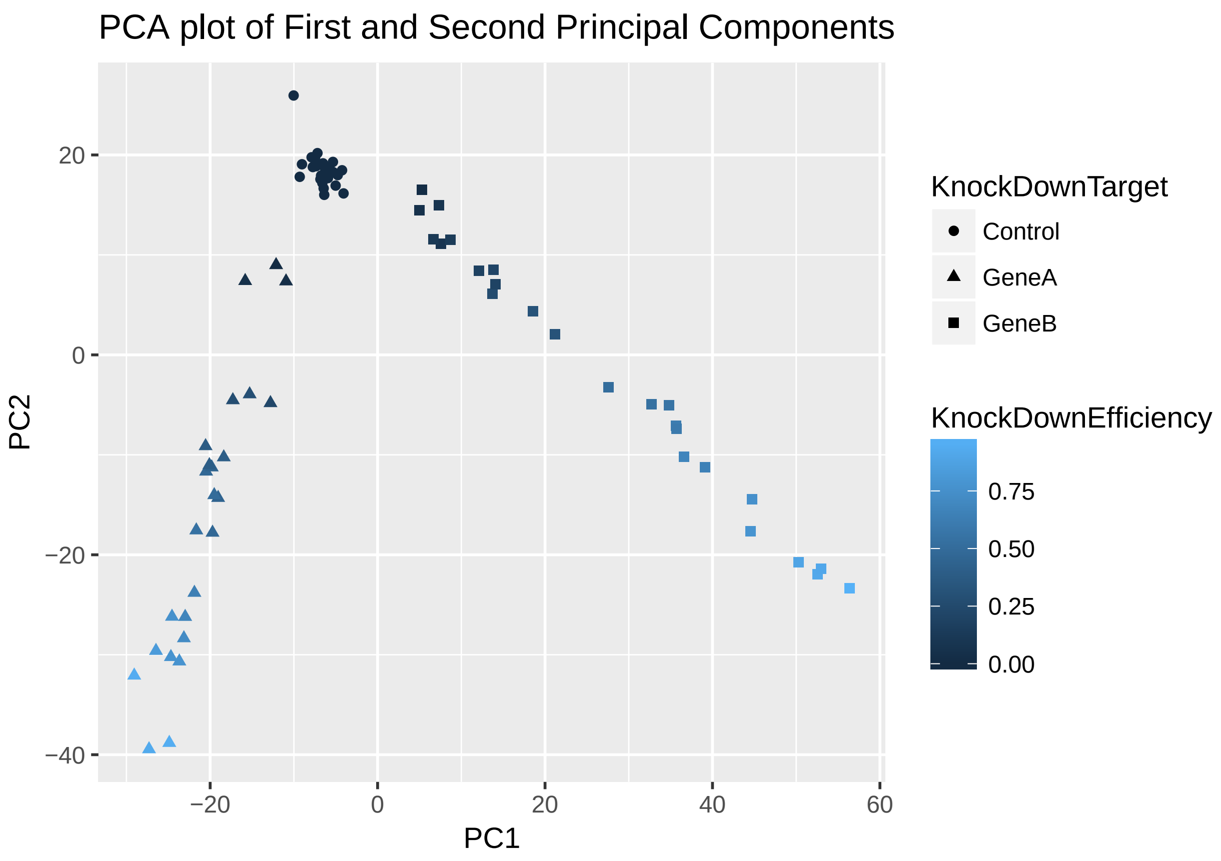
```
##     PC1     PC2     PC3     PC4     PC5     PC6     PC7     PC8     PC9
## 0.05092 0.08373 0.10470 0.12347 0.13872 0.15361 0.16802 0.18231 0.19656
##    PC10    PC11    PC12    PC13    PC14    PC15    PC16    PC17    PC18
## 0.21072 0.22476 0.23870 0.25259 0.26644 0.28016 0.29380 0.30735 0.32087
##    PC19    PC20    PC21    PC22    PC23    PC24    PC25    PC26    PC27
## 0.33432 0.34773 0.36110 0.37442 0.38770 0.40096 0.41413 0.42723 0.44026
##    PC28    PC29    PC30    PC31    PC32    PC33    PC34    PC35    PC36
## 0.45327 0.46623 0.47914 0.49199 0.50480 0.51759 0.53032 0.54297 0.55556
##    PC37    PC38    PC39    PC40    PC41    PC42    PC43    PC44    PC45
## 0.56811 0.58063 0.59312 0.60558 0.61800 0.63037 0.64271 0.65499 0.66724
##    PC46    PC47    PC48    PC49    PC50    PC51    PC52    PC53    PC54
## 0.67949 0.69164 0.70376 0.71579 0.72777 0.73971 0.75163 0.76351 0.77528
##    PC55    PC56    PC57    PC58    PC59    PC60    PC61    PC62    PC63
## 0.78703 0.79869 0.81031 0.82188 0.83343 0.84494 0.85642 0.86787 0.87919
##    PC64    PC65    PC66    PC67    PC68    PC69    PC70    PC71    PC72
## 0.89049 0.90172 0.91292 0.92401 0.93502 0.94602 0.95700 0.96791 0.97877
##    PC73    PC74    PC75
## 0.98945 1.00000 1.00000
```

Comment:

From this, it can be seen that 0.13872 of the variance (13.872%) is captured by the first 5 principle componenets. This can be seen from the Cumulative Proportion value, under PC5.

Question 3: Inspect the first 2 principle components: Make a plot of PC1 vs PC2, where knock down efficiency is indicated by color and knock down target is indicated by shape. Comment on the plot using a maximum of 75 words. 2p

```r
#form the pca of expresison values and the given "studydesign" dataset
#into one dataframe
plot_pca_exp <- data.frame(pca_exp$x, St_de)
#ggplot this new dataframe, colouring by knock-down efficiency
#andshaping points based on knock-down target
qplot(PC1, PC2, color=KnockDownEfficiency, shape=KnockDownTarget, data=plot_pca_exp)+
  ggtitle("PCA plot of First and Second Principal Components")
```



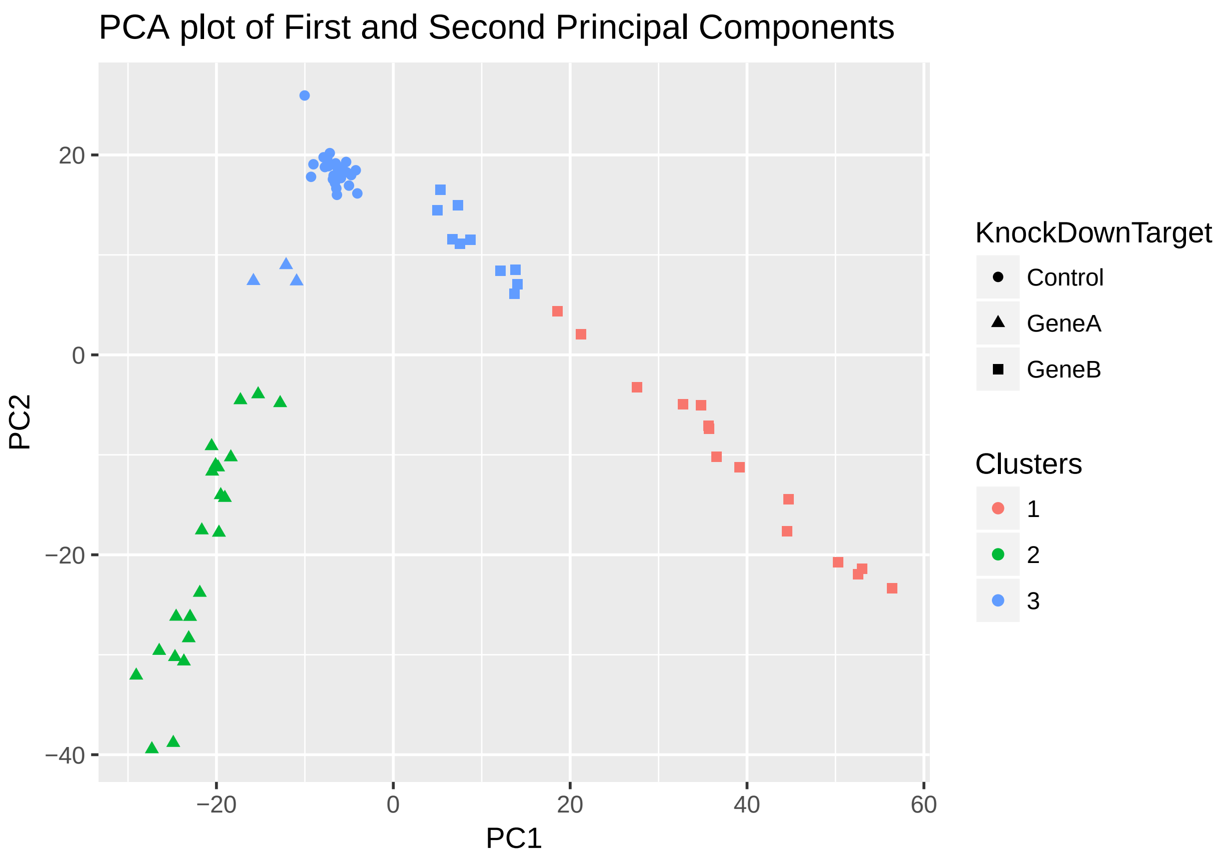
Comment:

It seems PCA1 shows the variance inherent between the knock down targets while the variance captured in PCA2 is synonymous with knock down efficiency.

As such, in this data set, its seems the greatest 2 individual contributors to the variance of the data are the knock down target and the knock down efficiency.

Question 4: Perform a K-means clustering of the data, using k=3 and 10 random starting points. Visualize the clustering by making a plot of PC1 vs PC2, where the clusters are indicated by color. Briefly comment on how the clustering corresponds to the known knockdown targets, using a maximum of 75 words. 3p

```
#Perform k-means clustering on the transpose of the expression data
#number of clusters to form = 3, using 10 random starts
exp_clst <- kmeans(t(exp), 3, nstart=10)
#subset clusters as a factor(for better plot)
Clusters  <- factor(exp_clst$cluster)

#ggplot PCA, same as before, but using clusters subset to colour
qplot(PC1, PC2, color=Clusters, shape=KnockDownTarget, data=plot_pca_exp)+
    ggtitle("PCA plot of First and Second Principal Components")
```



Comment:

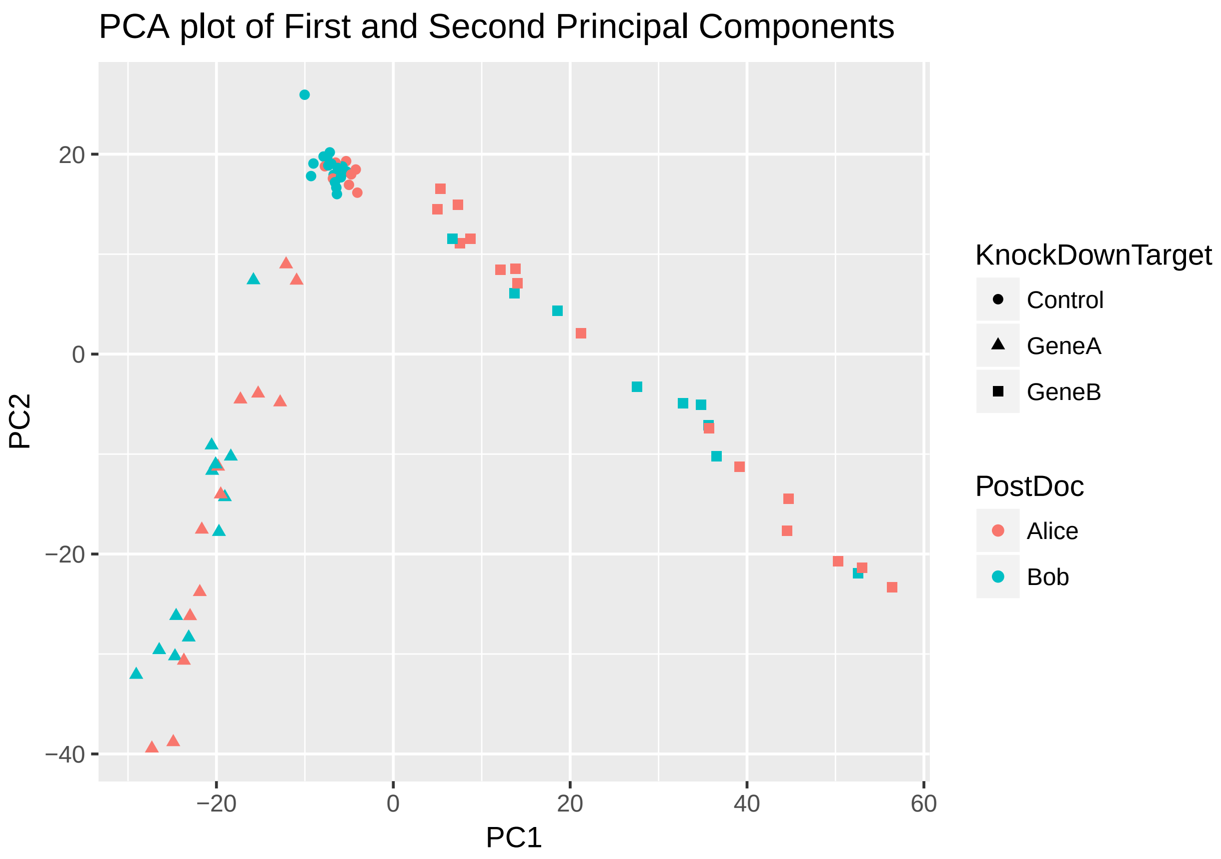
When k=3 the clusters assigned match primarily similarity to the knock-down-targets, this is unsurprising, because as aforementioned they carry the majority of the variance in the data.

The difference is that some of the gene A & B samples are part of the 3rd cluster, which otherwise consists entirely of controls.

This because their knock-down-efficiencies is closer to that of the controls than others in their target, and this is also being taken into account.

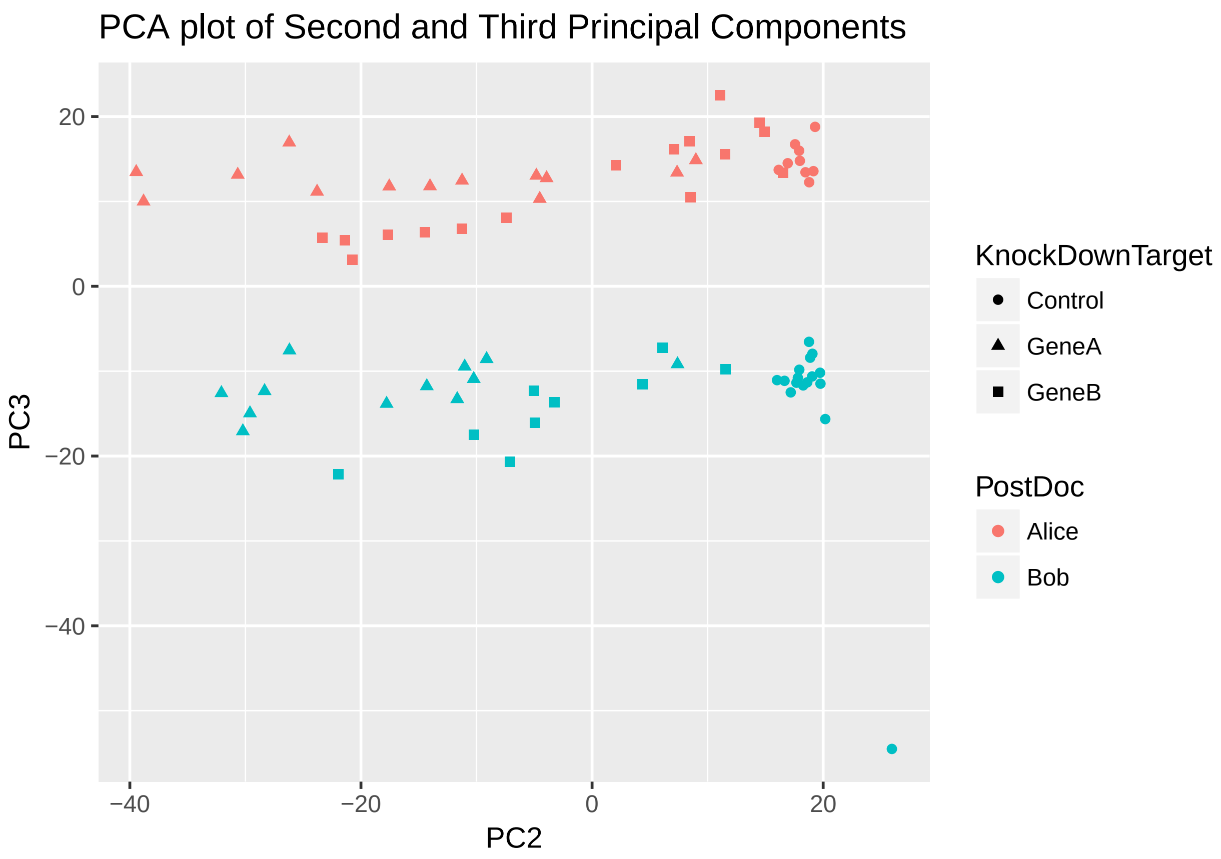
Question 5*: Using a maximum number of 3 plots, investigate whether there is any truth to the postdoc allegations: Can you see a difference in samples prepared by Alice and Bob? Is there any indication Bob has ruined a sample? Discuss you results using a maximum of 75 words. 4p

```r
# Plot to show similarities
# Colouring by postdoc
qplot(PC1, PC2, color=PostDoc, shape=KnockDownTarget, data=plot_pca_exp)+
 ggtitle("PCA plot of First and Second Principal Components")
```



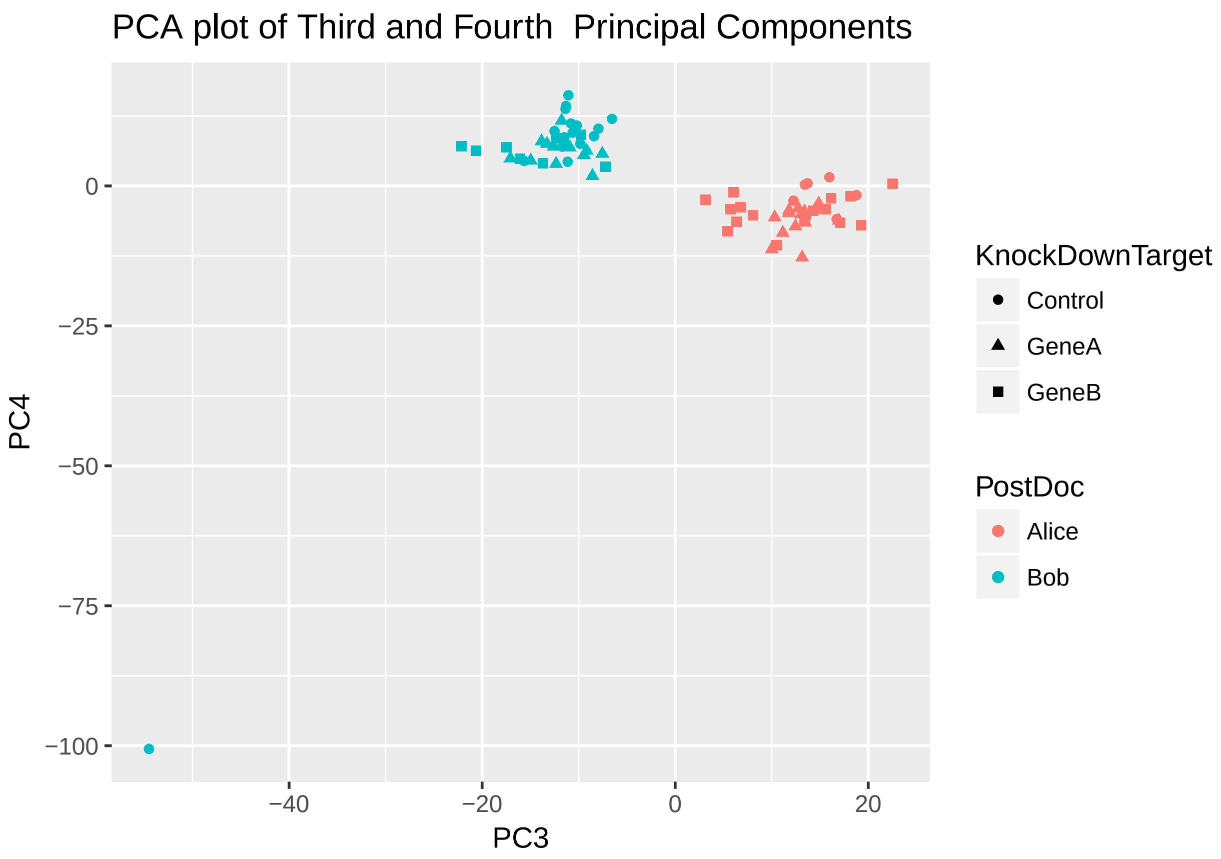
This shows that both post-docs follow the same general trend, with regards to the first 2 PCs.

```r
# plot to show 3rd most prominent variance is in postdocs
qplot(PC2, PC3, color=PostDoc, shape=KnockDownTarget, data=plot_pca_exp)+
  ggtitle("PCA plot of Second and Third Principal Components")
```



Here we can see PC2 (variance in knock-down-expression), alongside PC3, which seems to capture variance existing entirely between the post-docs. There is one outlier, a control by Bob in the bottom-right, the only sample >20 on PC2, it's likely this is the ruined sample.

```r
# plot to show The ruined sample & PC4
qplot(PC3, PC4, color=PostDoc, shape=KnockDownTarget, data=plot_pca_exp)+
  ggtitle("PCA plot of Third and Fourth  Principal Components")
```



This shows PC4 also carries variance based on post-docs, mostly due to the outlier.

Question 6: Based on all you observations in Question 1-5, discuss whether the experiment is still useful, or whether the postdocs have ruined it. Use a maximum of 100 words. 3p

The post-docs have introduced a lot of variance, so much so it is the third and fourth greatest contributor to variance in the data. To quantify the significance of this, one should use a statistical test.

How useful this data is depends on what it will be used for, it would be better to repeat the experiment, but the introduced difference could be counteracted by removing any systematic bias introduced by Alice or omitting those points and/or the single outlier caused by Bob entirely. This would perhaps make the data more representative of the true values.

Part 5: Differential Expression (DE) (total 18p)

Another PhD student in your lab, named Caroline, has already performed a DE analysis on a subset of the above dataset using the popular R-package limma (limma performs tests for DE using a modified version of gene-wise linear regressions). She has asked for you help in further analyzing the data. She is particularly interested in whether the two knockdowns affect the same genes.

The DE analysis file contains mean expression across all samples, and for each transcription factor knockdown (geneA and geneB) log fold change values (logFCs), p-values and p-values corrected for multiple testing using Benjamini-Hochberg.

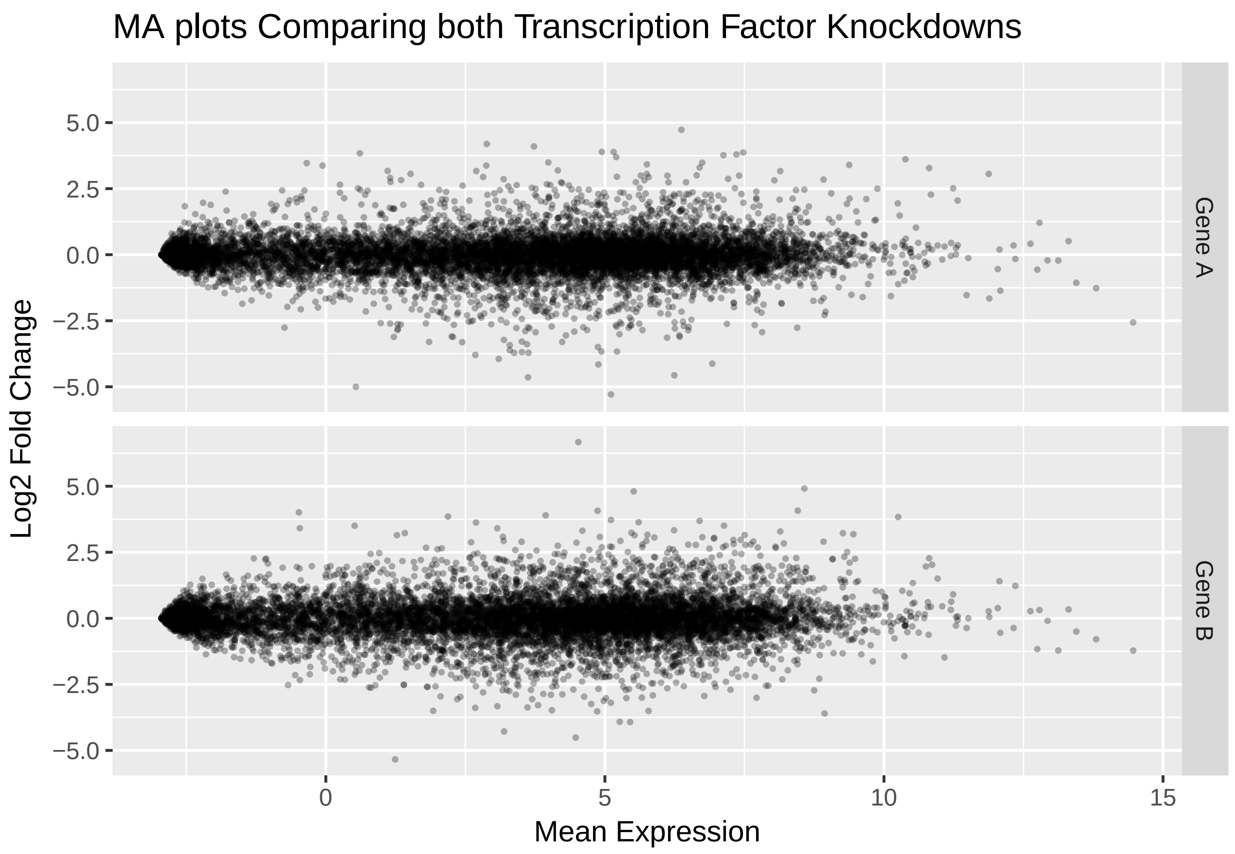
Question 1: Read the DE analysis ("DifferentialExpression.tab") results into R, and show the first few lines of the file. This can be found in http://people.binf.ku.dk/albin/teaching/bohta2017/data.zip, part 5. 1p

```
DE <- read.table("DifferentialExpression.tab", header = TRUE)
head(DE)
```

```
##    gene meanExpression      A.logFC    A.pvalue      A.FDR     B.logFC
## 1 Gene1     -2.5530584  0.611356920 0.062419401 0.36059735 -0.06796082
## 2 Gene2     10.8615216  0.195099082 0.705162135 0.94589257  2.02880013
## 3 Gene3      0.4656058 -0.442454825 0.323734729 0.75699168 -0.49773125
## 4 Gene4      5.3294517 -1.345297979 0.002686708 0.03827219  0.12547548
## 5 Gene5     -1.4903348 -0.047189904 0.910177555 0.99002931 -0.33892973
## 6 Gene6     -2.7898458 -0.009286818 0.971978672 0.99649239  0.45123665
##       B.pvalue      B.FDR
## 1 8.250010e-01 0.94580106
## 2 7.077611e-05 0.00132045
## 3 2.411661e-01 0.59217398
## 4 7.613912e-01 0.92022144
## 5 3.927040e-01 0.72925531
## 6 7.383572e-02 0.29522481
```

Question 2: As part of a single plot, produce two MA-plots (one for each knockdown). To mitigate overplotting, points on the plot should be transparent. 2p

```
#Subsetting the desired parts of DE dataframe
DE_trim <- data.frame(DE[,2:3], DE[6])
#renaming for better plot labels
colnames(DE_trim) <- c("meanExpression", "Gene A", "Gene B")
#melting with mean expression as id
DE_melt <- melt(DE_trim, "meanExpression")

#ggplot of mean expression
ggplot(data=DE_melt, aes(x=meanExpression, y=value)) +
  #adjust alpha and size to mitigate overplotting
  geom_point(alpha=0.3, size=0.5) +
  #subplot by Gene
  facet_grid(variable ~ .)+
  #labels
  xlab("Mean Expression") +
  ylab("Log2 Fold Change")+
  ggtitle("MA plots Comparing both Transcription Factor Knockdowns")
```



Question 3: Explain how an MA-plot can be used to investigate whether expression data has been properly normalized (max 75 words). 2p

Points are equally distributed around a Log2FoldChange of Zero (y=0), and as such the mean across probes should be close to zero. If the expresison data was not normalized, you would see points favouring one side of this horizontal line, and the mean would be further from zero.

If a line of best fit was introduced, in normalized data it should be close to parrallel with the X-axis, and intercept at Log2FoldChange (Y) = 0.

Question 4: Report the number of upregulated and downregulated genes that are significantly DE after correction for multiple testing (at alpha=0.05). Which knockdown target has the highest total number of DE genes? 2p

```
#Set alpha
Alpha <- 0.05

#Subset DE dataframe by knock-down gene
GeneA <- data.frame(DE[1], DE[3], DE[5])
GeneB <- data.frame(DE[1], DE[6], DE[8])

#Further subset by only significantly DE genes
```

```r
GeneA_DE <- subset(GeneA, GeneA[ , 3] < Alpha)
GeneB_DE <- subset(GeneB, GeneB[ , 3] < Alpha)

#Number of significantly DE genes from knock-down A
nrow(GeneA_DE)
```

```
## [1] 754
```

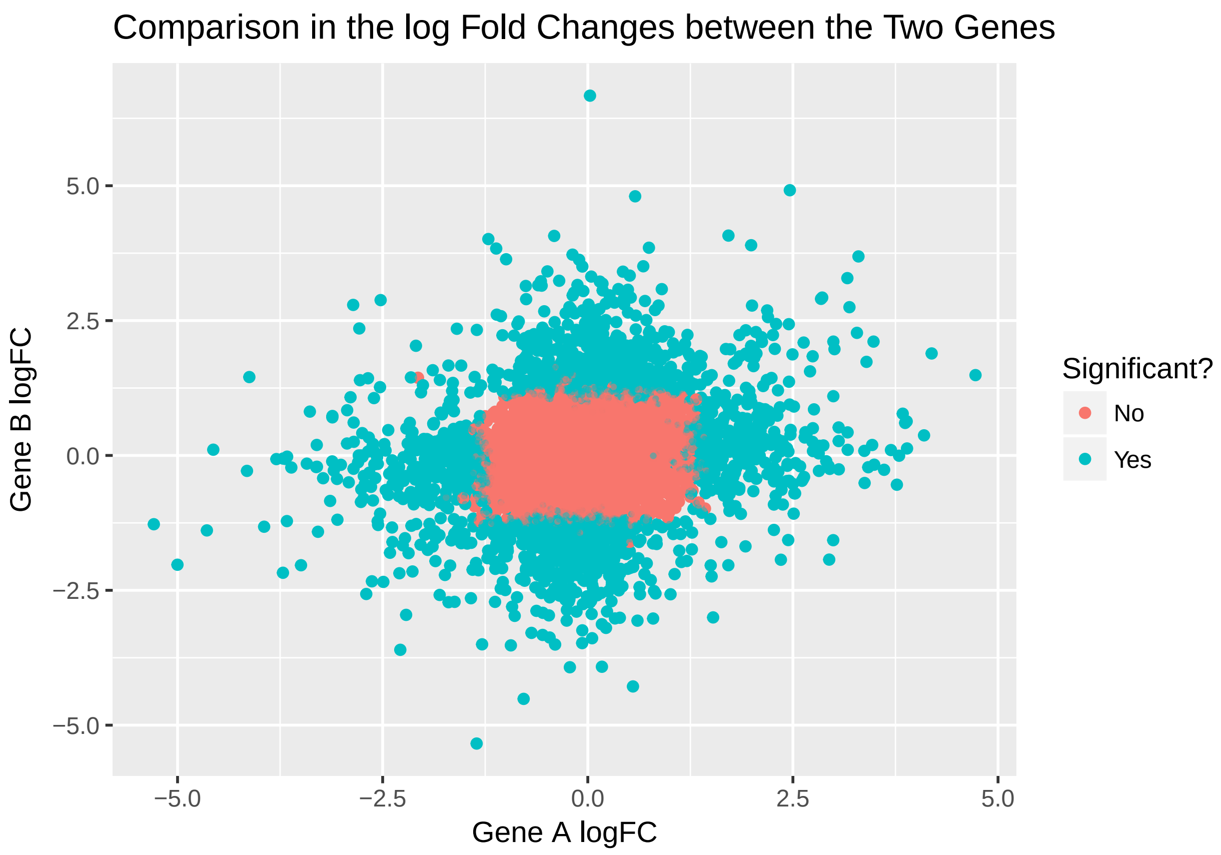
```r
#Number of significantly DE genes from knock-down B
nrow(GeneB_DE)
```

```
## [1] 1304
```

Gene B has the highest total number of significantly differentially expressed genes.

Question 5: Plot the two sets of logFCs (log2 fold changes) against each other. Color points based on whether they are significantly DE after multiple testing correction (at alpha=0.05) in either one or both of the knockdowns. 3p

```r
#copying DE data frame adn adding a column...
#iterating if/else statement over both FDR columns...
#if value in either is < alpha (0.05)...
#set value in new column to 1, else 0.
DE_copy <- cbind(significant = as.factor(ifelse(DE[,5]< Alpha | DE[,8]<Alpha, 1, 0)), DE)

#ggplot of logFC Gene A vs B
#colouring by significance column as generated above.
qplot(DE_copy$A.logFC, DE_copy$B.logFC, colour=DE_copy$significant)+
  #adjust alpha and size to mitigate overplotting
  geom_point(alpha=0.5, size=0.5) +
  xlab("Gene A logFC") +
  ylab("Gene B logFC") +
  ggtitle("Comparison in the log Fold Changes between the Two Genes")+
  #adding legend for colour
  scale_colour_discrete(name= "Significant?", labels=c("No", "Yes"))
```



Question 6: Calculate the Pearson correlation between the logFCs of the two knockdowns, and test whether this correlation is significant. 2p

```r
#Pearson test for association between paired samples,
#with significance test
cor.test(DE$A.logFC, DE$B.logFC, method = "pearson")
```

```
##
##  Pearson's product-moment correlation
##
## data:  DE$A.logFC and DE$B.logFC
## t = 18.604, df = 9998, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  0.1639031 0.2017922
## sample estimates:
##       cor
## 0.1829156
```

Comment: The post-docs have introduced a lot of variance, so much so it is the third and fourth greatest contributor to variance in the data. To quantify the significance of this, one should use a statistical test.

How useful this data is depends on what it will be used for, it would be better to repeat the experiment, but the introduced difference could be counteracted by removing any systematic bias introduced by Alice or omitting those points and/or the single outlier caused by Bob entirely. This would perhaps make the data more representative of the true values.

Question 7: Caroline asks you to construct a 2-by-2 contingency table showing whether genes are significantly DE (after correcting for multiple testing, alpha=0.05) in the two knockdowns. Perform a Fisher's Exact test for any association between the DE genes in the two knockdowns. 3p

```
#Set Alpha
Alpha <- 0.05

#Subset DE dataframe by knock-down gene
GeneA <- data.frame(DE[1], DE[3], DE[5])
GeneB <- data.frame(DE[1], DE[6], DE[8])

#Add significance yes/no (1/0) using ifelse, as in Q5.5,
#for each subset.
GeneA_sig <- cbind(sigA = as.factor(ifelse(GeneA[,3]< Alpha, 1, 0)), GeneA)
GeneB_sig <- cbind(sigB = as.factor(ifelse(GeneB[,3]< Alpha, 1, 0)), GeneB)

#concatenate just the significance values for each gene into a data frame
fishme <- data.frame(GeneA_sig$sigA, GeneB_sig$sigB)

#Make into contingency table
fishme <- table(fishme)

#Rename parameters for readability
colnames(fishme) <- c("no", "yes")
rownames(fishme) <- c("no", "yes")

fishme
```

```
##                GeneB_sig.sigB
## GeneA_sig.sigA   no  yes
##            no  8104 1142
##            yes  592  162
```

```
#perform fishers exact test
fisher.test(fishme)
```

```
##
##  Fisher's Exact Test for Count Data
##
## data:  fishme
## p-value = 2.116e-11
## alternative hypothesis: true odds ratio is not equal to 1
## 95 percent confidence interval:
##  1.604151 2.340773
## sample estimates:
## odds ratio
##   1.941789
```

Comment: The null hypothesis here is that the proportion of significantly differentailly expressed genes is the same for the Gene A knock-down, and for the Gene B knock-down. They two follow the same distribution.

Here we have a p-value < 2.116e-11 which is highly significant, as such; we can reject the null hypothesis and assert that the proportion of significantly differentailly expressed genes is different between the Gene A and Gene B knock-downs. They don't follow the same distribution.

Question 8: Based on Question 4-7, do you think the geneA and geneB transcription factors regulate the same genes? Use a maximum of 100 words. 3p

Q5.4) Shows only the knock-downs don't illicit the same effect.

Q5.5) Shows no visible correlation between the logFC values of the knock-downs. We see Gene A requires a greater LogFC before significance than gene B.

Q5.6) Shows a slight significant positive correlation, I suspect this is false, and due to the large sample size and chance.

Q5.7) The Fisher-Exact-Test suggests there is a significant difference between the proportions of significantly differentially expressed genes between the two knock-downs. The two knock-downs do not share a distribution.

This suggests that they regulate different genes, or have very few genes in common.