STAT115/215 BIO/BST282 HW2

Your Name

Due Date: Sun 2/21/2021 midnight

Note for Grader:

I wanted to let you know that I was unable to make the changes that were sent out on Friday February 19. I finished this assignment early on purpose because I knew I would not have time to work on it over the weekend. I hope it is clear from the slack messages that I put a lot of effort in and tried to make these changes, but in the end I found it more important for my other schoolwork and my mental health to stick to my planned schedule and not redo the entirety of problem 1.

I hope that you will consider this when grading Homework 2, as I did put over 30 hours of time into it. I think other students would appreciate leniency as well, considering that the update was so late (3 days before the submission of a 2 week long assignment).

Part I: Differential expression

In this HW, we will evaluate the differentially expressed genes and pathways between breast cancer and normal breast tissues. Our collaborator generated RNA-seq on ten pairs of breast tumors and matched adjacent normal tissues, located at /n/stat115/2021/HW2/raw_data1. The experiments were run in two batches, each batch with 5 pairs of samples, and the batch information is provided in batch.csv. We have run Salmon for gene quantification which is provided in Cannon at /n/stat115/2021/HW2/raw_data1/Salmon_results. Mapping between Ensembl id and gene symbol is provides in tx2gene.csv.

Problem I.1

Please install the following R/Bioconductor packages. Then try “library(package)” to make sure the package works.

Note: sva package with Combat function is used for batch effect removal;

DESeq2 package is used for differential gene expression analysis;

tximport package is used for importing transcript-level abundance into gene-level matrices for downstream analysis

ggplot2 package is used for general plotting in R;

pheatmap package is used for heatmap plotting;

dplyr package is used for data frame manipulations in R;

fgsea package is used for gene-set enrichment analysis.

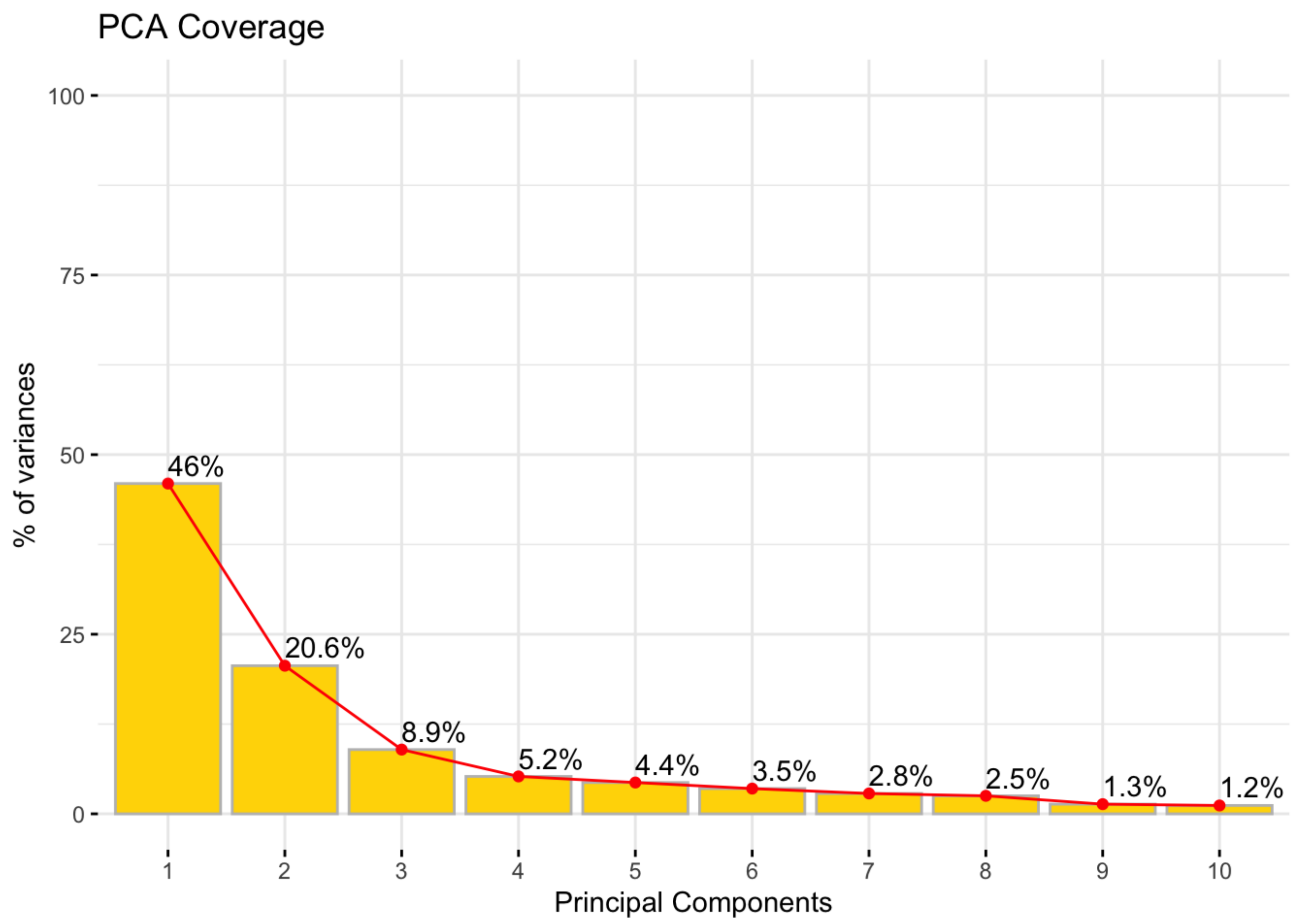
```
# ```{r install, eval = FALSE}
# if (!requireNamespace("BiocManager", quietly = TRUE))
#     install.packages("BiocManager")
# BiocManager::install("sva")
# BiocManager::install("DESeq2")
# BiocManager::install("tximport")
# install.packages(c("ggplot2", "dplyr",
#                    "pheatmap"))
# BiocManager::install("fgsea")
# BiocManager::install("ComplexHeatmap")
library(ggplot2)
library(sva)
library(DESeq2)
library(tximport)
library(dplyr)
library(fgsea)
library(pheatmap)
library(ComplexHeatmap)
library(factoextra)
library(dendextend)
library(apeglm)
library(EnhancedVolcano)
library(RColorBrewer)
library(tidyr)
```

Problem I.2

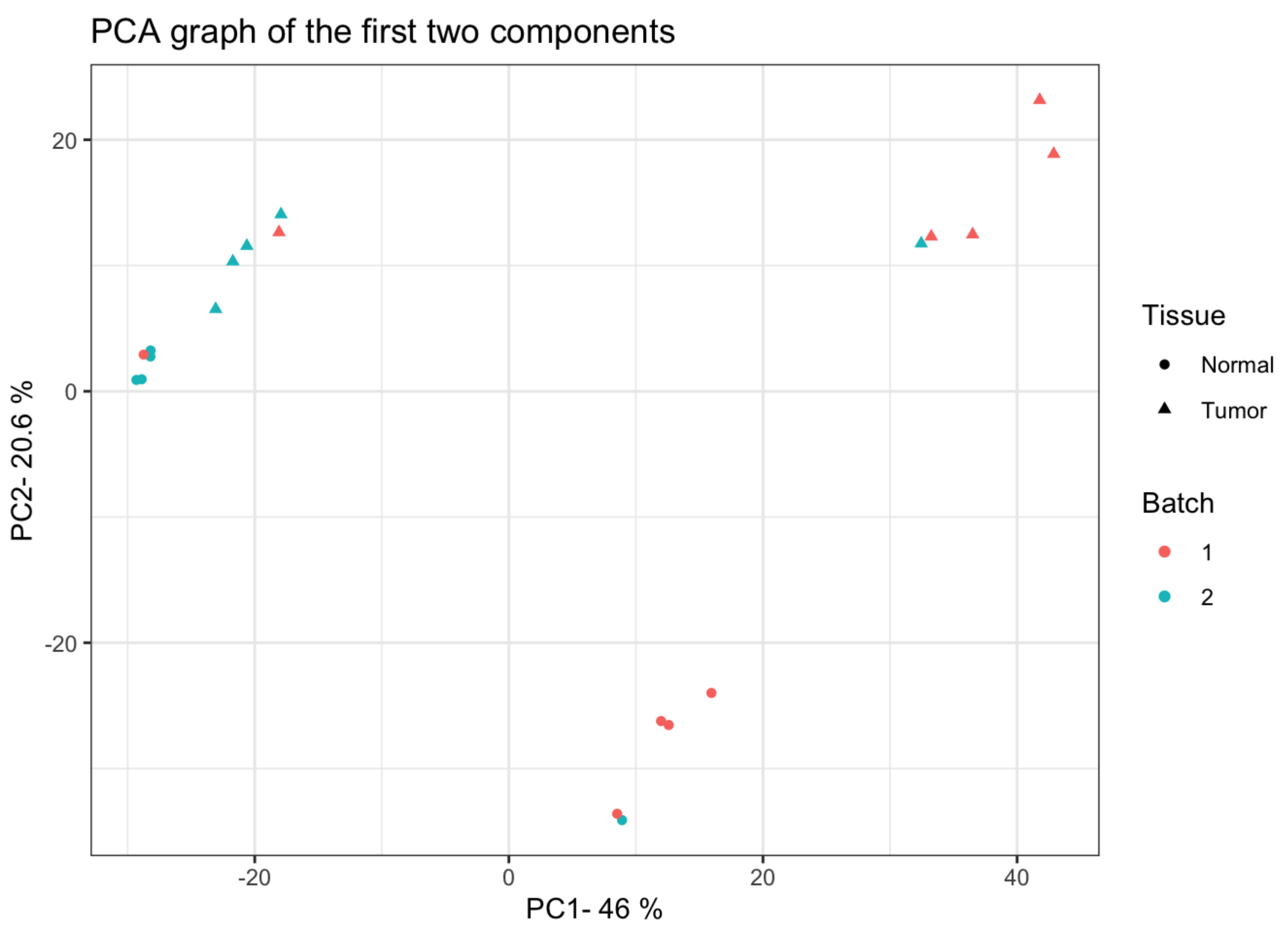
For RNA-seq analysis, visualization using principle component analysis (PCA) or hierarchical clustering is an essential step of exploratory analysis to identify potental batch effect. Please import transcript-level TPM values from Salmon output and convert to gene-level TPM values. Perform PCA of the samples using log2 transformed TPM values. Indicate different tissues (tumor or normal, using shape to show) and batch (1 or 2, using color to show) of each sample. Next try to use hierarchical clustering for these samples. Do the results suggest the presence of a batch effect?

For this question, you will load Salmon output at /n/stat115/2021/HW2/raw_data1/Salmon_results. You also need to read in batch information provided in /n/stat115/2021/HW2/raw_data1/batch.csv. Remember to convert Ensembl ID to gene symbol, using the mapping provided in /n/stat115/2021/HW2/raw_data1/tx2gene.csv.

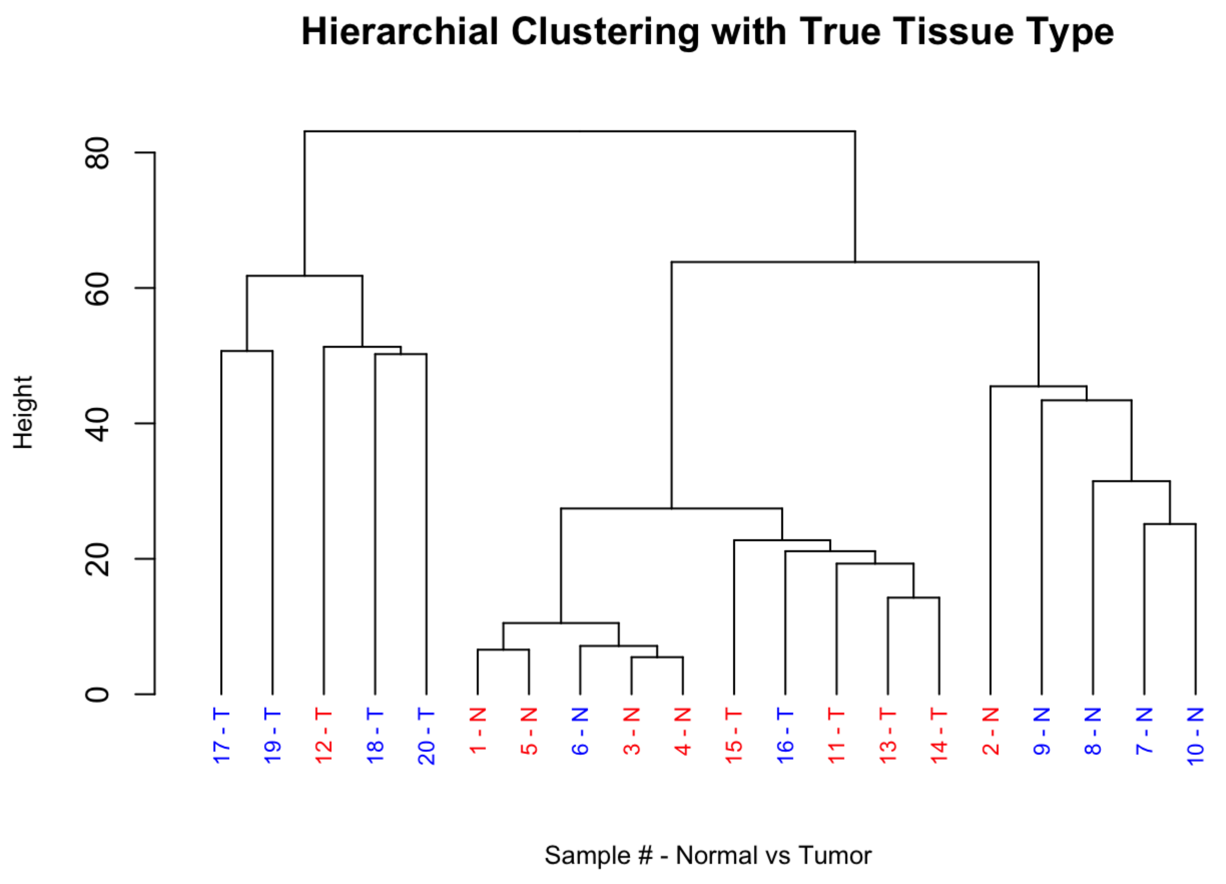
```r
### Your code here

# import transcript-level TPM values from Salmon output
files <- paste("./Salmon_results/", list.files(path = "./Salmon_results/"), sep = '')
tx2gene <- read.csv("tx2gene.csv")
txi <- tximport(files, type="salmon", tx2gene=tx2gene)

batch <- read.csv("batch.csv")

# convert to gene-level TPM values
counts <- txi$counts
tpm <- sapply(1:dim(counts)[2], function(idx){
  (counts[,idx]/sum(counts[,idx]) * 1000000) + 1
})
tpm_scaled <- scale(tpm)

log2tpm <- log2(tpm)
log2tpm_scaled <- scale(log2tpm)

# perform PCA of the samples using log2 transformed TPM values
res.pca <- prcomp(t(log2tpm_scaled))
pca.var.per = round(res.pca$sdev^2/sum(res.pca$sdev^2)*100,1)
fviz_eig(res.pca, addlabels=TRUE, ylim=c(0,100), geom = c("bar", "line"), barfill = "gol
d", barcolor="grey",linecolor = "red", ncp=10)+
labs(title = "PCA Coverage",
         x = "Principal Components", y = "% of variances")
```



```
# indicate different tissues (tumor or normal, using shape to show) and batch (1 or 2, u
sing color to show) of each sample
ggplot_data <- data.frame(
  Gene = colnames(res.pca$x),
  X = res.pca$x[,1],
  Y = res.pca$x[,2],
  Tissue = batch$tissue,
  Batch = as.factor(batch$batch)
)
ggplot(data = ggplot_data,aes(x=X,y=Y,label =Gene, shape = Tissue, color=Batch))+
  geom_point()+
  xlab(paste("PC1-",pca.var.per[1],"%",sep =" "))+
  ylab(paste("PC2-",pca.var.per[2],"%",sep =" "))+
  theme_bw()+
  ggtitle("PCA graph of the first two components")
```



```r
# try to use hierarchical clustering for these samples
d <- dist(t(log2tpm_scaled), method = "euclidean")
hc1 <- hclust(d, method = "complete" )
dend <- as.dendrogram(hc1)
labels(dend) <- paste(
  hc1$order, "-",
  sapply(batch$tissue, function(x) {substring(x, 1, 1)})[hc1$order]
)
labels_colors(dend) <- ifelse(batch$batch == 1, 'blue','red')[hc1$order]
dend %>%
  set("labels_cex", 0.7) %>%
  plot(main = "Hierarchial Clustering with True Tissue Type", xlab = "Sample # - Normal
 vs Tumor", ylab = 'Height', cex.lab = 0.8)
legend(-0.1, 200, legend = c("Batch 1","Batch 2"), col = c("blue","red"), lty = 1, cex =
0.6)
```



PCA Summary: The first two PCAs visualized in the plot account for 65.9% of the variation in scaled $log_2(TPM)$. In the plot of the first two PCAs, the shape represents the tissue type (circle as normal, triangle as tumor) and the color represents the batch (red as batch 1, blue as batch 2). The first two PCs do a very good job at separating the tissue types: A diagnonal line could split the tissue types perfectly. There is also clustering of batches: batch 1 (red) tends to be on the right half of the plot (positive values of PC1) while batch 2 (blue) tends to be on the left half (negative values of PC1). The batches are pretty well separated by the line PC1 = 0, with 4 mismatches.

Hierarchical Clustering Summary: The dendrogram resulting from hierarchical clustering of scaled $log_2(TPM)$ values, with color representing what batch the sample comes from (blue for batch 1, red for batch 2). Whether the sample is from tumor or normal tissue is also represented by a “T” or “N” in the label. While the hierarchical clustering clearly separates tumor and normal tissues, it perhaps even more clearly separates batch 1 from batch 2: you could get a pretty good separation of batches by cutting the tree into just 4 clusters (split at height 60, with 4 missmatches).

Based on the separability of batches by hierarchical clustering and principal component analysis, there is convincing evidence of a batch effect.

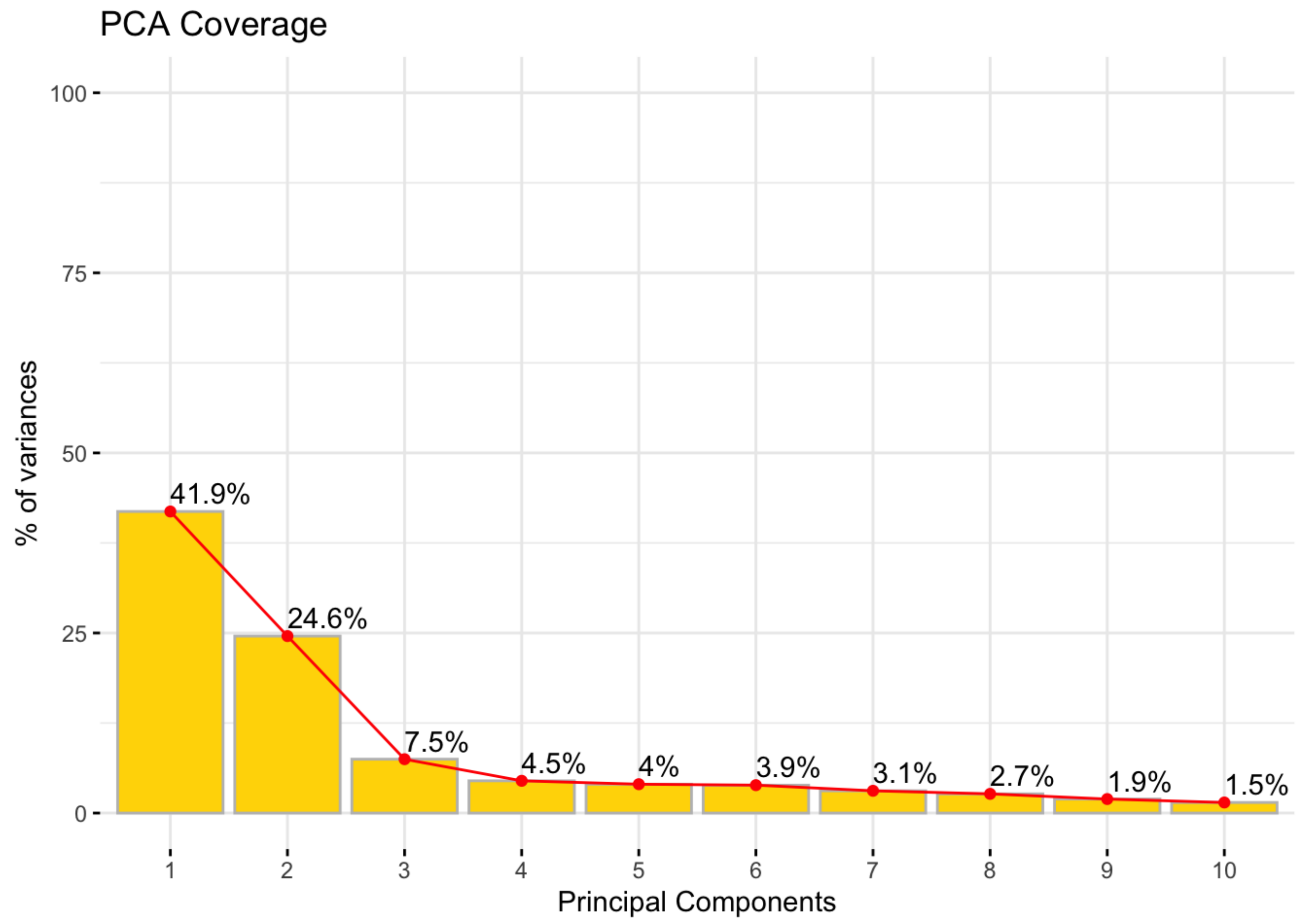
Problem I.3

Run COMBAT on the samples to remove the batch effect. Visualize the results using a similar PCA and hierarchical clustering as Problem 2. Provide evidence that the batch effects are successfully adjusted.

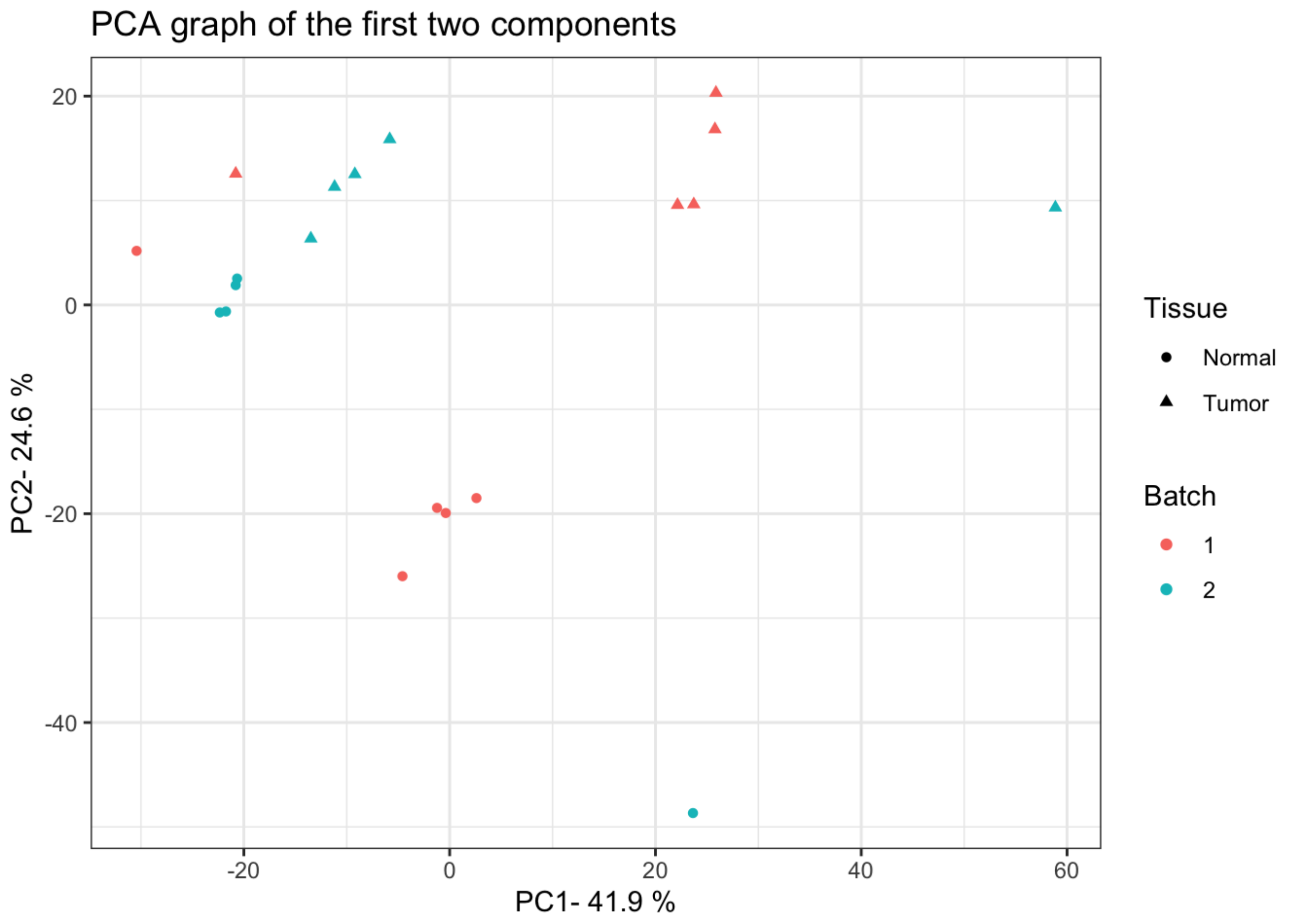
```
### Your code here
geneVars <- apply(tpm, MARGIN = 1, FUN = function(x) {sd(x)})
tpm_for_combat <- tpm[geneVars > 0,]

combat_dat <- ComBat(log2(tpm_for_combat), batch$batch)
```

```
## Found 436 genes with uniform expression within a single batch (all zeros); these will
not be adjusted for batch.
```

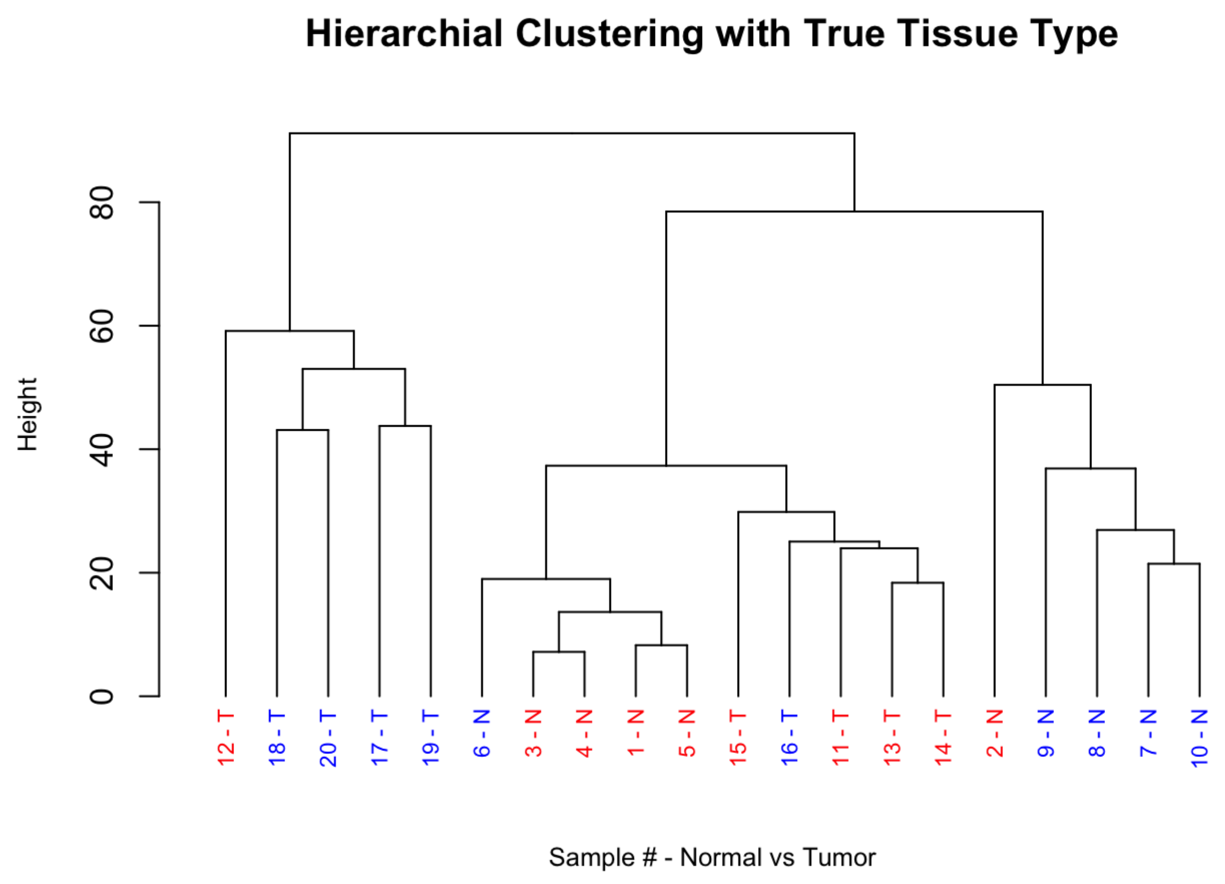
```
combat_dat_scaled <- scale(combat_dat)

combat.pca <- prcomp(t(combat_dat_scaled))

pca.var.per = round(combat.pca$sdev^2/sum(combat.pca$sdev^2)*100,1)
fviz_eig(combat.pca, addlabels=TRUE, ylim=c(0,100), geom = c("bar", "line"), barfill =
"gold", barcolor="grey",linecolor = "red", ncp=10)+
labs(title = "PCA Coverage",
         x = "Principal Components", y = "% of variances")
```



```
# indicate different tissues (tumor or normal, using shape to show) and batch (1 or 2, u
sing color to show) of each sample
ggplot_data <- data.frame(
  Gene = colnames(combat.pca$x),
  X = combat.pca$x[,1],
  Y = combat.pca$x[,2],
  Tissue = batch$tissue,
  Batch = as.factor(batch$batch)
)
ggplot(data = ggplot_data,aes(x=X,y=Y,label =Gene, shape = Tissue, color=Batch))+
  geom_point()+
  xlab(paste("PC1-",pca.var.per[1],"%",sep =" "))+
  ylab(paste("PC2-",pca.var.per[2],"%",sep =" "))+
  theme_bw()+
  ggtitle("PCA graph of the first two components")
```



```
# try to use hierarchical clustering for these samples
d <- dist(t(combat_dat_scaled), method = "euclidean")
hc1 <- hclust(d, method = "complete" )
dend <- as.dendrogram(hc1)
labels(dend) <- paste(
  hc1$order, "-",
  sapply(batch$tissue, function(x) {substring(x, 1, 1)})[hc1$order]
)
labels_colors(dend) <- ifelse(batch$batch == 1, 'blue','red')[hc1$order]
dend %>%
  set("labels_cex", 0.7) %>%
  plot(main = "Hierarchial Clustering with True Tissue Type", xlab = "Sample # - Normal
 vs Tumor", ylab = 'Height', cex.lab = 0.8)
legend(15, 50000, legend = c("Batch 1","Batch 2"), col = c("blue","red"), lty = 1, cex =
0.6)
```



I performed Combat on `log_2(tpm)` , as instructed in the lab. I only ran combat on genes with variances greater than zero, as directed by the TA in the slack. Then, I scaled these values as I had in problem I.2. It's important to perform combat on log2tpm, and then scale these values, so that the results are comparable to the scaled log2tpm values analyzed in I.2.

PCA Summary: After batch effect adjustment, the first principal component accounts 41.9% of the variation in scaled log2TPM, and the second accounts for 24.6%. The plot of the first two PCAs has the same axes, colors, and shapes as in Problem I.2. The batches look a bit more spread out – particluarly the two batch 2 points towards the right end of the plot. Batches are the same amount separable–separating samples at PC1 = -25 would give 4 mismatches, same as in I.2. Here, it seems they are more easily separable by tissue type (split at PC2 = -10). This plot looks like a rotation of that in I.2, and now PC1 accounts for batch while PC2 accounts for tissue.

Hierarchical Clustering Summary: The batches in the dendrogram are a bit more spread out. To separate batches to the same extent as described in I.2, there would have to be 6 clusters for 20 data points, rather than 4 clusters in I.2. It still seems, though, that samples within the same batch are more similar than samples within the same tumor type.

The batch effects are diminished, but definitely NOT perfectly adjusted.

Problem I.4

Run DESeq2 based on paired samples adjusting for the batch effect to identify differentially-expressed genes between tumor and normal tissues. How many genes are expressed higher in tumors than normal. Let's use 1) FDR < 0.01 and 2) Log2 fold change > 1 as the cutoff.

Note: please use the raw_count columns of the Salmon result and convert these to integer values for DESeq2.

Identify the top 5 most (by Log2FC) over expressed genes (FDR < 0.01) in tumor and normal, respectively.

```
### Your code here

# Run DESeq2 based on paired samples adjusting for the batch effect to identify differen
tially-expressed genes between tumor and normal tissues.
batch$batch <- as.factor(batch$batch)
batch$tissue <- as.factor(batch$tissue)
ddsTxi <- DESeqDataSetFromMatrix(
  countData = txi$counts, colData = batch, design = ~batch+tissue
)
dds <- DESeq(ddsTxi)
res <- results(dds, alpha = 0.01)

# How many genes are expressed higher in tumors than normal. Let's use 1) FDR < 0.01 and
2) Log2 fold change > 1 as the cutoff.
resLFC <- lfcShrink(dds, coef="tissue_Tumor_vs_Normal", type="apeglm", res = res)

n_overexpressed <- resLFC %>% data.frame() %>%
  filter(
    log2FoldChange > 1,
    padj < 0.01
  ) %>%
  nrow()
cat(n_overexpressed, "genes are expressed higher in tumors than normal.")
```

```
## 657 genes are expressed higher in tumors than normal.
```

```
# Identify the top 5 most (by Log2FC) over expressed genes (FDR < 0.01) in tumor and nor
mal, respectively.
top5over_tumor <- resLFC %>% data.frame() %>%
  filter(padj < 0.01) %>%
  arrange(-log2FoldChange) %>%
  rownames() %>%
  head(5)

cat(
  "\n\nThe top 5 over expressed genes (by Log2FC, with FDR < 0.01) in TUMOR samples ar
e:",
  paste(top5over_tumor, collapse = ', '), sep = '\n'
)
```

```
##
##
## The top 5 over expressed genes (by Log2FC, with FDR < 0.01) in TUMOR samples are:
## TRPA1, HOXB13, SLC30A8, CLEC3A, OR2B6
```

```
top5over_normal <- resLFC %>% data.frame() %>%
  filter(padj < 0.01) %>%
  arrange(log2FoldChange) %>%
  rownames() %>%
  head(5)

cat(
  "\nThe top 5 over expressed genes (by Log2FC, with FDR < 0.01) in NORMAL samples are:"
,
  paste(top5over_normal, collapse = ', '), sep = '\n'
)
```

```
##
## The top 5 over expressed genes (by Log2FC, with FDR < 0.01) in NORMAL samples are:
## MYL7, FABP7, KRT14, SOX10, SMYD1
```

To adjust for the batch effect, I included batch in the model design when building the DESeq Data Set. By default, the DESeqDataSetFromTximport function converts counts to integers, so this requirement is covered. I also specified `alpha = 0.01` when acquiring DESeq results, as instructed.

I found that there are **657 genes expressed higher in tumors than normal samples**, using the cutoffs Log2FoldChange > 1 and FDR < 0.01. The top 5 overexpressed genes in tumor samples are **TRPA1, HOXB13, SLC30A8, CLEC3A, and OR2B6**. The top 5 overexpressed genes in normal samples are **MYL7, FABP7, KRT14, SOX10, and SMYD1**.

Problem I.5

Visualize the differential gene expression values by making a volcano and an MA plot to summarize the differences between tumor and normal. In the volcano plot, draw the statistically significant (FDR < 0.01, Log2FC > 1) tumor up-genes red and down-genes blue.

Note: Be sure to use the lfcShrink function to get more robust estimates of the fold-changes for genes.

```
### Your code here

# MA plot
resApeT <- lfcShrink(dds, coef=2, type="apeglm", lfcThreshold=1)
```

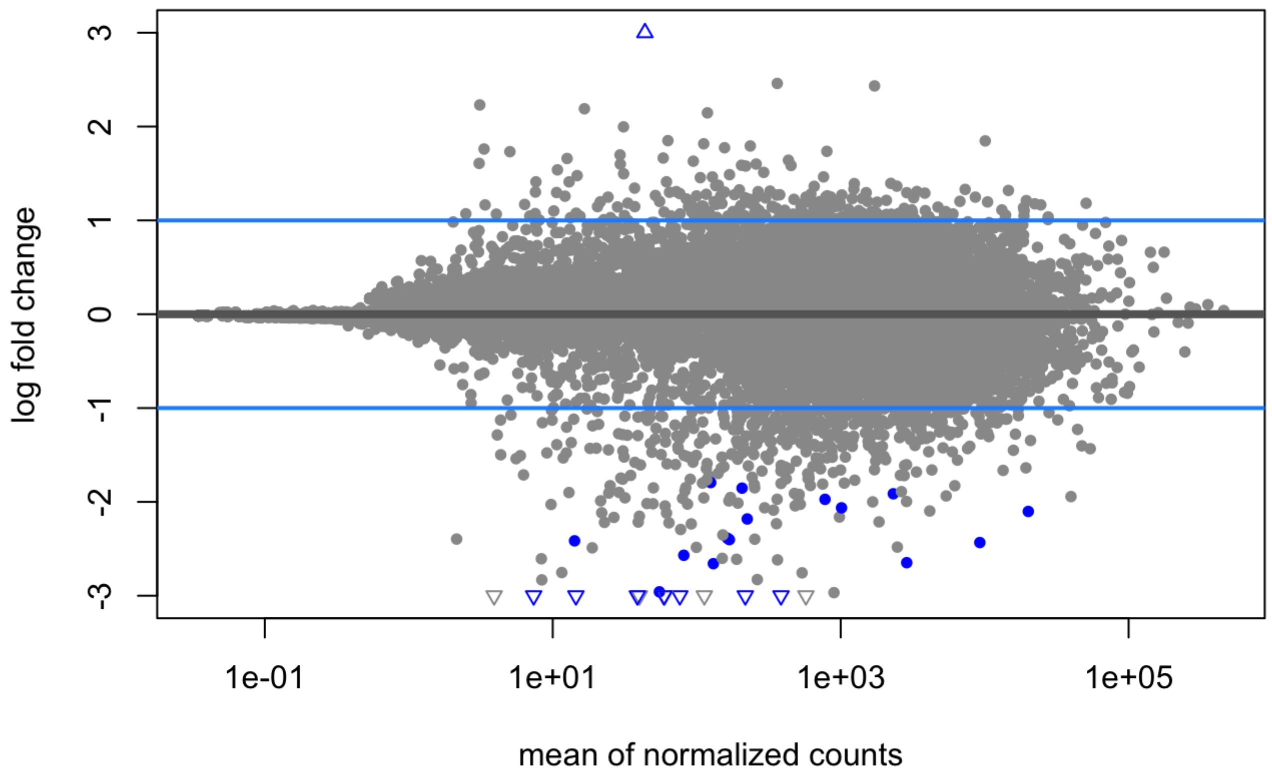
```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

```
## computing FSOS 'false sign or small' s-values (T=1)
```

```
plotMA(resApeT, ylim=c(-3,3), cex=.8)
```

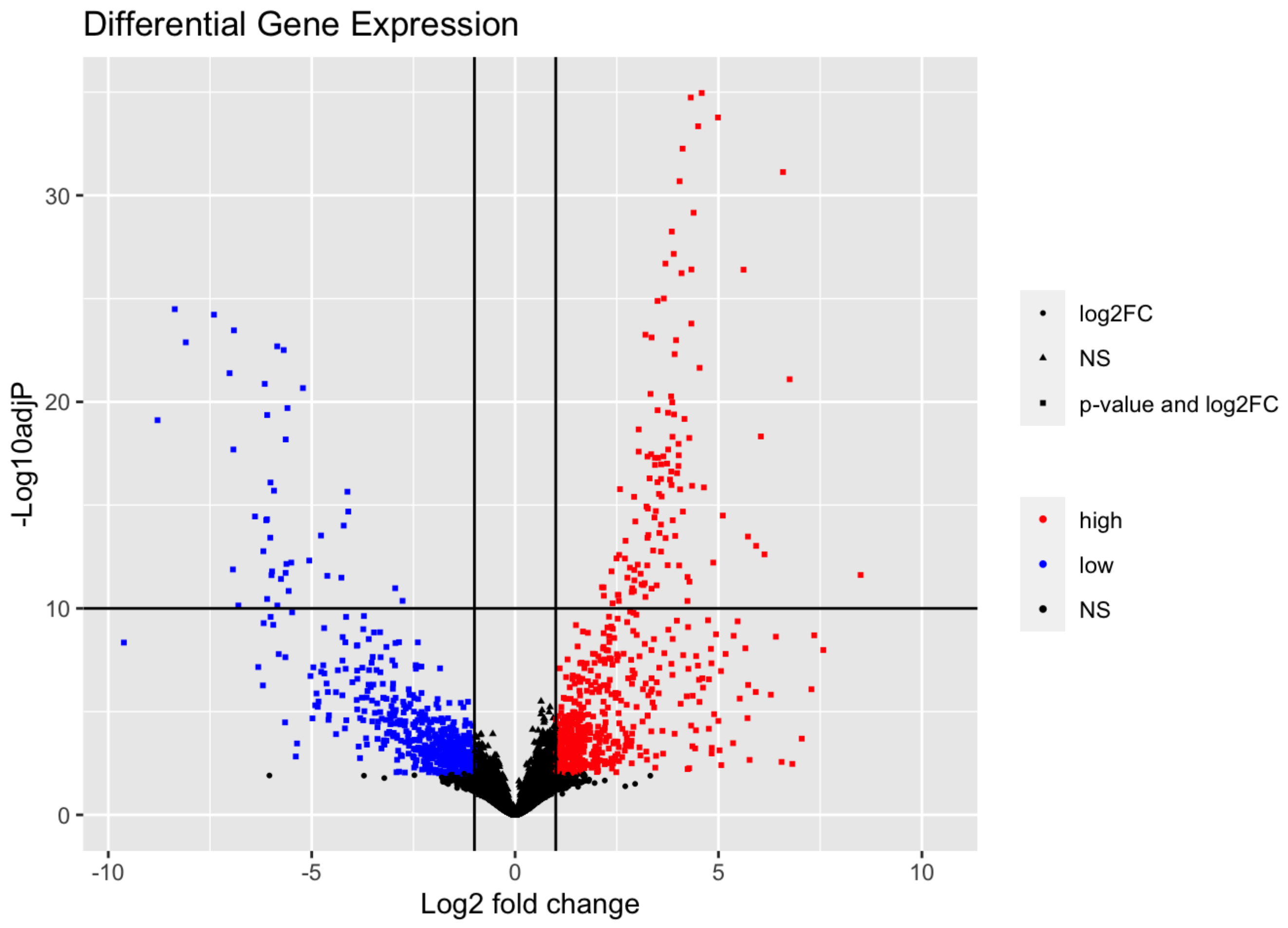
```
## thresholding s-values on alpha=0.005 to color points
```

```
abline(h=c(-1,1), col="dodgerblue", lwd=2)
```



```
# Volcano plot: draw the statistically significant (FDR < 0.01, Log2FC > 1) tumor up-gen
es red and down-genes blue.
up_genes <- rownames(resLFC)[which(resLFC$log2FoldChange > 1 & resLFC$padj < 0.01)]
down_genes <- rownames(resLFC)[which(resLFC$log2FoldChange < -1 & resLFC$padj < 0.01)]

plotdat <- resLFC %>% data.frame() %>%
  mutate(
    color = case_when(
      log2FoldChange > 1 & padj < 0.01 ~ 'high',
      log2FoldChange < -1 & padj < 0.01 ~ 'low',
      TRUE ~ 'NS'
    ),
    shape = case_when(
      abs(log2FoldChange) > 1 & padj < 0.01 ~ "p-value and log2FC",
      abs(log2FoldChange) > 1 & padj > 0.01 ~ "log2FC",
      TRUE ~ "NS"
    ),
    neglog10padj = -log10(padj)
  )

plotdat %>%
  ggplot(aes(x = log2FoldChange, y = neglog10padj, shape = shape, color = color)) +
    geom_point(size = 0.7) +
    theme(
      legend.title = element_blank()
    ) +
    scale_color_manual(
      breaks = c('high','low','NS'),
      values = c('red','blue','black')
    ) +
    xlab("Log2 fold change") +
    ylab("-Log10adjP") +
    ggtitle("Differential Gene Expression") +
    geom_hline(yintercept = 10) +
    geom_vline(xintercept = c(1, -1))
```

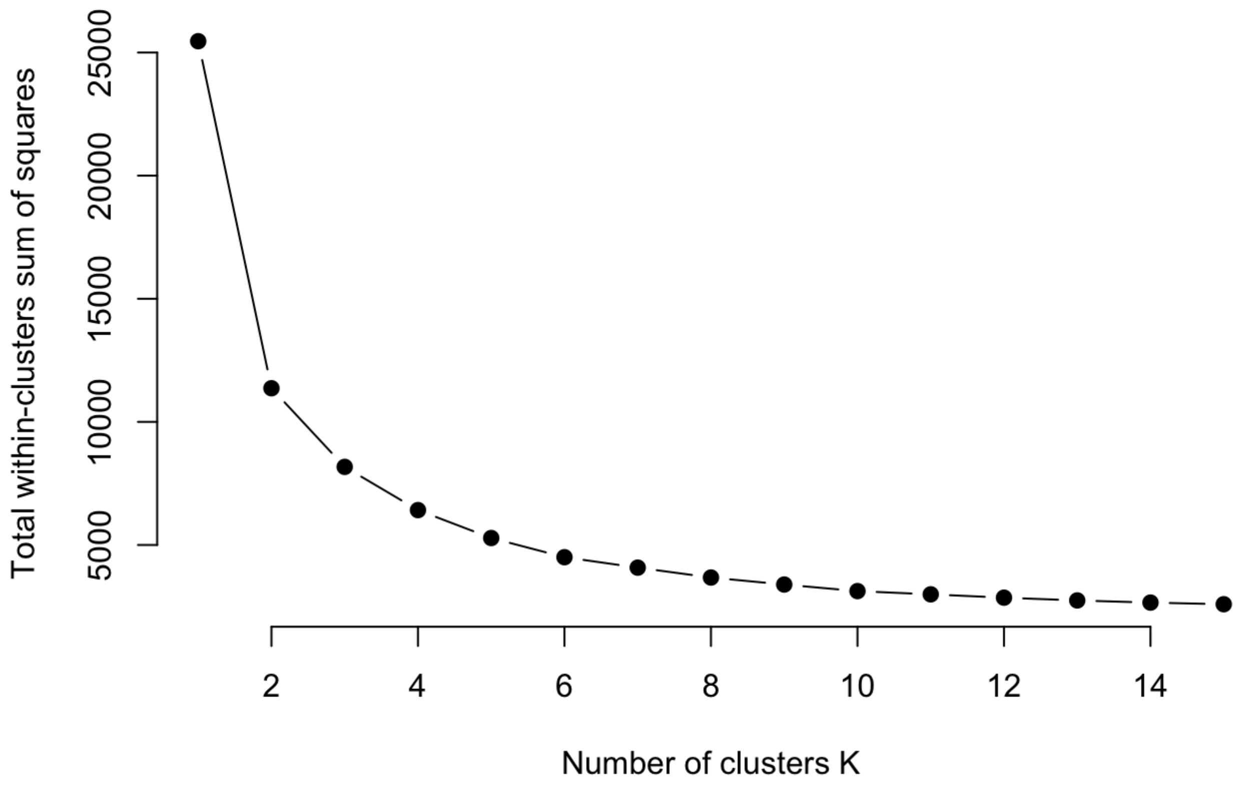
```
## Warning: Removed 2478 rows containing missing values (geom_point).
```



Problem I.6

Try kmeans (try k = 4 or 7) clustering to group differentially expressed genes into different clusters. How many genes are there in each cluster? Draw a heatmap showing gene clusters.

```
### Your code here

# subset Combat-corrected matrix by differentially expressed genes obtained from DESeq2
kmeans_dat <- data.frame(combat_dat) %>%
  filter(
    abs(resLFC$log2FoldChange)[geneVars > 0] > 1,
    resLFC$padj[geneVars > 0] < 0.01
  )

# dim(kmeans_dat) # 1274x20, we are clustering genes, so each row is a gene (observatio
n), each column is a variable (sample)

kmeans_dat <- na.omit(kmeans_dat)
kmeans_dat <- scale(kmeans_dat)

set.seed(123)

# function to compute total within-cluster sum of square
wss <- function(k) {
  kmeans(kmeans_dat, k, nstart = 10 )$tot.withinss
}

# Compute and plot wss for k = 1 to k = 15
k.values <- 1:15

# extract wss for 2-15 clusters
wss_values <- lapply(k.values, wss)

plot(k.values, wss_values,
       type="b", pch = 19, frame = FALSE,
       xlab="Number of clusters K",
       ylab="Total within-clusters sum of squares")
```



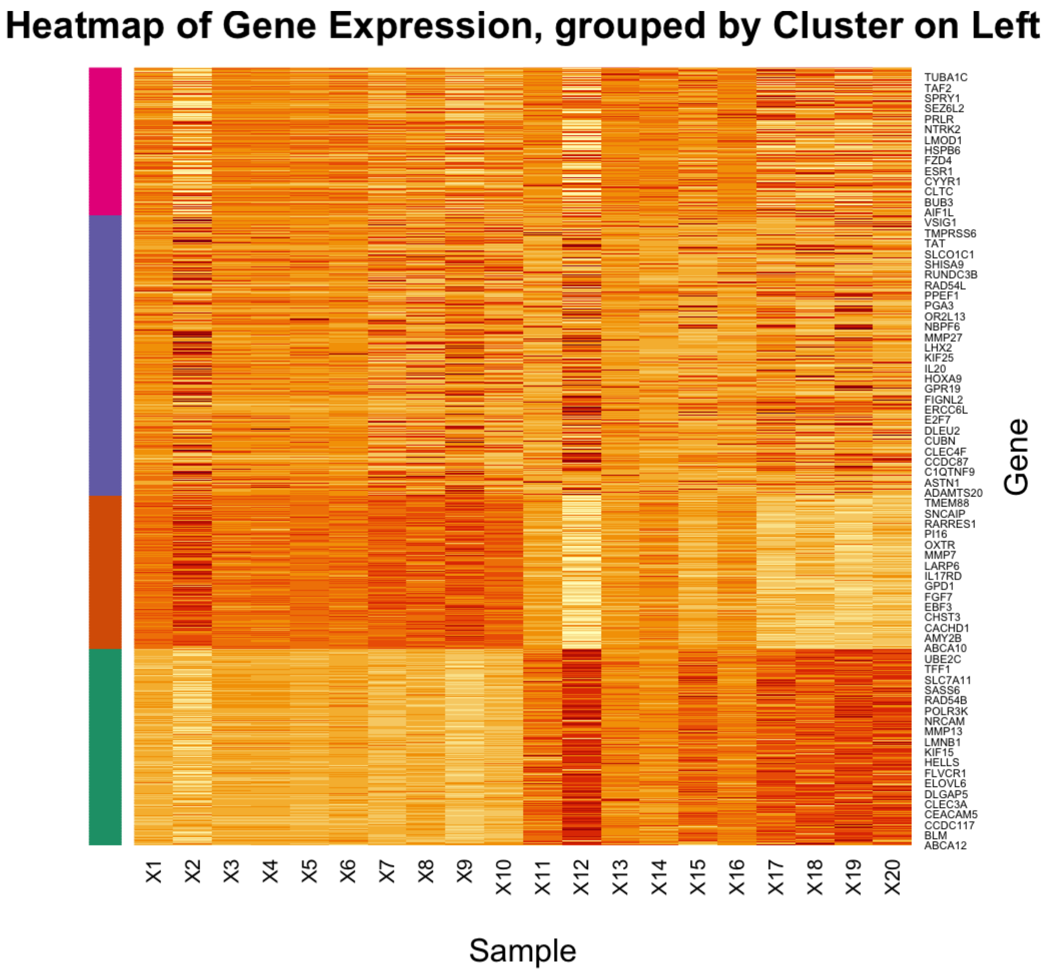
I am also running K means clustering on genes with non-zero variance, as I did with combat. Between 4 and 7 clusters, there is not a significant decrease in total within-clusters sum of squares, so I think 4 clusters will be sufficient. Just in case, I also perform k-means clustering with 4 clusters to see if any additional patterns are visibly obvious with more clusters.

k = 4

```
k4 <- kmeans(kmeans_dat, centers = 4, nstart = 25)
cat("The sizes of the four clusters are:", paste(k4$size, collapse = ', '))
```

```
## The sizes of the four clusters are: 322, 251, 459, 242
```

```
heatmap_dat <- kmeans_dat[order(k4$cluster),] # ordered by cluster
clust <- k4$cluster[order(k4$cluster)]
colSide <- brewer.pal(4, 'Dark2')[clust]
heatmap(
  heatmap_dat, Colv = NA, Rowv = NA,
  RowSideColors=colSide, xlab = 'Sample', ylab = 'Gene',
  main = 'Heatmap of Gene Expression, grouped by Cluster on Left'
)
```



With 4 clusters, some obvious clusters arise. The top two clusters (pink and purple on the left) have lower expression values relative to the bottom two.

The first has low expression in samples 2 and 12. With those two exceptions, this cluster has very slightly higher expressions in samples 11-20 than samples 1-10.

The second cluster has high expression in samples 2 and 12. With those exeptions, this cluster has very slightly higher expressions in samples 1-10 than 11-20.

The third cluster has a similar pattern to the second, with higher expression in samples 1-10, but to a greater extent. This cluster also has low expression in sample 12, unlike the second cluster.

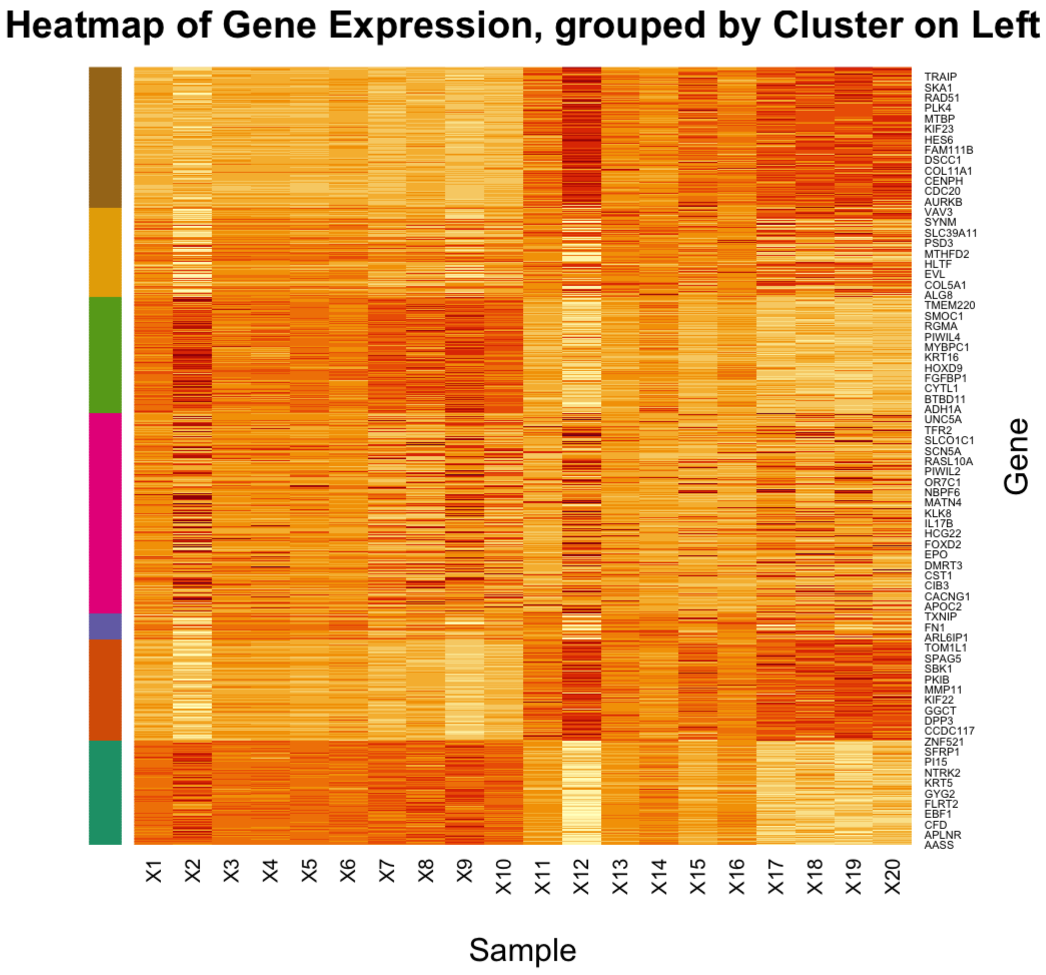
The final cluster has a similar pattern to the first, with higher expression in samples 11-20, but to a much greater extent (darker orange colors). Further, this cluster has high expression in sample 12, unlike the first cluster.

k = 7

```
k7 <- kmeans(kmeans_dat, centers = 7, nstart = 25)
cat("The sizes of the seven clusters are:", paste(k7$size, collapse = ', '))
```

```
## The sizes of the seven clusters are: 171, 166, 42, 329, 189, 146, 231
```

```
heatmap_dat <- kmeans_dat[order(k7$cluster),] # ordered by cluster
clust <- k7$cluster[order(k7$cluster)]
colSide <- brewer.pal(7, 'Dark2')[clust]
heatmap(
  heatmap_dat, Colv = NA, Rowv = NA,
  RowSideColors=colSide, xlab = 'Sample', ylab = 'Gene',
  main = 'Heatmap of Gene Expression, grouped by Cluster on Left'
)
```



I don’t see any obvious visual patterns that arise with 7 clusters above those that were seen with 4. The first and second to last cluster reflect the last cluster of the earlier plot. The second and fifth cluster here have the same patterns as the first cluster of the earlier plot. The third and last cluster here reflects the second to last of the earlier plot. Finally, the fourth clusters here show the same patterns as the second cluster of the earlier plot.

Because of the minimal decrease in variance and the lack of obvious new patterns with 7 clusters over 4, I think the clustering with k = 4 is best.

Problem I.7: For graduate students only

If you run DESeq2 without removing batch effect, how many differential genes do you get? How do their K-means clustering look? Does batch effect removal gives more interpretable results?

```
### Your code here

# run DESeq2 without removing batch effect
ddsTxi_nobatch <- DESeqDataSetFromMatrix(
  countData = txi$counts, colData = batch, design = ~tissue
)
```

```
## converting counts to integer mode
```

```
dds_nobatch <- DESeq(ddsTxi_nobatch)
```

```
## estimating size factors
```

```
## estimating dispersions
```

```
## gene-wise dispersion estimates
```

```
## mean-dispersion relationship
```

```
## final dispersion estimates
```

```
## fitting model and testing
```

```
## -- replacing outliers and refitting for 633 genes
## -- DESeq argument 'minReplicatesForReplace' = 7
## -- original counts are preserved in counts(dds)
```

```
## estimating dispersions
```

```
## fitting model and testing
```

```
res_nobatch <- results(dds_nobatch, alpha = 0.01)

resLFC_nobatch <- lfcShrink(dds_nobatch, coef="tissue_Tumor_vs_Normal", type="apeglm", r
es = res_nobatch)
```

```
## using 'apeglm' for LFC shrinkage. If used in published research, please cite:
##     Zhu, A., Ibrahim, J.G., Love, M.I. (2018) Heavy-tailed prior distributions for
##     sequence count data: removing the noise and preserving large differences.
##     Bioinformatics. https://doi.org/10.1093/bioinformatics/bty895
```

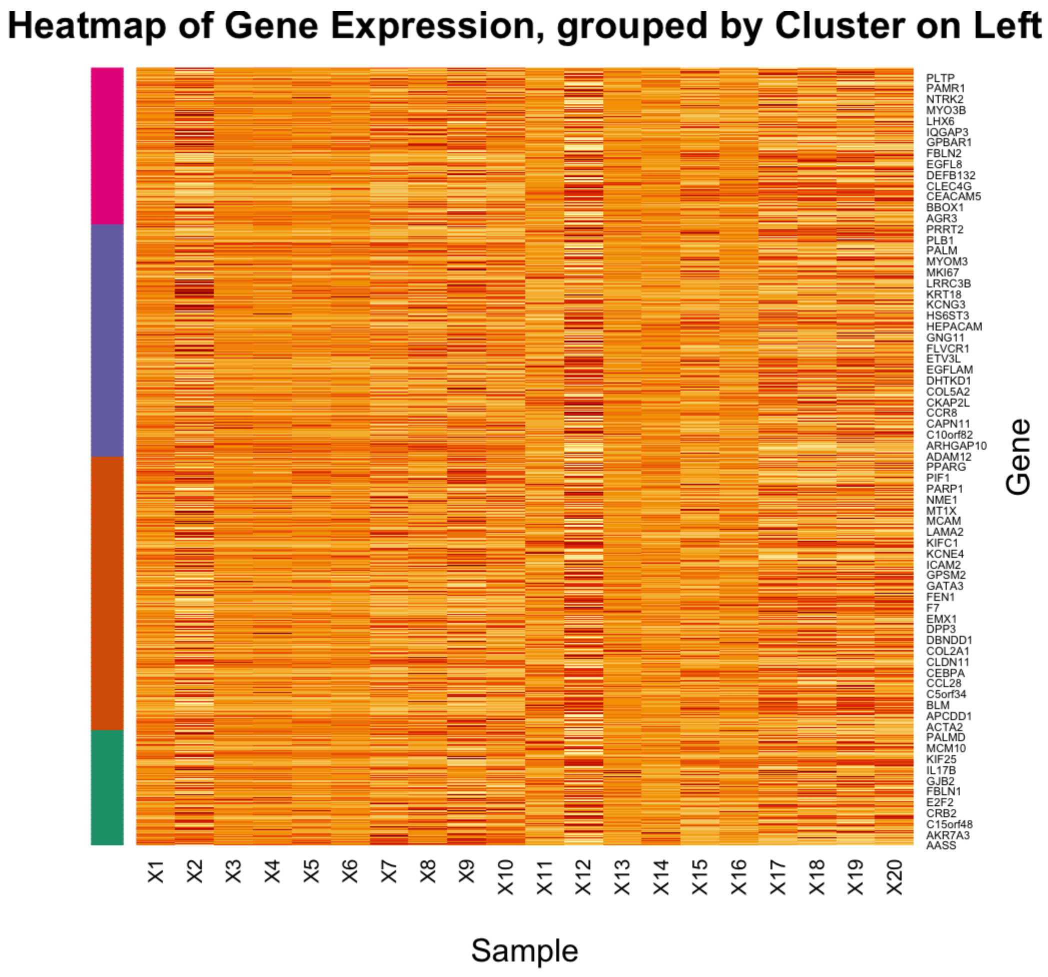
```
# how many differential genes do you get?
n_overexpressed <- resLFC_nobatch %>% data.frame() %>%
  filter(
    abs(log2FoldChange) > 1,
    padj < 0.01
  ) %>%
  mutate(
    over_under = ifelse(log2FoldChange > 0, 'over','under')
  ) %>%
  group_by(over_under) %>%
  summarize(.groups = NULL, n = length(log2FoldChange))

cat(n_overexpressed$n %>% sum(), "genes are differentially expressed.", n_overexpressed
$n[1], "are expressed higher in tumors than normal,", n_overexpressed$n[2], "are express
ed higher in normal than tumors.")
```

```
## 935 genes are differentially expressed. 638 are expressed higher in tumors than norma
l, 297 are expressed higher in normal than tumors.
```

```
# How do their K-means clustering look?
# subset uncorrected matrix by differentially expressed genes obtained from DESeq2
kmeans_dat_nobatch <- data.frame(log2tpm) %>%
  filter(
    abs(resLFC_nobatch$log2FoldChange) > 1,
    resLFC_nobatch$padj < 0.01
  )

kmeans_dat_nobatch <- na.omit(kmeans_dat_nobatch)
kmeans_dat_nobatch <- scale(kmeans_dat_nobatch)

k4_nobatch <- kmeans(kmeans_dat_nobatch, centers = 4, nstart = 25)
cat("The sizes of the four clusters are:", paste(k4_nobatch$size, collapse = ', '))
```

```
## The sizes of the four clusters are: 139, 328, 280, 188
```

```
heatmap_dat <- kmeans_dat[order(k4_nobatch$cluster),] # ordered by cluster
clust <- k4_nobatch$cluster[order(k4_nobatch$cluster)]
colSide <- brewer.pal(4, 'Dark2')[clust]
heatmap(
  heatmap_dat, Colv = NA, Rowv = NA,
  RowSideColors=colSide, xlab = 'Sample', ylab = 'Gene',
  main = 'Heatmap of Gene Expression, grouped by Cluster on Left'
)
```



Here, I again used 4 clusters as decided in problem I.6. This means it is a direct comparison to look at this heatmap and the one in Problem I.6 using 4 clusters.

935 genes are differentially expressed. 638 are expressed higher in tumors than normal, 297 are expressed higher in normal than tumors.The sizes of the four clusters are: 139, 328, 188, 280.

There are no distinct patterns across the clusters when batch effects are not removed. In problem I.6, there are clear patterns even across only 4 clusters. In conclusion, batch effect removal gives much more interpretable results.

Problem I.8

From the batch-removed DESeq2 run results, extract the top 200 tumor-upregulated genes (by Log2FC, FDR < 0.01). Run DAVID GO analysis (http://david.abcc.ncifcrf.gov/ (http://david.abcc.ncifcrf.gov/)) to see whether these genes are enriched in specific biological process (BP), pathways, etc.

```
### Your code here

# top 200 tumor-upregulated genes (by Log2FC, FDR < 0.01)
top200 <- resLFC %>% data.frame() %>%
  filter(padj < 0.01) %>%
  arrange(-log2FoldChange) %>%
  head(200)

# print out top 200 gene names to copy and paste into DAVID
# top200 %>% rownames() %>% paste(collape = '\n') %>% cat()
```

What I Did

I copied and pasted the list of 200 genes into DAVID, kept the identifier as it's default (AFFYMETRIX_3PRIME_IVT_ID), selected "Gene List", and clicked Submit List. Then, I entered Homo sapiens as the species int he main panel and submitted this to the conversion tool. Once conversion was finished, I clicked "Convert All". Then I clicked "Submit Converted List to DAVID as a Gene List". Back in the DAVID tab, the list and species were pre-selected. I selected the "Functional Annotation Tool", then "Functional Annotation Clustering".

Results

The most commonly singificantly enriched terms seem to relate to cell division and the p53 signaling pathway.

Cell Division

The annotation cluster with the largest enrichment score has an enrichment score of 25.44, and includes keywords relating to the cell cycle and cell division (cell cycle, mitosis, cell division, etc.). the next highest score is 13.93 and has keywords also related to cell division, but specifically about actions and parts of the cell involved in the process (sister chromatid cohesion, kinetochore, etc.). In fact, the first 10 clusters only contain words related to cell division.

Cyclin

Cluster 11 relates to Cyclin, which is related to cell division, and the p53 signaling pathway. This cluster has a much lower enrichment score of 2.53, but all terms still have significant p-values even after multiple hypothesis adjustments.

Estrogen and pf53

Cluster 12 includes significant terms about the estogen-responsive protein Efp and the p53 signaling.

HOX

Cluster 13 includes significant terms about Homeobox and DNA binding.

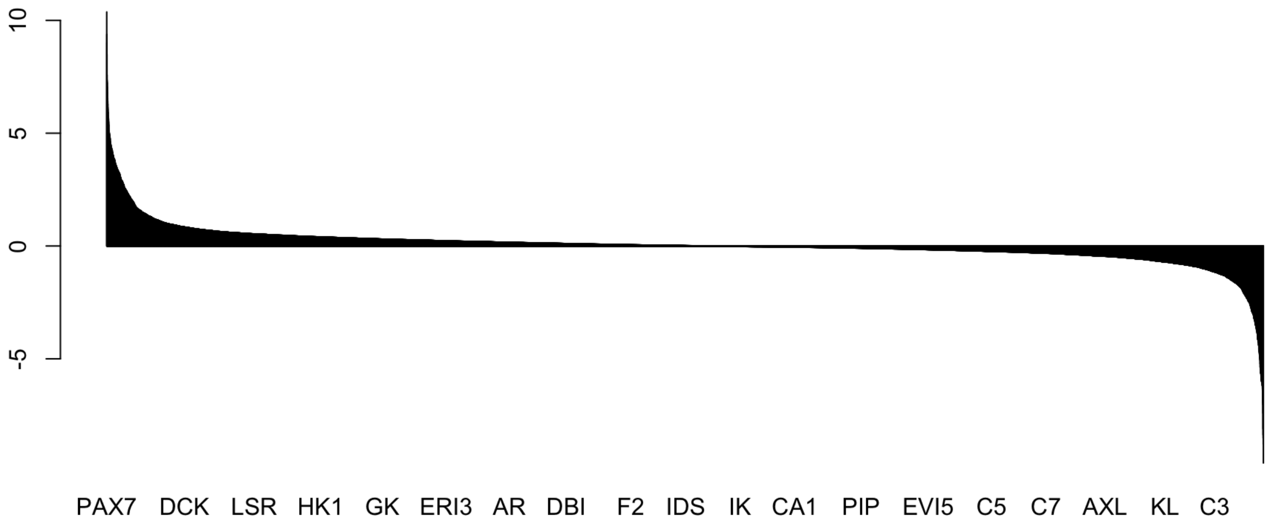
All further clusters have many non-significant terms, so I won't go into their descriptions.

Problem I.9: For graduate students only

Run Gene Set Enrichment analysis (http://www.broadinstitute.org/gsea/index.jsp (http://www.broadinstitute.org/gsea/index.jsp)) using the summary statistics from problem 4. Show the top five gene sets or pathways that best capture the differentially expressed genes between tumor than normal. Comment on the biological relevance of the results. Plot GSEA enrichment plot for an interesting pathway.

Mapping between gene sets and pathways is provided in /n/stat115/2021/HW2/raw_data1/c2.cp.kegg.v7.1.symbols.gmt file.

```
### Your code here
ranks <- resLFC$log2FoldChange
names(ranks) <- rownames(resLFC)
ranks <- sort(ranks, decreasing = T)
barplot(ranks)
```



```
pathways <- gmtPathways('c2.cp.kegg.v7.1.symbols.gmt')
fgseaRes <- fgsea(pathways, ranks, minSize=15, maxSize = 500)
```

```
## Warning in preparePathwaysAndStats(pathways, stats, minSize, maxSize, gseaParam, : Th
ere are ties in the preranked stats (1.86% of the list).
## The order of those tied genes will be arbitrary, which may produce unexpected result
s.
```

```
## Warning in fgseaMultilevel(...): For some pathways, in reality P-values are less
## than 1e-10. You can set the `eps` argument to zero for better estimation.
```

```
#Show the top five gene sets or pathways
head(fgseaRes[order(padj, -abs(NES)), ], n=5)
```

```
##                                               pathway         pval         padj
## 1:                                    KEGG_CELL_CYCLE 1.000000e-10 0.0000000176
## 2:                                KEGG_OOCYTE_MEIOSIS 1.580989e-06 0.0001391271
## 3:                                KEGG_DNA_REPLICATION 4.097163e-06 0.0002403669
## 4:                            KEGG_RETINOL_METABOLISM 7.045695e-06 0.0003100106
## 5: KEGG_METABOLISM_OF_XENOBIOTICS_BY_CYTOCHROME_P450 3.319069e-05 0.0011683124
##      log2err         ES       NES size
## 1:        NA  0.7269101  2.334271  118
## 2: 0.6435518  0.6313062  1.991313  108
## 3: 0.6105269  0.7748147  2.042676   36
## 4: 0.6105269 -0.6988509 -2.058796   59
## 5: 0.5573322 -0.6595570 -1.996878   65
##                                   leadingEdge
## 1:     CDC25C,PKMYT1,CCNE2,BUB1B,BUB1,CDC45,...
## 2:      SPDYC,CDC25C,PKMYT1,CCNE2,BUB1,PLK1,...
## 3:       MCM4,POLE2,DNA2,MCM2,PCNA,RNASEH2A,...
## 4:     ADH4,RDH5,UGT2B11,RPE65,DGAT2,CYP1A1,...
## 5: ADH4,UGT2B11,AKR1C2,AKR1C1,CYP1A1,AKR1C3,...
```

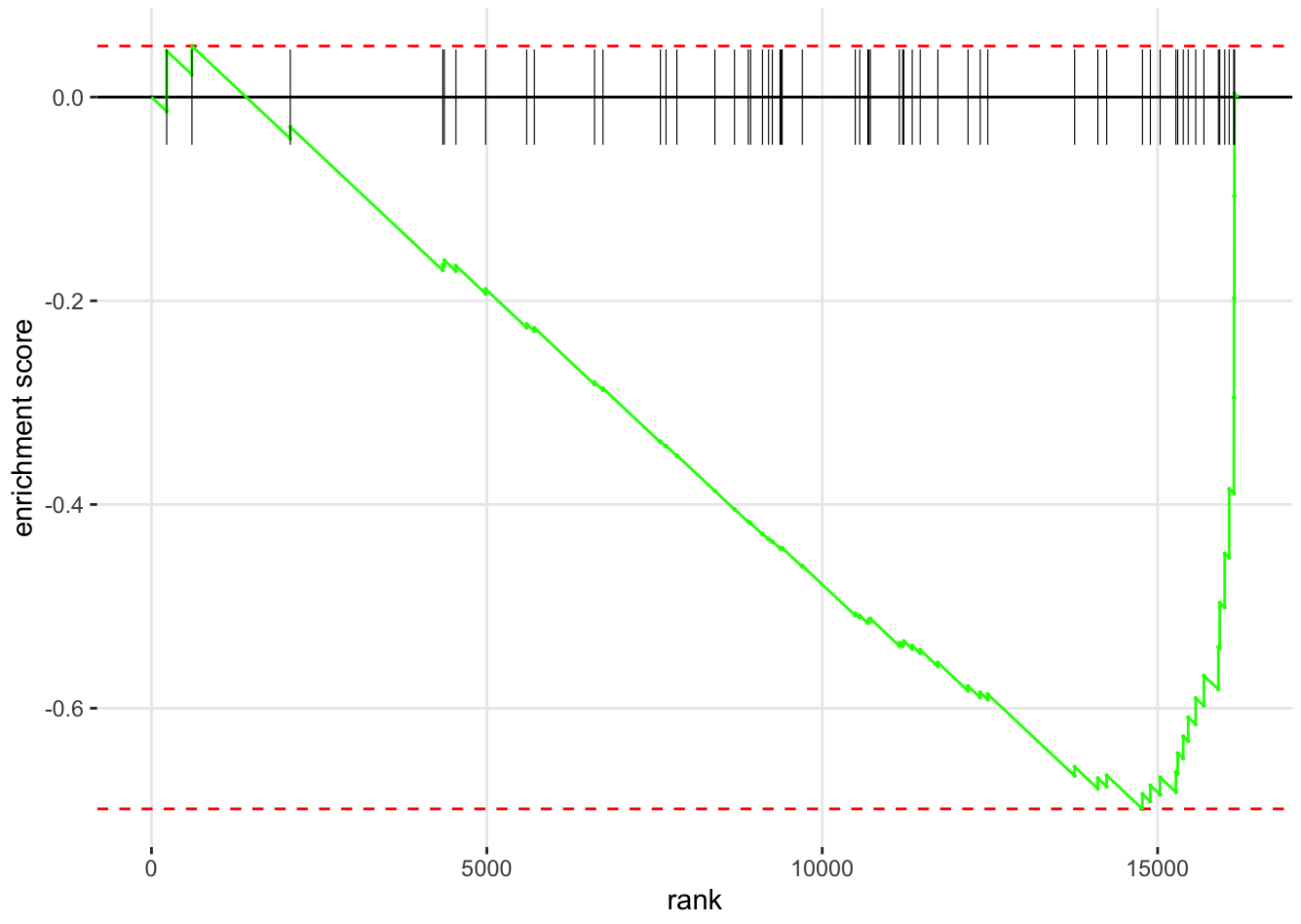
The top five gene sets that best capture the differentially expressed genes between breast cancer and normal are the cell cycle, oocyte meiosis, dna replication, retinol metabolism, and metabolism of xenobiotics by cytochrome p450. These top categories line up pretty well with what was seen in DAVID's functional annotation clustering.

Biologically, the significance of sets relating to cell cycle (the first 3) means that a major contributor to the difference between breast cancer cells and normal cells have to do with altered cell cycle behavior. This makes a lot of sense considering that cancer is caused by exatly that–abnormal cell division.

Studies have confirmed the association between breast cancer and retinol metabolism (e.g. "Defective RA metabolism in local populations of cells is thought to be a main initiator of mammary tumor formation and progression." link (https://faseb.onlinelibrary.wiley.com/doi/abs/10.1096/fasebj.25.1_supplement.lb324)) as well as xenobiotic metabolism (e.g. "The presence of different xenobiotic metabolizing enzymes may have a role in determining the intrinsic drug resistance of breast cancer" link (https://pubmed.ncbi.nlm.nih.gov/8492228/)).

Below is a plot of the GSEA enrichment for the retinol metabolism set:

```
plotEnrichment(pathways[["KEGG_RETINOL_METABOLISM"]], ranks)
```



Part II: Sample classification

We provide you z-score normalized expression data of 50 breast tumor samples, 50 normal breast samples (your training and cross-validation data), and 20 samples without diagnosis (your testing data). We want to use the 100 samples with known diagnosis to train machine learning models in order to predict the 20 unknown samples.

You will need the following libraries in R: ggplot2 and ggfortify for plotting, MASS and caret for machine learning, and pROC is for evaluating testing performance. The YouTube video on caret (https://youtu.be/z8PRU46I3NY (https://youtu.be/z8PRU46I3NY)) and the package documentation (http://topepo.github.io/caret/index.html (http://topepo.github.io/caret/index.html)) might be helpful.

All data for Part II are provided at /n/stat115/2021/HW2/raw_data2.

```
library(ggplot2)
library(ggfortify)
library(pROC)
library(caret)
library(e1071)
library(ROCR)
```

```
diagnosis <- read.table("diagnosis.txt", header = 1)
unknown <- read.table("unknown_samples.txt", header = 1)
pheno <- read.table("BRCA_phenotype.txt", header = 1)
zscore <- read.table("BRCA_zscore_data.txt", header = 1)

dim(zscore)
```

```
## [1]   100 20501
```

```
dim(unknown)
```

```
## [1]    20 20501
```

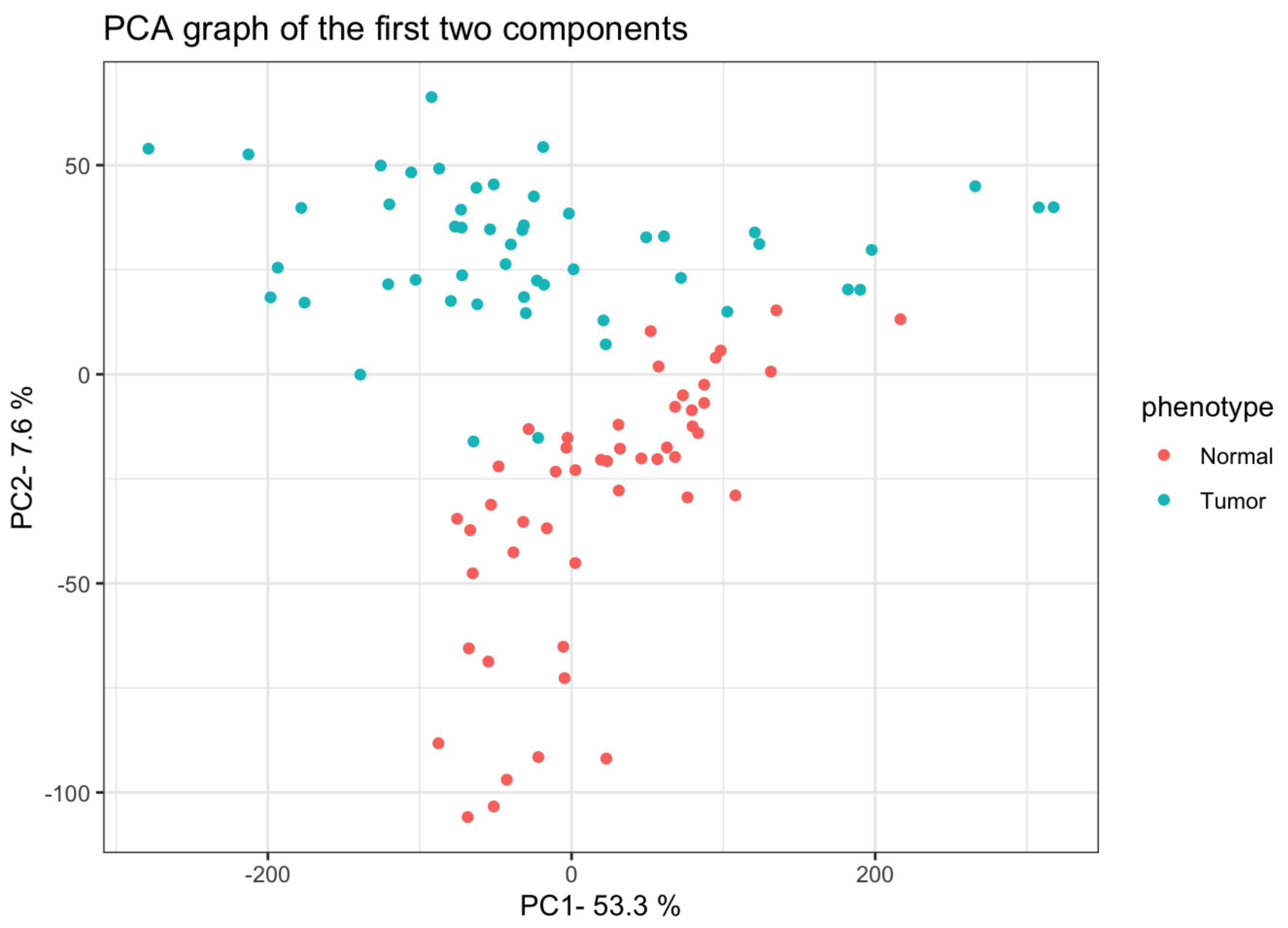
The data is already in the form we want, with genes as variables (columns), and samples as observations (rows). I won't transpose the data before running PCA as I did in problem I.

Problem II.1

Run PCA for dimension reduction on the 100 samples with known labels, and draw these 100 samples in a 2D plot. Do cancer and normal separate from the first two PCs? Would this be sufficient to classify the unknown samples?

z-score normalized data are provided in BRCA_zscore_data.txt. Phenotype data is in BRCA_phenotype.txt.

```
### Your code here
res.pca <- prcomp(zscore, scale = TRUE)
pca.var.per = round(res.pca$sdev^2/sum(res.pca$sdev^2)*100,1)

pca_data <- res.pca$x %>% data.frame() %>%
  mutate(sample = rownames(res.pca$x)) %>%
  merge(pheno)

ggplot(data = pca_data, aes(x=PC1, y=PC2, color = phenotype))+
  geom_point()+
  xlab(paste("PC1-",pca.var.per[1],"%",sep =" "))+
  ylab(paste("PC2-",pca.var.per[2],"%",sep =" "))+
  theme_bw()+
  ggtitle("PCA graph of the first two components")
```



Cancer and normal tissues separate pretty well using the first PCs. This would likely be enough to classify the unknown samples with something like K nearest neighbors or a support vector machine.

Problem II.2

Draw a plot showing the cumulative % variance captured from the top 100 PCs. How many PCs are needed to capture 90% of the variance?

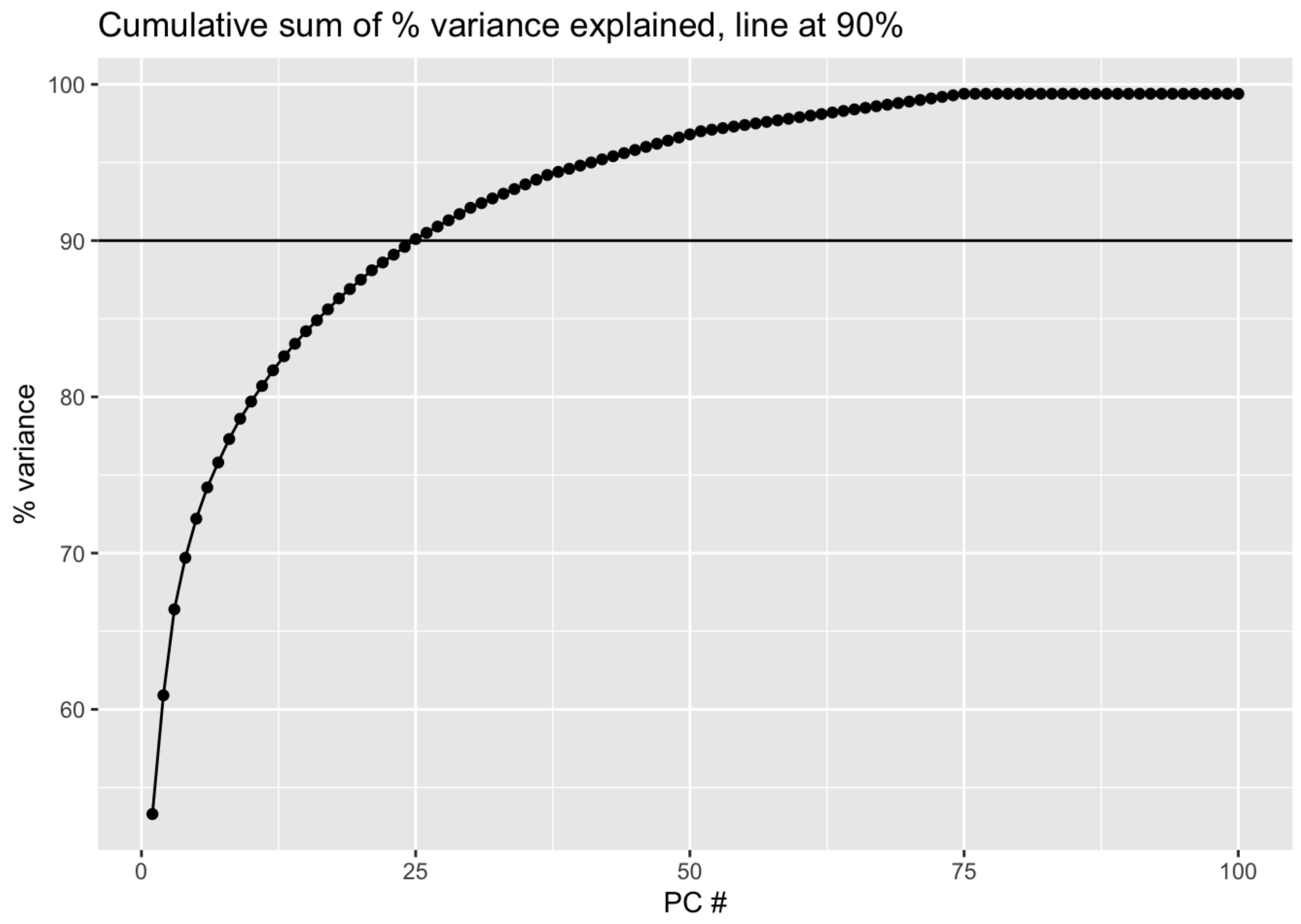
```
which(cumsum(pca.var.per) > 90)[1]
```

```
## [1] 25
```

```
sum(pca.var.per[1:25])
```

```
## [1] 90.1
```

```
toplot <- data.frame(
  PC = 1:100,
  cumsum = cumsum(pca.var.per)
)
toplot %>%
  ggplot(aes (x = PC, y = cumsum)) +
  geom_point() +
  geom_line() +
  geom_hline(yintercept = 90) +
  ggtitle("Cumulative sum of % variance explained, line at 90%") +
  xlab("PC #") +
  ylab("% variance")
```



25 Principal components are needed to capture at least 90% of the variance in gene expression. These 25 PCs capture 90.1%, but if one fewer PC was used this would drop to 89.6%.

Problem II.3

Apply machine learning methods (KNN, logistic regression, Ridge regression, LASSO, ElasticNet, random forest, and support vector machines) on the top 25 PCs of the training data and 5-fold cross validation to classify the samples. caret and MASS already implemented all of the machine learning methods, including cross-validation, so calling each is only one command. In order to get consistent results from different runs, use set.seed().

Set up data

```
train_dat <- pca_data %>%
  select(phenotype, paste("PC", 1:25, sep = ''))
test_dat <- predict(res.pca, unknown) %>%
  data.frame() %>%
  select(paste("PC", 1:25, sep = ''))
test_y <- as.factor(diagnosis$phenotype)

control <- trainControl(method="cv", number=5, savePredictions = TRUE,classProbs =  TRUE
)
metric <- "Accuracy"
lambda <- 10^seq(-3, 3, length = 100)
```

KNN

```
set.seed(123)
knn <- train(phenotype~., data=train_dat, method="knn",
             metric=metric, trControl=control,
             tuneGrid=expand.grid(k=seq(1,10,1))
)
# knn$results
```

Logistic regression

```
set.seed(123)
logistic = train(
  form = as.factor(phenotype) ~ .,
  data = train_dat,
  metric = metric, trControl=control,
  method = "glm",
  family = "binomial"
)
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

```
## Warning: glm.fit: algorithm did not converge
```

```
## Warning: glm.fit: fitted probabilities numerically 0 or 1 occurred
```

Ridge regression Note that in CARET, if alpha = 0 then a ridge regression model is fit, and if alpha = 1 then a lasso model is fit

```
set.seed(123)
ridge <- train(
  as.factor(phenotype) ~., data = train_dat, method = "glmnet",
  metric = metric, trControl=control,
  tuneGrid = expand.grid(alpha = 0, lambda =  c(seq(0.1, 2, by =0.1) ,  seq(2, 5, 0.5) ,
seq(5, 25, 1)))
)
# ridge$bestTune
```

LASSO

```
set.seed(123)
lasso <- train(
  phenotype ~., data = train_dat, method = "glmnet",
  metric = metric, trControl=control,
  tuneGrid = expand.grid(alpha = 1, lambda = c(seq(0.1, 2, by =0.1) ,  seq(2, 5, 0.5) ,
 seq(5, 25, 1)))
)
# lasso$bestTune
```

ElasticNet

```
set.seed(123)
elastic <- train(
  phenotype ~., data = train_dat, method = "glmnet",
  metric = metric, trControl=control,
  tuneGrid = expand.grid(
    alpha = seq(0.1, 1, by =0.1),
    lambda = c(seq(0.1, 2, by =0.1) ,  seq(2, 5, 0.5) , seq(5, 25, 1))
  )
  # tuneLength = 10
)
# elastic$results %>% arrange(-Accuracy)
# elastic$bestTune
```

Random forest

```
set.seed(123)
forest <- train(
  phenotype ~., data = train_dat, method = "rf",
  metric = metric, trControl=control,
  tuneLength = 10
)
# forest$results
# forest$bestTune
```

Support vector machines

```
set.seed(123)
svml <- train(
  phenotype ~., data = train_dat, method = "svmLinear",
  metric = metric, trControl=control, preProcess = c("center","scale"),
  tuneGrid = expand.grid(C = c(0.01, 0.05, 0.1, 0.5, 1, 10))
)
# svml$results
# svml$bestTune
# plot(svml)
```

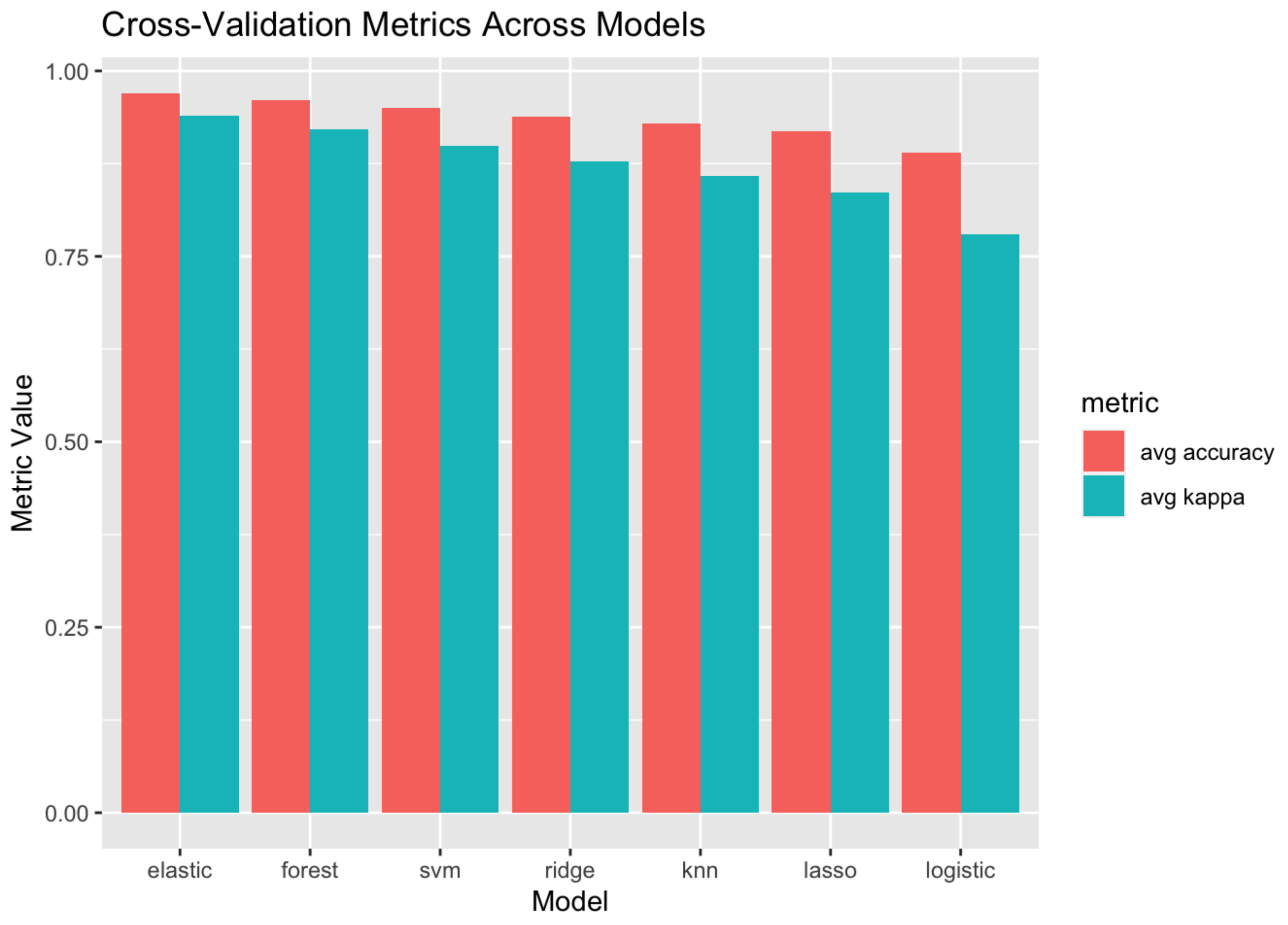
Problem II.4

Summarize the performance of each machine learning method, in terms of accuracy and kappa.

```
models <- list(knn=knn,logistic=logistic,ridge=ridge,
               lasso=lasso,elastic=elastic,forest=forest, svm = svml)
nmodels <- length(models)

acc <- resamples(models) %>% summary(metric = 'Accuracy')
kap <- resamples(models) %>% summary(metric = 'Kappa')

metrics_dat <- data.frame(
  model = rep(acc$statistics %>% data.frame() %>% rownames(), 2),
  metric = c(rep('avg accuracy', nmodels), rep('avg kappa', nmodels)),
  value = c(acc$statistics %>% data.frame() %>% pull(Accuracy.Mean),
            kap$statistics %>% data.frame() %>% pull(Kappa.Mean))
)
metrics_dat
```

```
##       model       metric     value
## 1       knn avg accuracy 0.9294236
## 2  logistic avg accuracy 0.8899499
## 3     ridge avg accuracy 0.9388972
## 4     lasso avg accuracy 0.9188972
## 5   elastic avg accuracy 0.9700000
## 6    forest avg accuracy 0.9604762
## 7       svm avg accuracy 0.9499499
## 8       knn    avg kappa 0.8582988
## 9  logistic    avg kappa 0.7800010
## 10    ridge    avg kappa 0.8778385
## 11    lasso    avg kappa 0.8365887
## 12  elastic    avg kappa 0.9400000
## 13   forest    avg kappa 0.9208219
## 14      svm    avg kappa 0.8995929
```

```
metrics_dat %>%
  ggplot(aes(x = reorder(model, -value), y = value, fill = metric)) +
  geom_bar(stat = 'identity', position = 'dodge') +
  ggtitle("Cross-Validation Metrics Across Models") +
  xlab("Model") +
  ylab("Metric Value")
```



This plot shows the average accuracy and kappa across the five folds of cross validation. Elastic net has the highest average cross validation accuracy and kappa, followed by random forest and knn. SVM, lasso, and logistic regression are all tied for last.

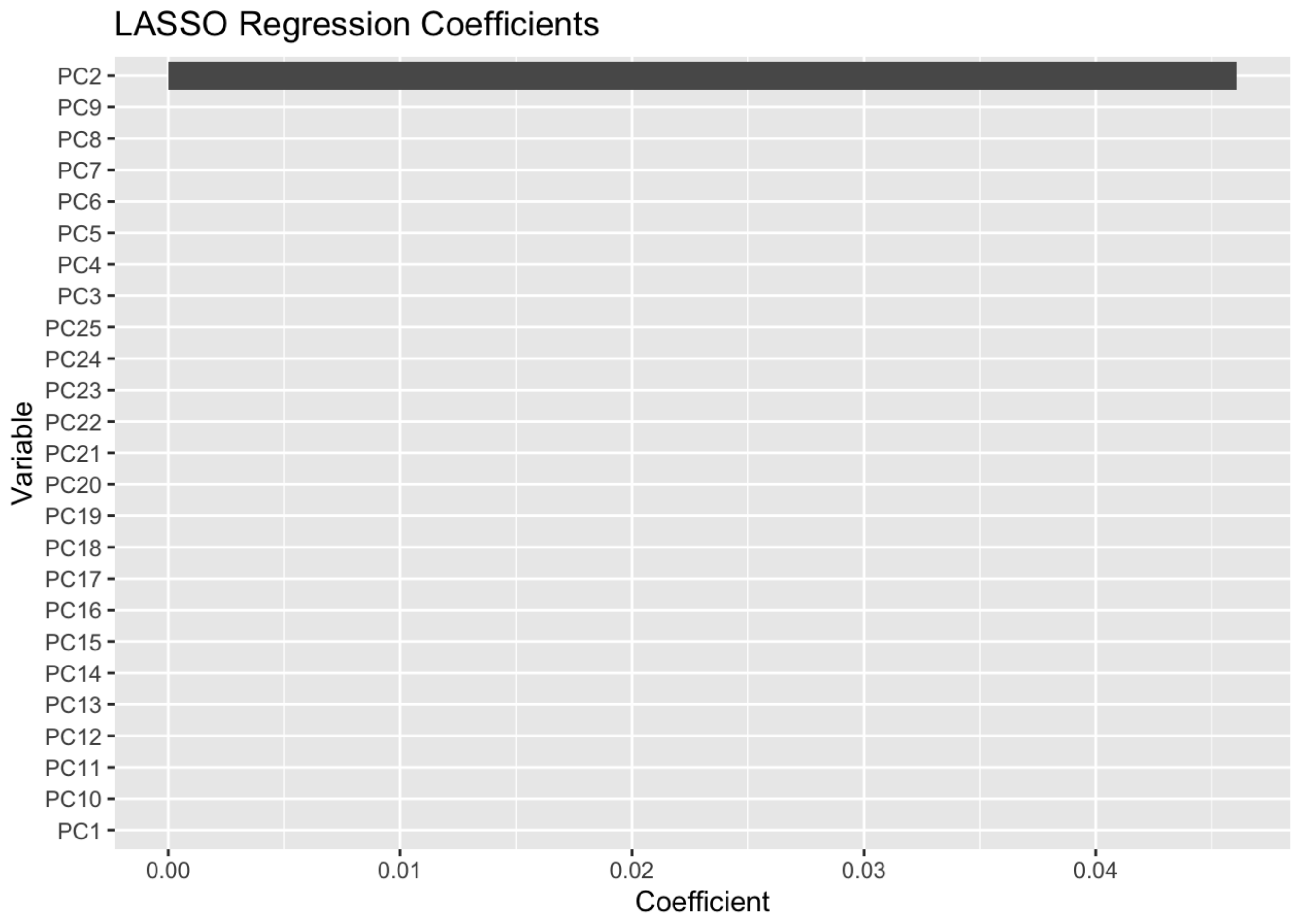
Problem II.5: For Graduate students only

Compare the performance difference between logistic regression, Ridge, LASSO, and ElasticNet. In LASSO, how many PCs have non-zero coefficient? In ElasticNet, Ridge, and LASSO, what is the lamda respectively?

```
coef(lasso$finalModel, lasso$bestTune$lambda) %>% as.matrix() %>% data.frame() %>%
  filter(rownames(.) != '(Intercept)')
```

```
##              X1
## PC1  0.00000000
## PC2  0.04609329
## PC3  0.00000000
## PC4  0.00000000
## PC5  0.00000000
## PC6  0.00000000
## PC7  0.00000000
## PC8  0.00000000
## PC9  0.00000000
## PC10 0.00000000
## PC11 0.00000000
## PC12 0.00000000
## PC13 0.00000000
## PC14 0.00000000
## PC15 0.00000000
## PC16 0.00000000
## PC17 0.00000000
## PC18 0.00000000
## PC19 0.00000000
## PC20 0.00000000
## PC21 0.00000000
## PC22 0.00000000
## PC23 0.00000000
## PC24 0.00000000
## PC25 0.00000000
```

```
coef(lasso$finalModel, lasso$bestTune$lambda) %>% as.matrix() %>% data.frame() %>%
  filter(rownames(.) != '(Intercept)') %>%
  ggplot(aes(y = reorder(rownames(.), X1), x = X1)) +
  geom_bar(stat = 'identity') +
  ylab("Variable") +
  xlab("Coefficient") +
  ggtitle("LASSO Regression Coefficients")
```



With lasso regression, all coefficients are zero except for PC2.

```
elastic$bestTune %>%
  mutate(method = 'elastic') %>%
  rbind(
    lasso$bestTune %>%
      mutate(method = 'lasso')
  ) %>%
  rbind(
    ridge$bestTune %>%
      mutate(method = 'ridge')
  )
```

```
##    alpha lambda  method
## 1    0.1    0.1 elastic
## 2    1.0    0.1   lasso
## 23   0.0    3.5   ridge
```

The optimal lambda is 0.1 for lasso, 3.5 ridge regression, and 0.2 for elastic net.

Problem II.6

Use the PCA projections in Q1 to obtain the first 25 PCs of the 20 unknown samples. Use one method that performs well in Q4 to make predictions. Caret already used the hyper-parameters learned from cross-validation to train the parameters of each method on the full 100 training data. You just need to call this method to make the predictions.

Expression data for the 20 unknown samples are provided in unknown_samples.txt.

```
elasticPredict <- predict(elastic, newdata = test_dat)
elasticPredict
```

```
##  [1] Tumor  Normal Tumor  Normal Tumor  Normal Tumor  Normal Tumor  Normal
## [11] Tumor  Normal Tumor  Normal Tumor  Normal Tumor  Normal Tumor  Tumor 
## Levels: Normal Tumor
```

Because elastic net had the best cross validation metrics, I used this model for predictions.

Problem II.7: For Graduate students only

Can you find out the top 3 genes that are most important in this prediction method in Q6? Do they have some known cancer relevance?

```
gene_coefficients <- abs(res.pca$rotation[,1:25]) %*%
    abs(coef(elastic$finalModel, elastic$bestTune$lambda)[2:26,])
gene_coefficients %>% data.frame() %>%
  arrange(-abs(.)) %>%
  head(3)
```

```
##                  .
## PVALB  0.001927499
## NVL    0.001891869
## DNAJA4 0.001863675
```

The three genes with the largest absolute value gene-specific coefficients are PVALB, NVL, and DNAJA4. All three of these genes have known cancer relevence.

PVALB: The study published in Breast Cancer Management, Vol 7, No 2, “Gene expression profiling of triple-negative breast tumors with different expression of secreted protein acidic and cysteine rich (SPARC)” (https://www.futuremedicine.com/doi/full/10.2217/bmt-2017-0019) reports PVALB as being associated with breast cancer.

NVL: The study “KIF11 Functions as an Oncogene and Is Associated with Poor Outcomes from Breast Cancer” (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6639218/) reports NVL as highly upregulated among individuals with breast cancer, and notes that its function remains unknown in breast cancer. This paper suggests that NVL may be a diagnostic in breast cancer, as individuals with high expressions of NVL showed overall better survival compared to those with lower expression.

DNAJA4: An entire study was dedicated to the relationship between this gene family, heat shock proteins, and breast cancer: “Comprehensive transcriptomic analysis of heat shock proteins in the molecular subtypes of human breast cancer” (https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6022707/). They report that DNAJA4 was upregulated among cancer samples.

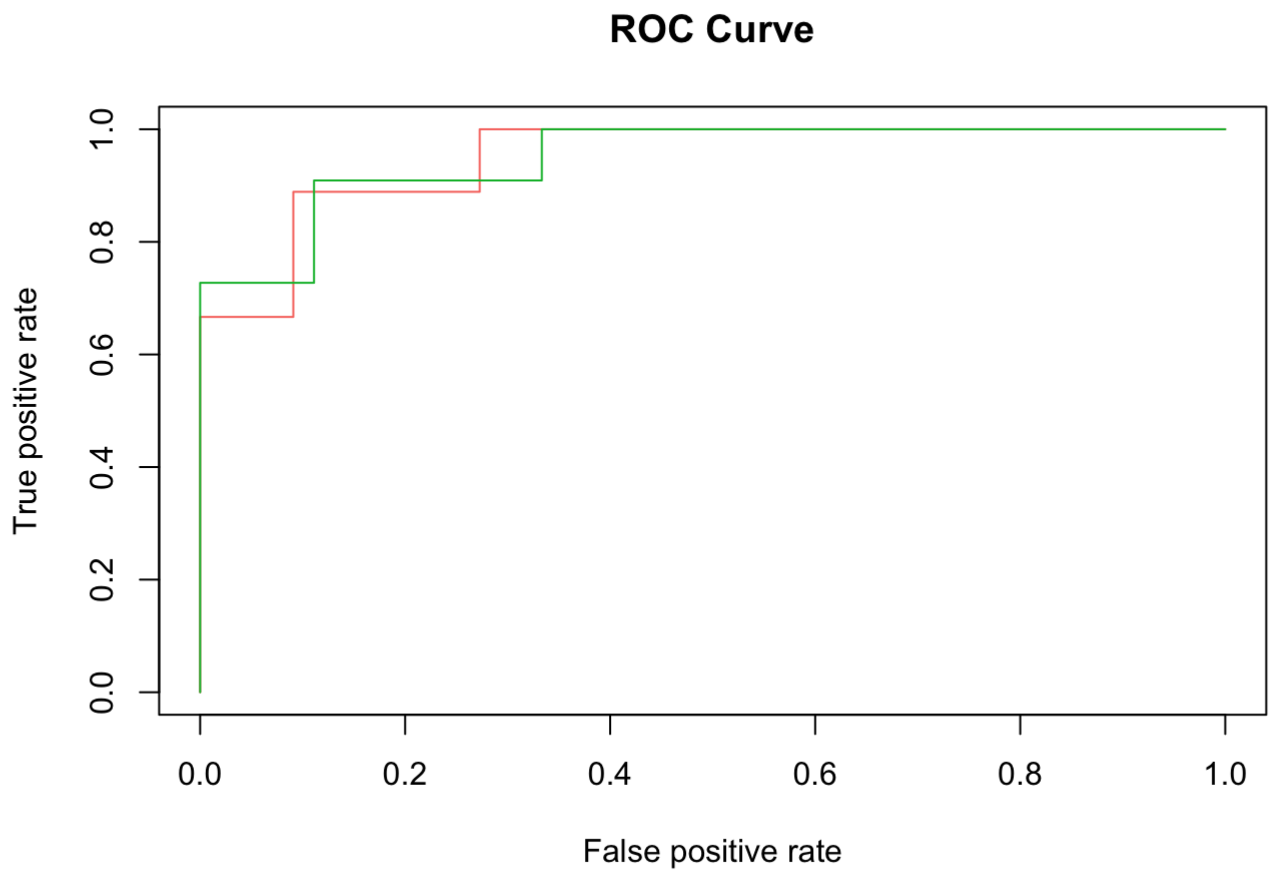
Problem II.8

Suppose a pathologist later made diagnosis on the 20 unknown samples (load the diagnosis.txt file). Based on this gold standard, draw an ROC curve of your predictions in Q6. What is the prediction AUC?

```
### Your code here
elasticPredict <- predict(elastic, newdata = test_dat)
confusionMatrix(elasticPredict, test_y)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction Normal Tumor
##     Normal      8     1
##     Tumor       1    10
##
##                Accuracy : 0.9
##                  95% CI : (0.683, 0.9877)
##     No Information Rate : 0.55
##     P-Value [Acc > NIR] : 0.0009274
##
##                   Kappa : 0.798
##
##  Mcnemar's Test P-Value : 1.0000000
##
##             Sensitivity : 0.8889
##             Specificity : 0.9091
##          Pos Pred Value : 0.8889
##          Neg Pred Value : 0.9091
##              Prevalence : 0.4500
##          Detection Rate : 0.4000
##    Detection Prevalence : 0.4500
##       Balanced Accuracy : 0.8990
##
##        'Positive' Class : Normal
##
```

```r
elasticPredictProb <- predict(elastic, newdata = test_dat, type = 'prob')

pretty_colours <- c("#F8766D","#00BA38")
classes <- levels(test_y)
# For each class
for (i in 1:length(classes)) {
  # Define which observations belong to class[i]
  true_values <- ifelse(test_y==classes[i],1,0)
  # Assess the performance of classifier for class[i]
  pred <- prediction(elasticPredictProb[,i],true_values)
  perf <- performance(pred, "tpr", "fpr")
  if (i==1)
  {
     plot(perf,main="ROC Curve",col=pretty_colours[i])
  }
  else
  {
     plot(perf,main="ROC Curve",col=pretty_colours[i],add=TRUE)
  }
  # Calculate the AUC and print it to screen
  auc.perf <- performance(pred, measure = "auc")
  print(auc.perf@y.values)
}
```



```
## [[1]]
## [1] 0.9494949
##
## [[1]]
## [1] 0.9494949
```

The ROC AUC is 0.9494949, which is very good.