Gene expression cancer RNA-Seq

STAT 437 Project 2 Data Set Analysis

Shawn Petersen(011691731), Laura Pelayo(011750956)

Washington State University

May 06, 2022

Abstract

This research paper studies cancer prediction from gene expression data using PCA and SPCA methods, in addition a data set containing email spam was analyzed with PCA methods

1. Introduction

2. Data Analysis Approach

Two datasets were used in this project 2 analysis.

- The first data set was the Gene Expression Cancer RNA-Seq Data Set which is hosted by the UC Irvine Machine Learning Repository. This data set consists of 801 cancer patients, each exhibiting one of five types of cancer. For each patient, 20,531 gene expressions are recorded as real-valued features.
- The second data set used in this analysis was the Spam Data Set, which is also hosted by the UC Irvine Machine Learning Repository. This data set has 4,601 emails which are described by 57 real-valued features, each of which is labeled (tagged) spam or non-spam.

PCA was used as a tool for finding patterns in high-dimensional data, such as the data set provided for this project on the cancer gene and email spam.

3. Data Cleaning

For this project, genes that contained less than 300 values of 0's were selected for the data set. From the remaining set, 1000 genes were randomly selected for a more manageable data set to analyze with the various techniques. The data was also scaled to minimize the effects that relatively large values have on clustering metrics.

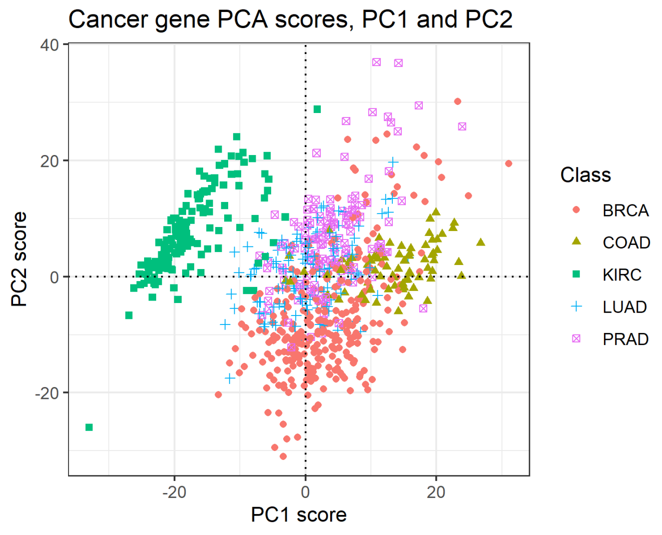
4. Results Analysis

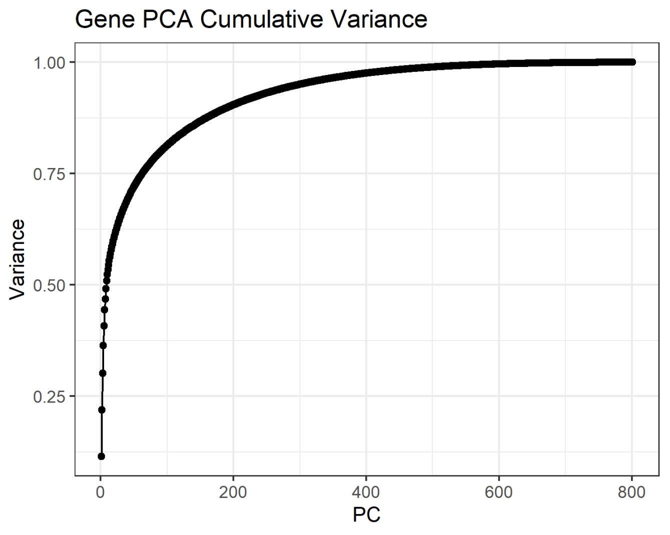
4.1 PCA & SPCA for Patterns in Cancer Data

In this section, PCA and SPCA are applied to the sub-data set to analyze patterns in cancer genes. The resulting data set was used for the following analyses with PCA and SPCA.

PCA was applied to the data resulting in 801 principal components in total, one per sample. The variance was calculated for each of the principal components, ordered by eigenvalues. A plot of the first two principal component scores was created. Looking at the visualization of the scores for the first two components class clusters we can clearly seen, which KIRC and COAD appear to be the most distinct, with the other 3 classes, BRCA, LUAD, and PRAD, having a significant amount of overlap. Both of these plots can be seen below in **Figure 1** and **Figure 2**:

Figure 3 is a graph of the PCA eigenvalues, which is telling how much variance there is in the data in that direction for each PCA. The plot shows the first 10 PCA dimensions, , the first two PCA's have a total of 21.9% of the variation





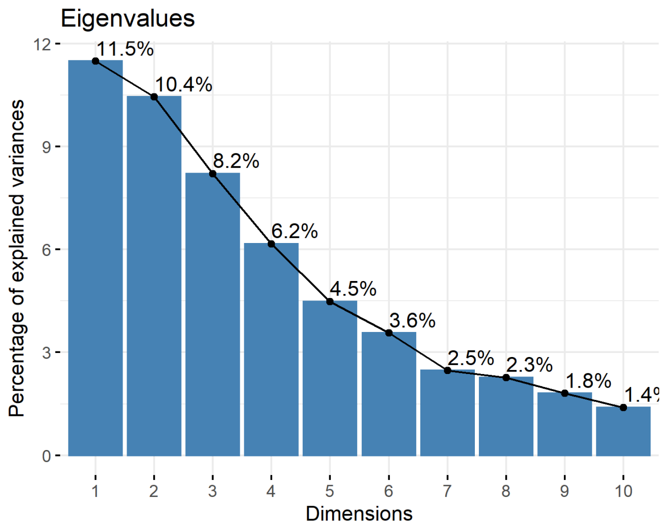
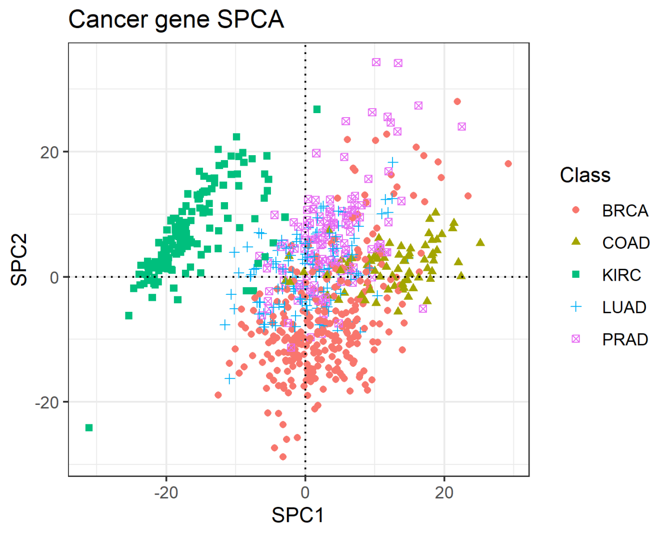


Figure 1 (left) - **Figure 3** (bottom-center) - **Figure 2** (right)

The first 8 principal components (of 801) were calculated at $\approx 49\%$ of the total explained variance. The graph shows a significant amount of the information that characterizes the cancer gene data is present in relatively few principal components. Observation:

- The cancer gene score plot visualized clusters for each of the cancer classes. KIRC and COAD gene samples were the most distinct clusters of all the gene classes, having little overlap with other classes.
- The other three classes, BRCA, LUAD, and PRAD overlap heavily
- The class with the largest dispersion was BRCA which spanned the largest range across positive and negative PC2 scores.

SPCA was also calculated the elasticnet::spca function for comparison with the PCA results. For this calculatoin, the adjusted partial variance explained as well as the adjusted cumulative partial variance explained were computed and plotted. A score plot of the first 2 sparse principal components was also created, similar to that which was constructed for the PCA section. Both of these plots can be seen below in **Figure 1** and **Figure 2**:



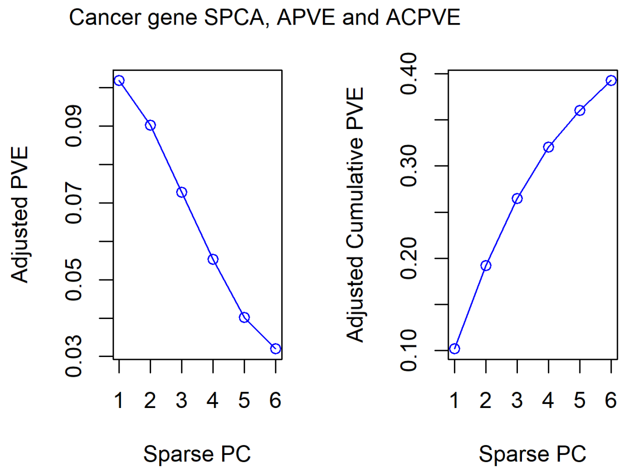


Figure 4 (left) - **Figure 5** (right)

Comparing the results between PCA and SPCA, we see that even though the first 6 sparse PCs account for less $\approx 40\%$ of the variance explained, which was less than that of classical PCA at $% for the same number of components. The score plots are still very similar for the PCA and SPCA plots. It can be seen clearly clusters of the cancer gean class with KIRC samples being the most distinctly different.

Please also investigate and address the following when doing data analysis:

(1.a) Are there genes for which linear combinations of their expressions explain a significant proportion of the variation of gene expressions in the data set? Note that each gene corresponds to a feature, and a principal component based on data version is a linear combination of the expression measurements for several genes.

- There were no genes among the top 10 loadings that are common between the first three principal components in PCA. To determine this, the top 10 absolute values of the loadings for PC 1, 2, and 3, were extracted from the PCA calculation. We then did an intersection between PCA (1, 2), (1, 3), and (2, 3), to see if the same genes were present in the top 10 loadings for each principal component. No common genes were found among the first 3 PC for PCA (See Appendix for R code)

(1.b) Ideally, a type of cancer should have its "signature," i.e., a pattern in the gene expressions that is specific to this cancer type. From the "labels.csv," you will know which expression measurements belong to which cancer type. Identify the signature of each cancer type (if any) and visualize it. For this, you need to be creative and should try both PCA and Sparse PCA.

To identify signatures for each class, the scores of the first two components of PCA were analyzed mathematically. Results of the cancer genes scores:

- BRCA score mean: PC1=3.068 PC2=-7.3087
- COAD score mean: PC1=13.275 PC2=1.813
- KIRC score mean: PC1=-17.461 PC2=7.503
- LAUD score mean: PC1=0.422 PC2=0.663
- PRAD score mean: PC1=3.925 PC2=6.339

(1.c) There are 5 cancer types. Would 5 principal components, obtained either from PCA or Sparse PCA, explain a dominant proportion of variability in the data set, and serve as the signatures of the 5 cancer types? Note that the same set of genes were measured for each cancer type.

The first 5 PCs from PCA accounted for $\approx 40\%$ of the cumulative variance explained and the first 5 SPCs from SPCA accounted for $\approx 35\%$ of the cumulative variance explained. Both methods were similar in results of Eigenvalues and KIRC samples having the best distinct cluster.

4.1 PCA for Spam Detector

For the spam experiment, PCA was applied to email data to determine whether emails can be reliably classified as spam or non-spam. The first step of this section was to load the data and remove the testid column which contained sample labels predicted during a previous analysis. The data was checked for missing values,which none were found.

In **Figure 6** shows the highly correlated features in the form of a corrlation matrix.

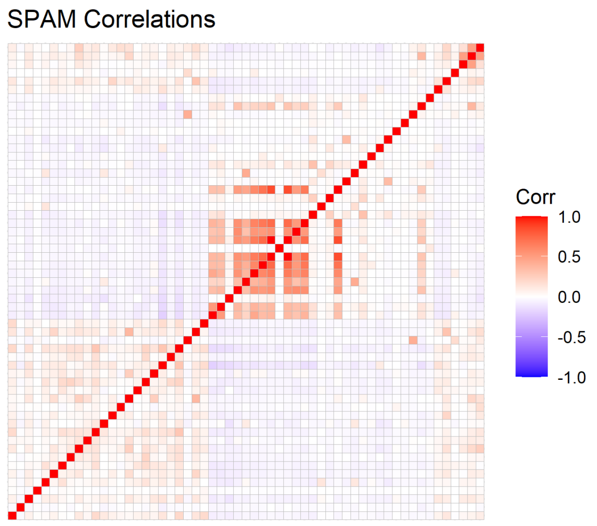
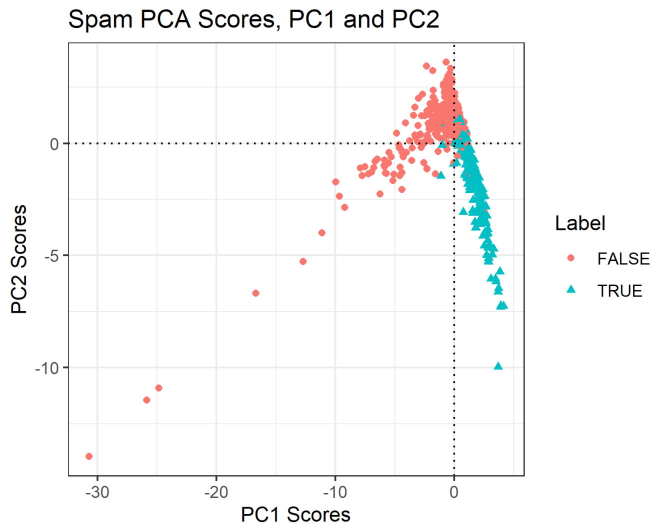


Figure 6 (left)

The cumulative variance of each principal component was computed and plotted, seen in figure **Figure 7** and **Figure 8**. The first two principal components only accounted for 18.5% of the spam variance. The score plot of these principal components shows a clear pattern where each class label, spam vs. non-spam. Both were clustered and visually sepeated. The samples with TRUE PC1 scores and FALSE PC2 scores were clearly labeled as spam.



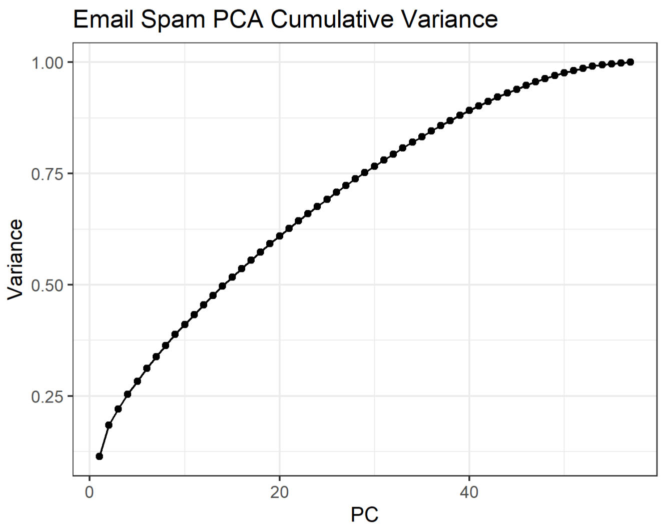


Figure 7 (left) - **Figure 8** (right)

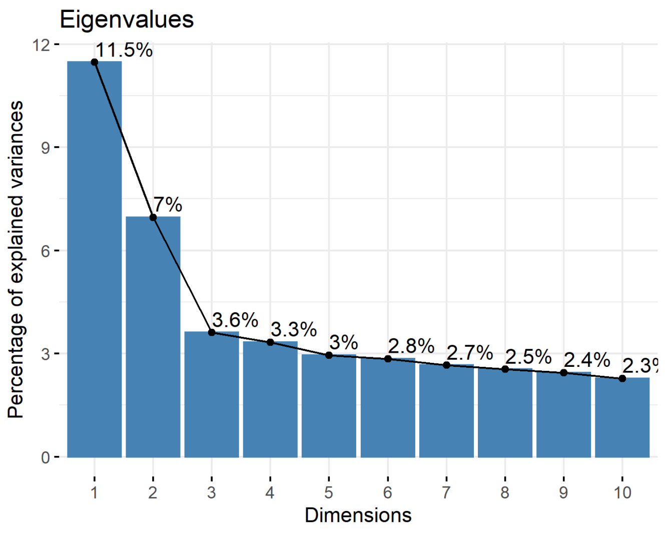


Figure 9 (center)

Figure 9 is a graph of the PCA eigenvalues, which is telling how much variance there is in the data in that direction for each PCA. The plot shows the first 10 PCA dimensions, the first two PCA's have a total of 18.7% of the variation

Conclusion

High-dimensional data is becoming increasingly common in many fields. Often, the presentation of high dimensional data can be inadequate. We find a redundancy of information and it is not always easy to process big data effectively. A common approach is to apply a dimension reduction technique that will model our data in a lower dimensional setting, that captures the core importance of our data. Principle component analysis (PCA) tries to explain the variance found in our data. PCA helps us remove noise from our data and we are also able to efficiently create consistent 2-D visualizations. PCA is an important place to start with any unsupervised learning, but not all datasets will work well with PCA. With our cancer dataset, PCA and SPCA were not very helpful. KIRC and COAD gene samples had the clearest clusters compared to the other three classes, BRCA, LUAD, and PRAD. Looking at our eigenvalues chart **figure 3**, it is easy to understand why. The variance of our data is evenly spread and there isn't specifically one or two components that explain the majority of how our data is behaving. On the other hand, for our SPAM dataset, there is some overlap with TRUE and FALSE, but overall, there is a clear distinction between an email that is classified as spam or not when we look at our graph **figure 7**. If we also look at our eigenvalues chart **figure 9**, we can see that the majority of variance is explained with the first two principal components.

###——————————————————————————————————————

5. Appendix

5.1 Import Libraries

The first chunk of code imports the required libraries

```
library(dplyr)
library(tidyverse)
library(viridis)
library(plotly)
library(ggridges)
library(tibble)
library(cluster)
library(ggdendro)
library(ggcorrplot)
library(latex2exp)
library(ISLR)
library(sparsepca)
library(elasticnet)
library(reshape2)
library(ggbiplot)
library(ggplot2)
library(corrplot)
```

5.2 Data File load

The first step of the analysis is to load the data and label files into R for future processing.

```
# Load data sets
df_data = read.csv("data/data.csv")
df_labels = read.csv("data/labels.csv")
```

5.3 Task (2.a) - Analysis of gene expression data

Please use `set.seed(123)` for random sampling via the command `sample`.

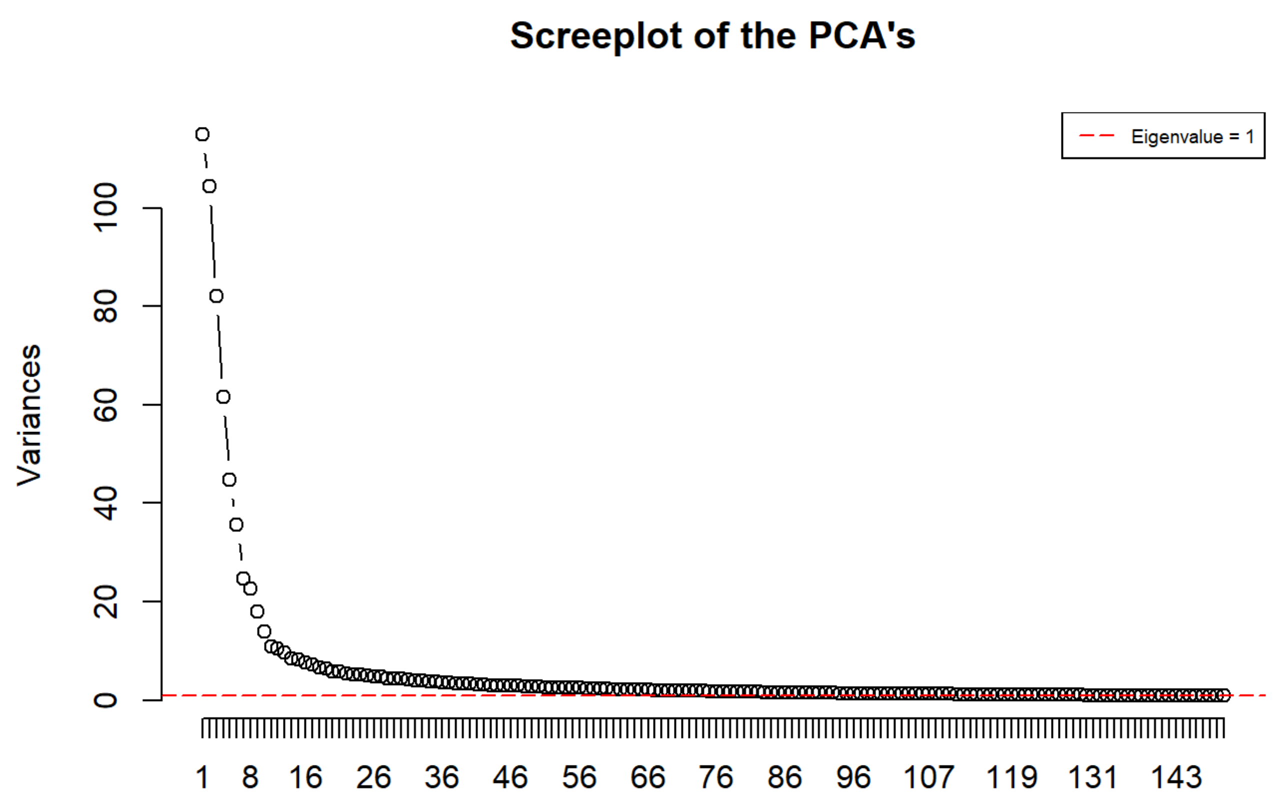
- Filter out genes (from “data.csv”) whose expressions are zero for at least 300 subjects, and save the filtered data as R object “gexp2.”
- Use the command `sample` to randomly select 1000 genes and their expressions from “gexp2,” and save the resulting data as R object “gexp3.”
- Use the command `scale` to standardize the gene expressions for each gene in “gexp3.” Save the standardized data as R object “stdgexpProj2.”

You will analyze the standardized data.

```
library(factoextra)

set.seed(123)

# Filter columns with <=300 zero's
gexp2 = df_data[colSums(df_data == 0) <= 300]

sample.column.list = gexp2[,1]

# Random sample 1000 genes(columns)
gexp3 = gexp2[sample(ncol(gexp2), size = 1000), drop = F]

gexp3 = cbind(sample.column.list, gexp3)
colnames(gexp3)[1] = "X"

# Scale columns to have a std=1 and mean=0
stdgexpProj2 = gexp3 %>% mutate_at(c(2:1001), ~(scale(.) %>% as.vector))

# PCA coveriance
pca.cov = cov(stdgexpProj2[,2:1001])
pca.eigen = eigen(pca.cov)
str(pca.eigen)
```

```
## List of 2
##  $ values : num [1:1000] 114.9 104.4 82.1 61.7 44.8 ...
##  $ vectors: num [1:1000, 1:1000] -0.051993 -0.065404 0.048076 -0.013043 -0.000298
...
##  - attr(*, "class")= chr "eigen"
```

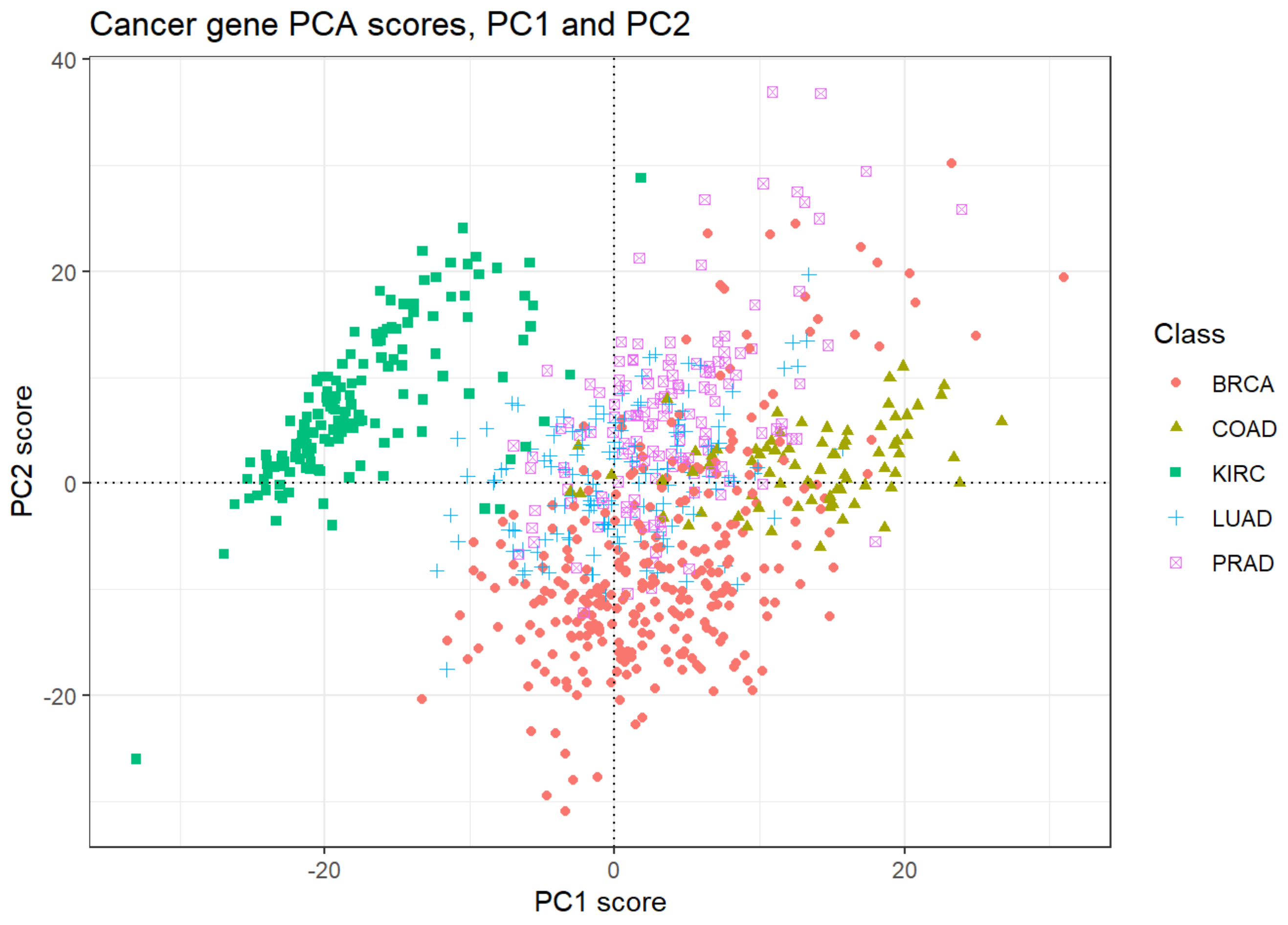
```
#PVE
pca.pve = pca.eigen$values / sum(pca.eigen$values)

# Calculate PCA
pca1 = prcomp(stdgexpProj2[,2:1001])

eig.val = get_eigenvalue(pca1)


{
screeplot(pca1, type = "l", npcs = 150, main = "Screeplot of the PCA's")
abline(h = 1, col="red", lty=5)
legend("topright", legend=c("Eigenvalue = 1"),
       col=c("red"), lty=5, cex=0.6)
}
```



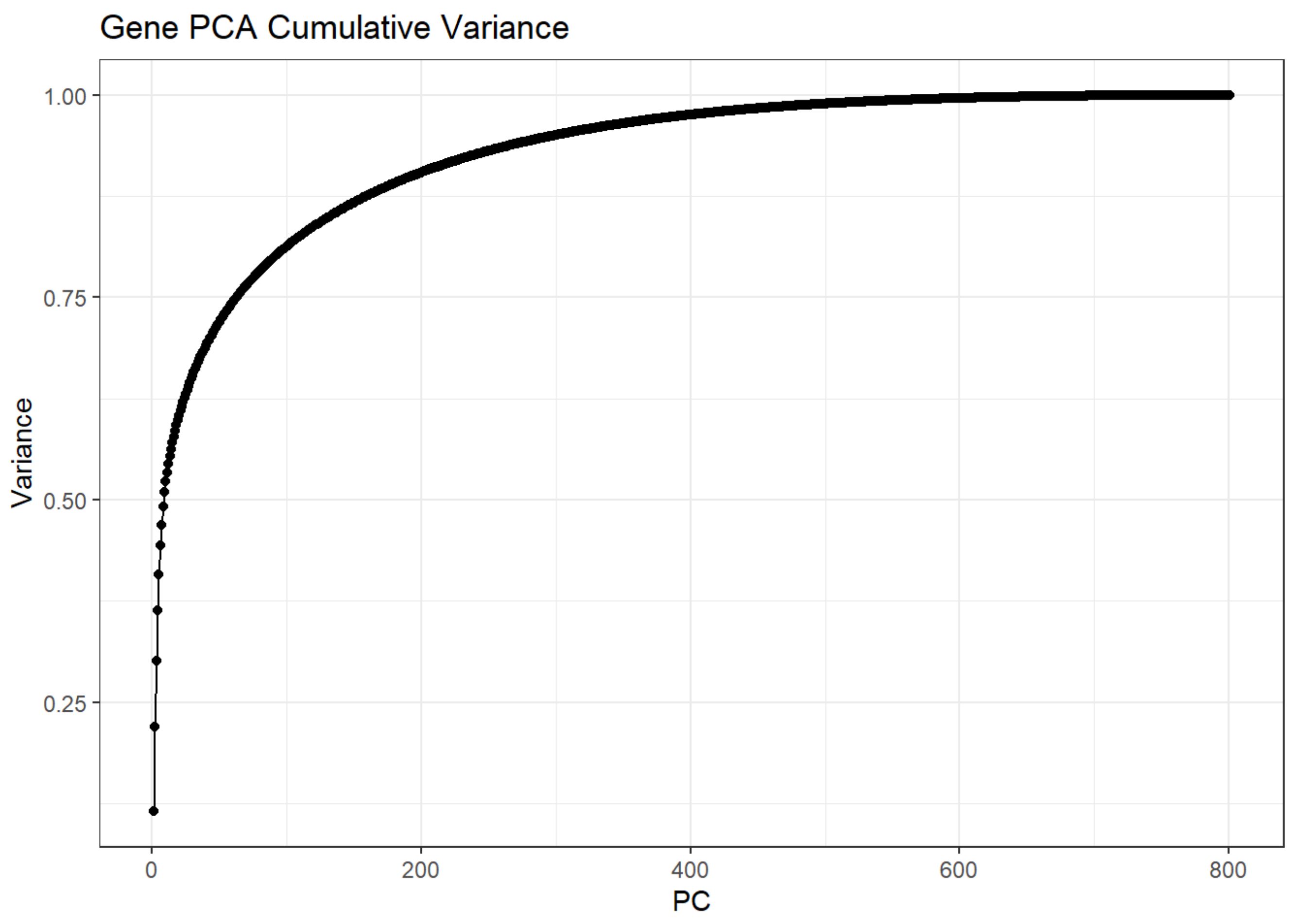
```
# Plot eigenvalues
{
fviz_eig(pca1,
         addlabels = TRUE,
         ncp=10,
         main = "Eigenvalues")
ggsave("figures/pca_eigenvalues.png", device="png", width=5, height=4)
}


# Plot PCA box plot
#fviz_pca_ind(pca1,
#             col.ind = "cos2", # Color by the quality of representation
#             gradient.cols = c("#00AFBB", "#E7B800", "#FC4E07"),
#             repel = TRUE     # Avoid text overlapping
#             )


# Number of PCA columns
ncol(pca1$x)
```

```
## [1] 801
```

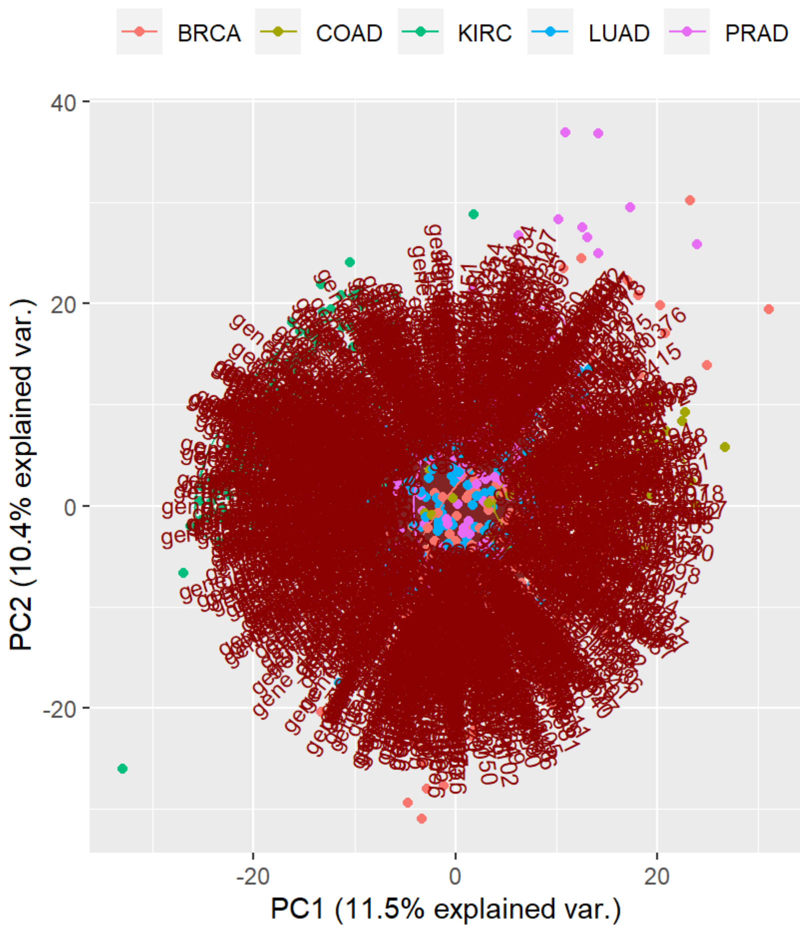
```
# Plotting PCA scores
pca.scores = as.data.frame(pca1$x[, 1:2])
pca.scores$Class = df_labels$Class

# Plot final graph of PCA
ggplot(pca.scores, aes(PC1, PC2, color = Class, shape = Class)) +
    geom_point() + theme_bw() +
   labs(title = "Cancer gene PCA scores, PC1 and PC2", x = "PC1 score", y = "PC2 sco
         re") + geom_hline(yintercept = 0, linetype = "dotted") + geom_vline(xinterc
         ept = 0, linetype = "dotted")
```



```
# Save plot to figures
ggsave("figures/pca_geneplot.png", device="png", width=5, height=4)



# Computing the cumulative variance for each subsequent PC
cum.sum = cumsum((nrow(stdgexpProj2[,2:1001]) - 1) * ((pca1$sdev)^2) / sum(stdgexpPr
         oj2[,2:1001]^2))
cum.data = data.frame("PC"=1:length(cum.sum), "CumulativeVariance"=cum.sum)

ggplot(cum.data, aes(x=PC, y=CumulativeVariance)) +
   geom_point() + geom_line() + theme_bw() +
   labs(title="Gene PCA Cumulative Variance", x="PC", y="Variance")
```



```
ggsave("figures/pca_cumcurve.png", device="png", width=5, height=4)



# Plotting standard biplot (testing)
{
g = ggbiplot(pca1,
         obs.scale=1,
         var.scale=1,
         groups=df_labels$Class,
         ellipse=TRUE,
         circle=TRUE)

g = g + scale_color_discrete(name = '')
g = g + theme(legend.direction = 'horizontal',
               legend.position = 'top')
g
ggsave("figures/pca_biplot.png", device="png", width=5, height=4)
}
g
```

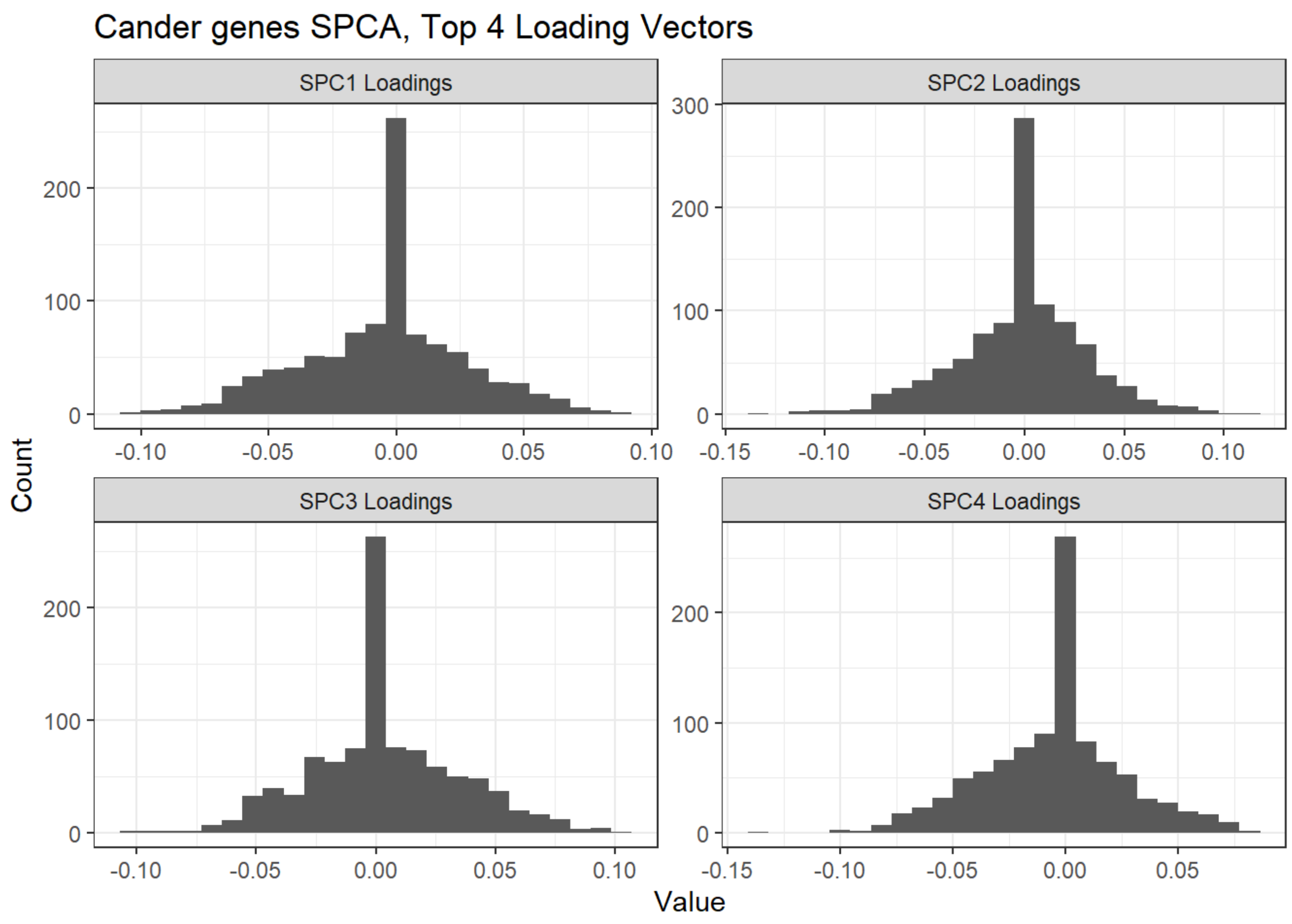


### Task (2.b) Apply Sparse PCA, provide visualizations of low-dimensional structures, and report your findings. Note that you need to use “labels.csv” for the task of discovering patterns. Your laptop may not have sufficient computational power to implement Sparse PCA with many principal components. So, please pick a value for the sparsity controlling parameter and a value for the number of principal components to be computed that suit your computational capabilities.

```
# Setting the seed for the chunk
set.seed(123)

{
ptm = proc.time()
# Applying PCA to the data
spca = elasticnet::spca(stdgexpProj2[,2:1001], K=6, type="predictor", sparse="penalt
         y", lambda=1e-6, para=rep(1e-6, 6), max.iter=200, eps.conv=1e-3)

# Stop the clock
proc.time() - ptm
}
```

```
##    user  system elapsed
##  108.91   32.11  142.61
```

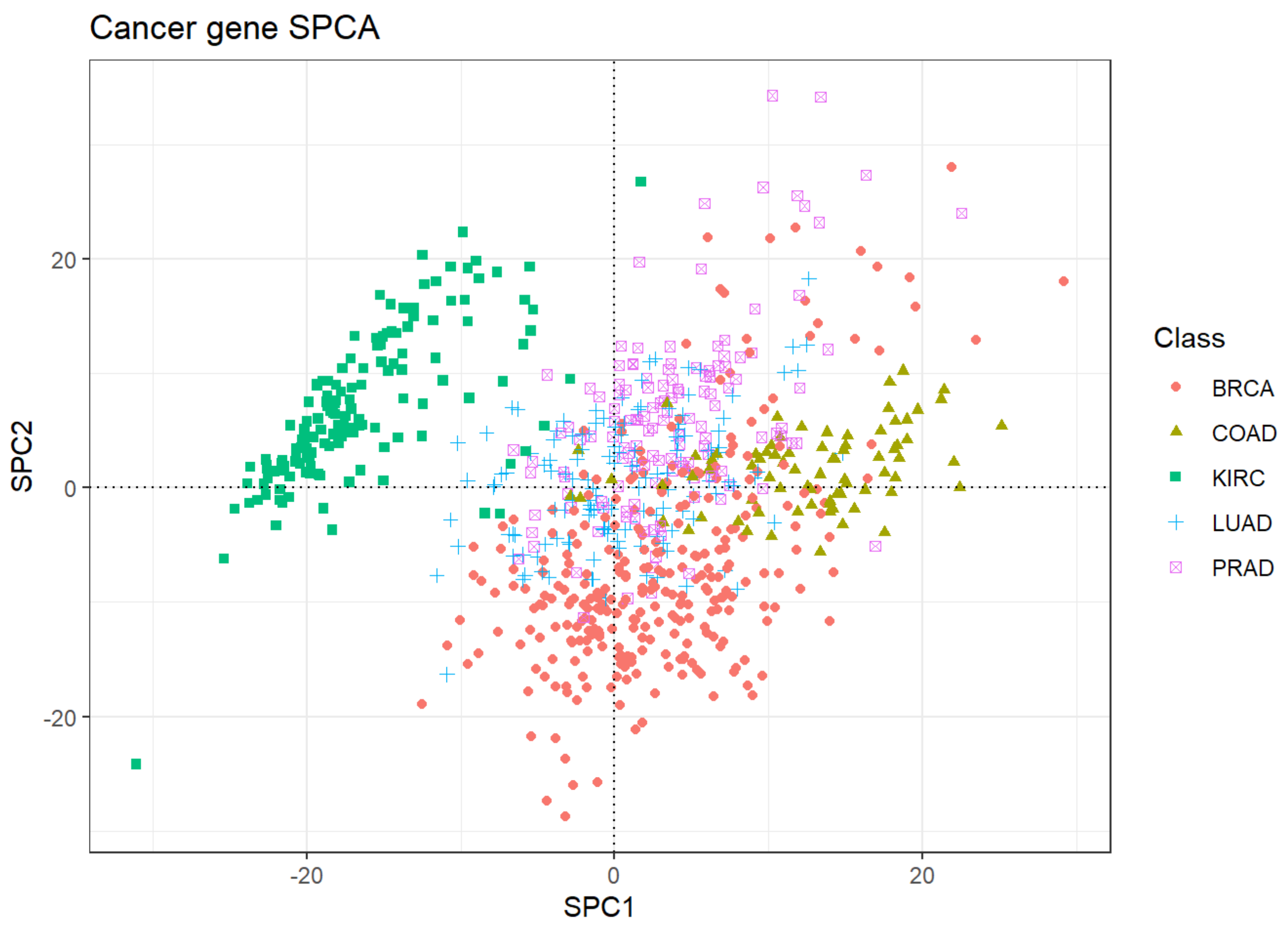
```
# Inspecting first 4 loading vectors
lvs = spca$loadings[,1:4]
colnames(lvs) = paste("SPC", 1:4, sep="")
colnames(lvs) = paste(colnames(lvs), "Loadings", sep=" ")

lvsMelt = melt(lvs)

ggplot(lvsMelt, aes(x=value)) +
   geom_histogram(bins=25) +
   facet_wrap(~Var2, scales="free") +
   theme_bw() +
   labs(title="Cander genes SPCA, Top 4 Loading Vectors", x="Value", y="Count")
```



```
# Adjusted PVE and CPVE
spca.pve = spca$pev



{
png(filename="figures/spca_cumcurve.png", width=5, height=4, units="in", res=300)
par(mfrow=c(1, 2))
plot(spca.pve, type="o", ylab="Adjusted PVE", xlab="Sparse PC", col="blue")
plot(cumsum(spca.pve), type="o", ylab="Adjusted Cumulative PVE", xlab="Sparse PC", c
         ol="blue")
mtext("Cancer gene SPCA, APVE and ACPVE", side=3, line=1.5, adj=1.7)
dev.off()
}
```

```
## png
##   2
```

```
# SPCA plots
slv2 = spca$loadings[,1:2]
ssv2 = as.matrix(stdgexpProj2[2:1001]) %*% slv2

test = as.data.frame(ssv2)
test$Class = df_labels$Class

ggplot(test, aes(PC1, PC2, color=Class, shape=Class)) +
   geom_point() +
   theme_bw() +
   labs(title="Cancer gene SPCA", x="SPC1", y="SPC2") +
   geom_hline(yintercept=0, linetype="dotted") +
   geom_vline(xintercept=0, linetype="dotted")
```



```
ggsave("figures/spca_geneplot.png", device="png", width=5, height=4)
```

Task B Analysis of SPAM email data set

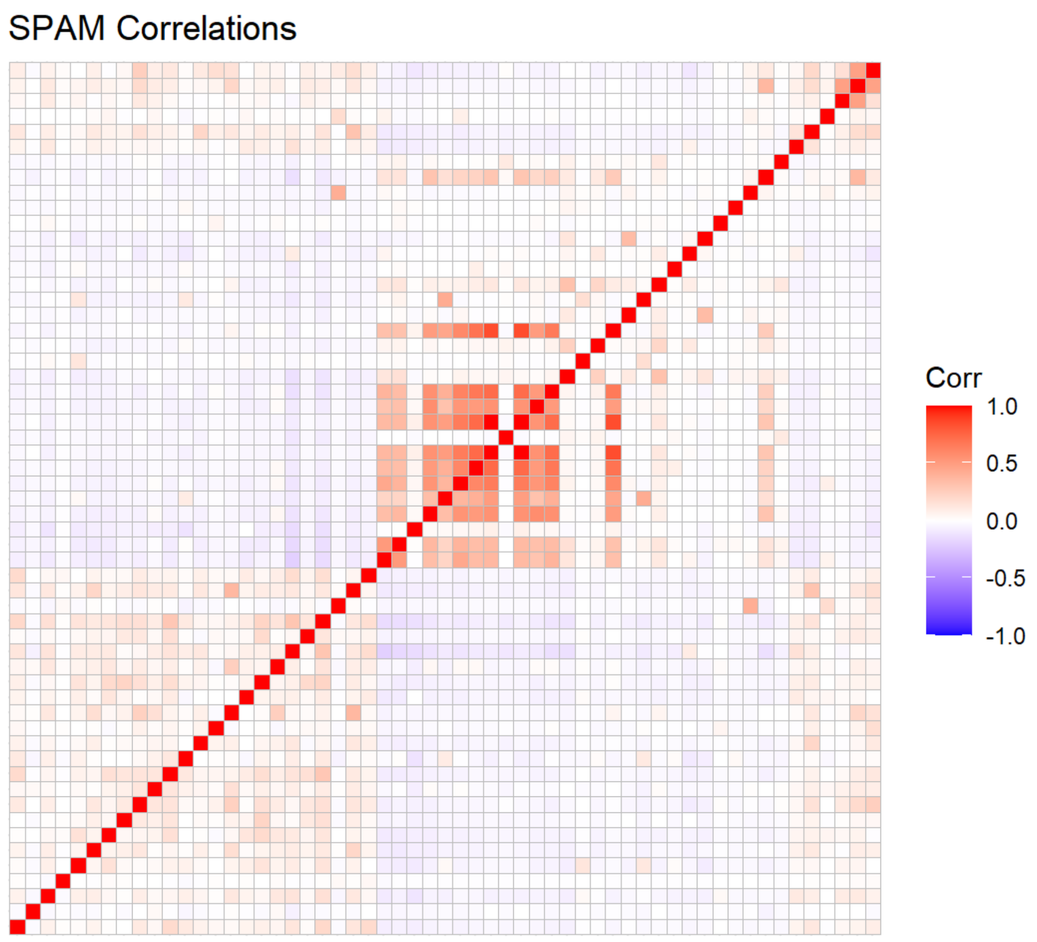
```
# Loading the data
spam.file = read.csv("data/SPAM.csv")

# Removing the `testid` column
spam.file = spam.file %>% select(-testid)

# Checking for missing values
sapply(spam.file, function(x) sum(is.na(x)))
```

```
##       spam       make    address        all        X3d        our       over 
##          0          0          0          0          0          0          0 
##     remove   internet      order       mail    receive       will     people 
##          0          0          0          0          0          0          0 
##     report  addresses       free   business      email        you     credit 
##          0          0          0          0          0          0          0 
##       your       font       X000      money         hp        hpl     george 
##          0          0          0          0          0          0          0 
##       X650        lab       labs     telnet       X857       data       X415 
##          0          0          0          0          0          0          0 
##        X85 technology      X1999      parts         pm     direct         cs 
##          0          0          0          0          0          0          0 
##    meeting   original    project         re        edu      table conference 
##          0          0          0          0          0          0          0 
##        ch.      ch..1      ch..2      ch..3      ch..4      ch..5    crl.ave 
##          0          0          0          0          0          0          0 
##   crl.long    crl.tot 
##          0          0
```

```
# Splitting data from labels
spam.data = spam.file[2:length(spam.file)]
spam.labels = spam.file[1]

# Checking for highly correlated values
spam.corr = cor(spam.data)
dim(spam.corr)[2]
```

```
## [1] 57
```

```
ggcorrplot(spam.corr) +
   theme(axis.text.x=element_blank(), axis.text.y=element_blank()) +
   labs(title="SPAM Correlations")
```

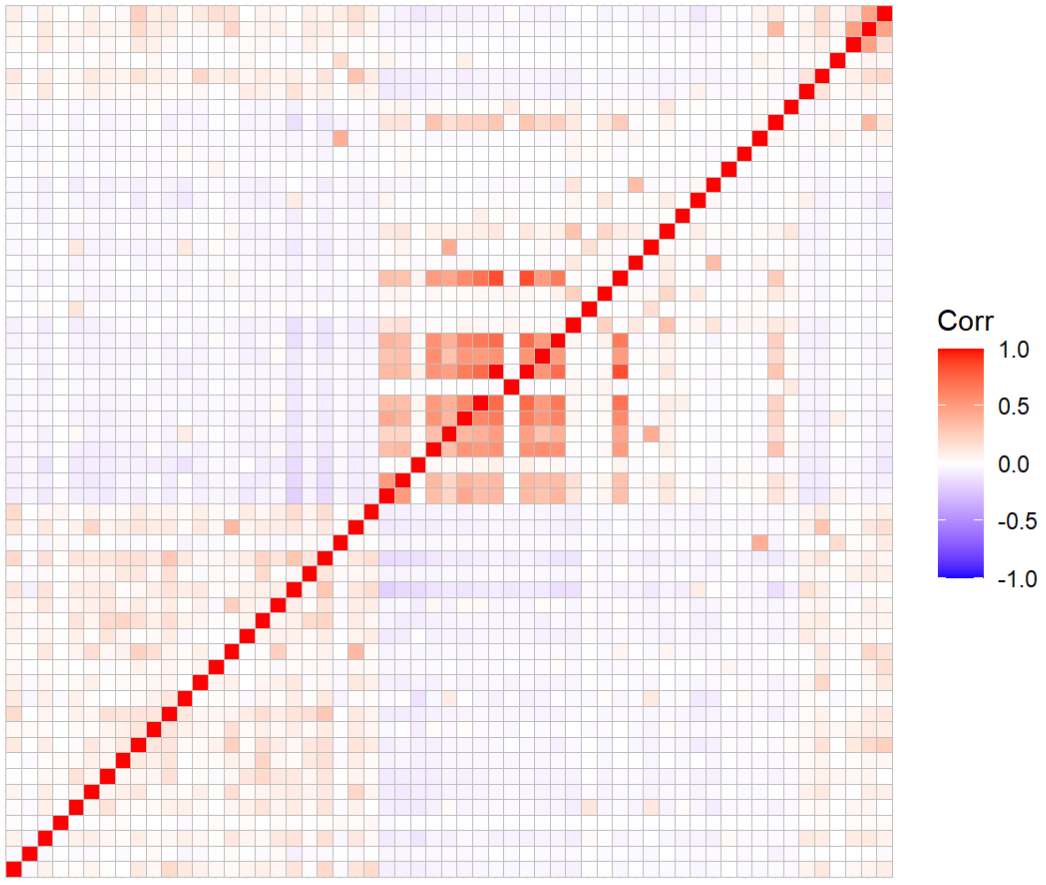


```
ggsave("figures/spam-corr.png", device="png", width=5, height=4)


# Removing variables correlated > 0.9
high.correlation = as.data.frame(spam.corr)
high.correlation[!lower.tri(high.correlation)] = 0
remove.high.correlation = spam.corr[,!apply(high.correlation, 2, function(x) any(x >
         0.9))]
remove.high.correlation=as.data.frame(remove.high.correlation)
dim(remove.high.correlation)[2]
```

```
## [1] 56
```

```
# Checking cleaned up correlations
ggcorrplot(remove.high.correlation) +
   theme(axis.text.x=element_blank(), axis.text.y=element_blank()) +
   labs(title="Cleaned Correlations")
```

Cleaned Correlations



```
ggsave("figures/2.3-clean-corr.png", device="png", width=5, height=4)

# Extracting columns with correlations < |0.9| from the original data
cleanCols = colnames(remove.high.correlation)
spamClean = spam.file %>% select(all_of(cleanCols))

# Adding labels back into the cleaned set
spamClean$spam = spam.labels$spam

# Determine which feature(s) were removed
originalFeatures = colnames(spam.file)
cleanedFeatures = colnames(spamClean)
setdiff(originalFeatures, cleanedFeatures)
```

```
## [1] "X857"
```

Task (3.a)

Use `set.seed(123)` wherever the command `sample` is used or cross-validation is implemented, randomly select without replacement 300 observations from the data set and save them as training set “train.RData,” and then randomly select without replacement 100 observations from the remaining observations and save them as “test.RData.” You need to check if the training set contains observations from both classes; otherwise, no model can be trained.

```
# Setting the seed for the chunk
set.seed(123)

# Randomly sampling 25% from the full data
rows = sample(1:nrow(spamClean), floor(0.25*nrow(spamClean)), replace=FALSE)
random.spam = as.data.frame(spamClean[rows,])



# Splitting the data into training and test set, 75-25%
train.index = sample(1:nrow(random.spam), floor(0.75*nrow(random.spam)), replace=FAL
         SE)

train.Rdata = random.spam[train.index,]
test.Rdata = random.spam[-train.index,]

# look at class balance
table(train.Rdata$spam)
```

```
##
## FALSE  TRUE
##   516   346
```

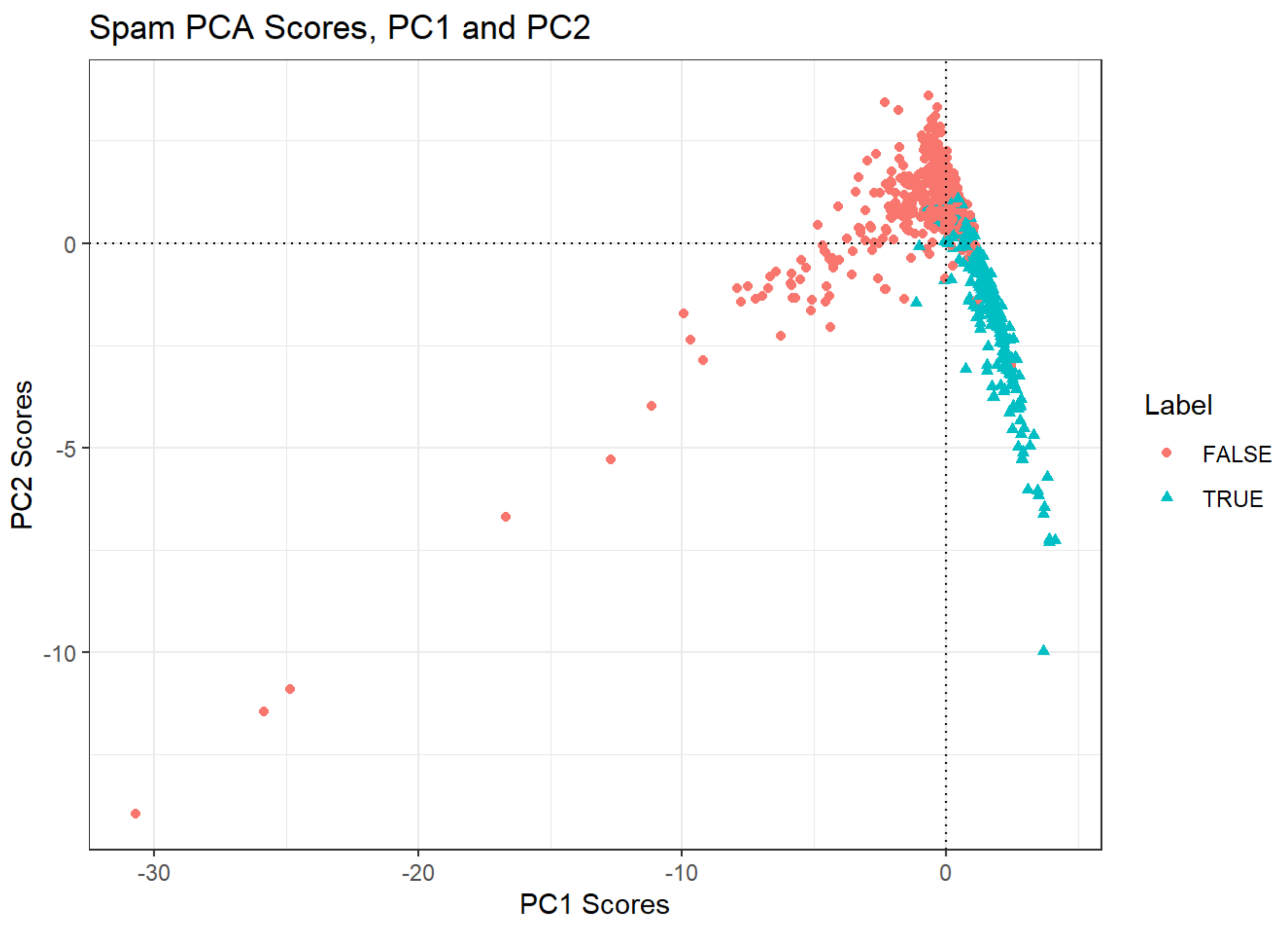
```
table(test.Rdata$spam)
```

```
##
## FALSE  TRUE
##   171   117
```

Task (3.b)

Apply PCA to the training data “train.Rdata”

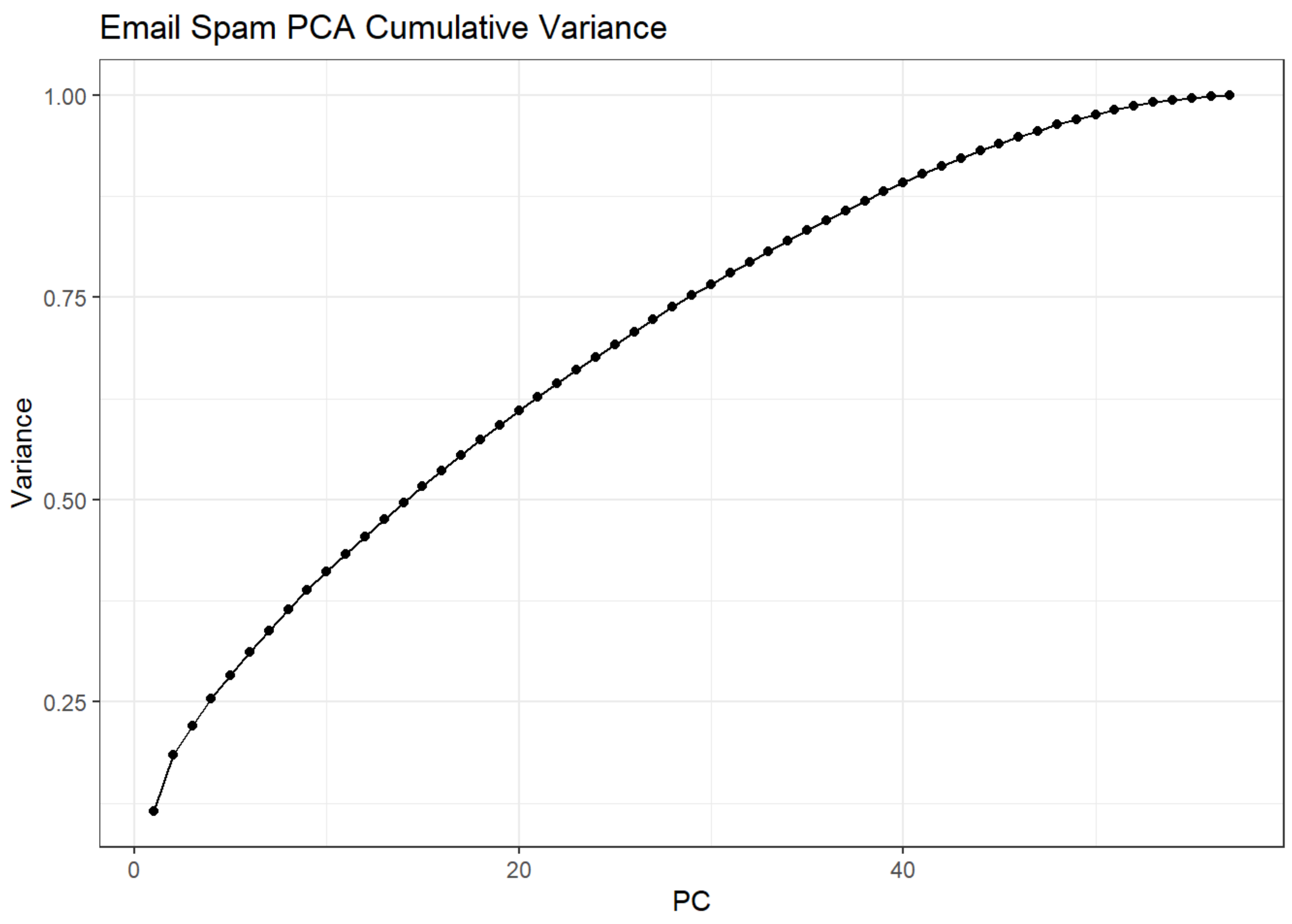
```r
# Scale data
train.Rscaled = train.Rdata %>% mutate_at(c(1:length(train.Rdata)), ~(scale(.) %>% a
         s.vector))

# PCA to the scaled training data
spam.pca = prcomp(train.Rscaled)

# Plot eigenvalues
{
fviz_eig(spam.pca,
         addlabels = TRUE,
         main = "Eigenvalues")
ggsave("figures/spam_pca_eigenvalues.png", device="png", width=5, height=4)
}


# Plot first two PC
email.scores = as.data.frame(spam.pca$x[,1:2])
email.scores$Label = train.Rdata$spam

ggplot(email.scores, aes(PC1, PC2, color=Label, shape=Label)) +
   geom_point() +
   geom_hline(yintercept=0, linetype="dotted") +
   geom_vline(xintercept=0, linetype="dotted") +
   theme_bw() +
   labs(title="Spam PCA Scores, PC1 and PC2", x="PC1 Scores", y="PC2 Scores")
```



```
ggsave("figures/spam_pca.png", device="png", width=5, height=4)

# Computing and plotting the cumulative variance explained by each PC
cum.sum = cumsum((nrow(train.Rscaled) - 1) * ((spam.pca$sdev)^2) / sum(train.Rscaled
         ^2))
cat("Cumulative Variance", cum.sum, "\n")
```

```
## Cumulative Variance 0.1147237 0.1843545 0.2204656 0.2537726 0.2832737 0.3117638
0.3383498 0.3637834 0.3882576 0.4109931 0.4329735 0.4547098 0.4758558 0.4966359 0.51
68625 0.5361624 0.5552188 0.5736394 0.5919802 0.6095737 0.6267358 0.6436995 0.660203
7 0.6762363 0.6921416 0.7075688 0.7228755 0.7380818 0.7524277 0.7664463 0.7802477 0.
7937602 0.8070153 0.8200702 0.8329289 0.8452564 0.8572666 0.8691502 0.8804262 0.8913
174 0.9020487 0.9121545 0.9216693 0.930804 0.9393182 0.9477437 0.9556277 0.9631666
0.9701993 0.9761977 0.9815406 0.986594 0.9908892 0.9938534 0.996189 0.9982003 1
```

```
cum.data = data.frame("PC"=1:length(cum.sum), "CumulativeVariance"=cum.sum)

ggplot(cum.data, aes(x=PC, y=CumulativeVariance)) +
   geom_point() +
   geom_line() +
   theme_bw() +
   labs(title="Email Spam PCA Cumulative Variance", x="PC", y="Variance")
```



```
ggsave("figures/spam_cum.png", device="png", width=5, height=4)
```

Exaplanation of 1a, b, c

```
# PC1 loading
pc1.load = pca1$rotation[,1]
pc1.score = abs(pc1.load)
pc1.sort = sort(pc1.score, decreasing=TRUE)
pc1.names = names(pc1.sort[1:10])
pc1.names
```

```
##  [1] "gene_7017"  "gene_16974" "gene_6543"  "gene_18072" "gene_5823"
##  [6] "gene_64"    "gene_12827" "gene_5138"  "gene_7165"  "gene_9642"
```

```
# PC2 loading
pc2.load = pca1$rotation[,2]
pc2.score = abs(pc2.load)
pc2.sort = sort(pc2.score, decreasing=TRUE)
pc2.names = names(pc2.sort[1:10])
pc2.names
```

```
##  [1] "gene_16493" "gene_1387"  "gene_16506" "gene_11734" "gene_1113"
##  [6] "gene_19102" "gene_15197" "gene_1959"  "gene_15033" "gene_13750"
```

```
# PC3 loading
pc3.load = pca1$rotation[,3]
pc3.score = abs(pc3.load)
pc3.sort = sort(pc3.score, decreasing=TRUE)
pc3.names = names(pc3.sort[1:10])
pc3.names
```

```
##  [1] "gene_218"   "gene_9021"  "gene_6875"  "gene_19213" "gene_14172"
##  [6] "gene_19835" "gene_14525" "gene_970"   "gene_8165"  "gene_10703"
```

```
# Seeing if any loadings are common
intersect(pc1.names, pc2.names)
```

```
## character(0)
```

```
intersect(pc1.names, pc3.names)
```

```
## character(0)
```

```
intersect(pc2.names, pc3.names)
```

```
## character(0)
```

```
# Getting PCA scores for each class of cancer for PCs 1 and 2
cancer.types = as.data.frame(pca1$x[, 1:2])
cancer.types$Class = df_labels$Class

# scores by cancer type
brca.score = cancer.types %>% select(everything()) %>% filter(Class == "BRCA") %>% s
         elect(-Class)

coad.score = cancer.types %>% select(everything()) %>% filter(Class == "COAD") %>% s
         elect(-Class)

kirc.score = cancer.types %>% select(everything()) %>% filter(Class == "KIRC") %>% s
         elect(-Class)

laud.score = cancer.types %>% select(everything()) %>% filter(Class == "LUAD") %>% s
         elect(-Class)

prad.score = cancer.types %>% select(everything()) %>% filter(Class == "PRAD") %>% s
         elect(-Class)

# Finding the mean cancer type
bra.mean = sapply(brca.score, mean)
message(sprintf("BRCA score mean: PC1=%s PC2=%s", bra.mean[1], bra.mean[2]))
```

```
## BRCA score mean: PC1=3.06832732054158 PC2=-7.30877232569077
```

```
coad.mean = sapply(coad.score, mean)
message(sprintf("COAD score mean: PC1=%s PC2=%s", coad.mean[1], coad.mean[2]))
```

```
## COAD score mean: PC1=13.2754675537057 PC2=1.81373316264351
```

```
kirc.mean = sapply(kirc.score, mean)
message(sprintf("KIRC score mean: PC1=%s PC2=%s", kirc.mean[1], kirc.mean[2]))
```

```
## KIRC score mean: PC1=-17.4616120493357 PC2=7.50314328174563
```

```
luad.mean = sapply(laud.score, mean)
message(sprintf("LAUD score mean: PC1=%s PC2=%s", luad.mean[1], luad.mean[2]))
```

```
## LAUD score mean: PC1=0.422474663904983 PC2=0.663631231353945
```

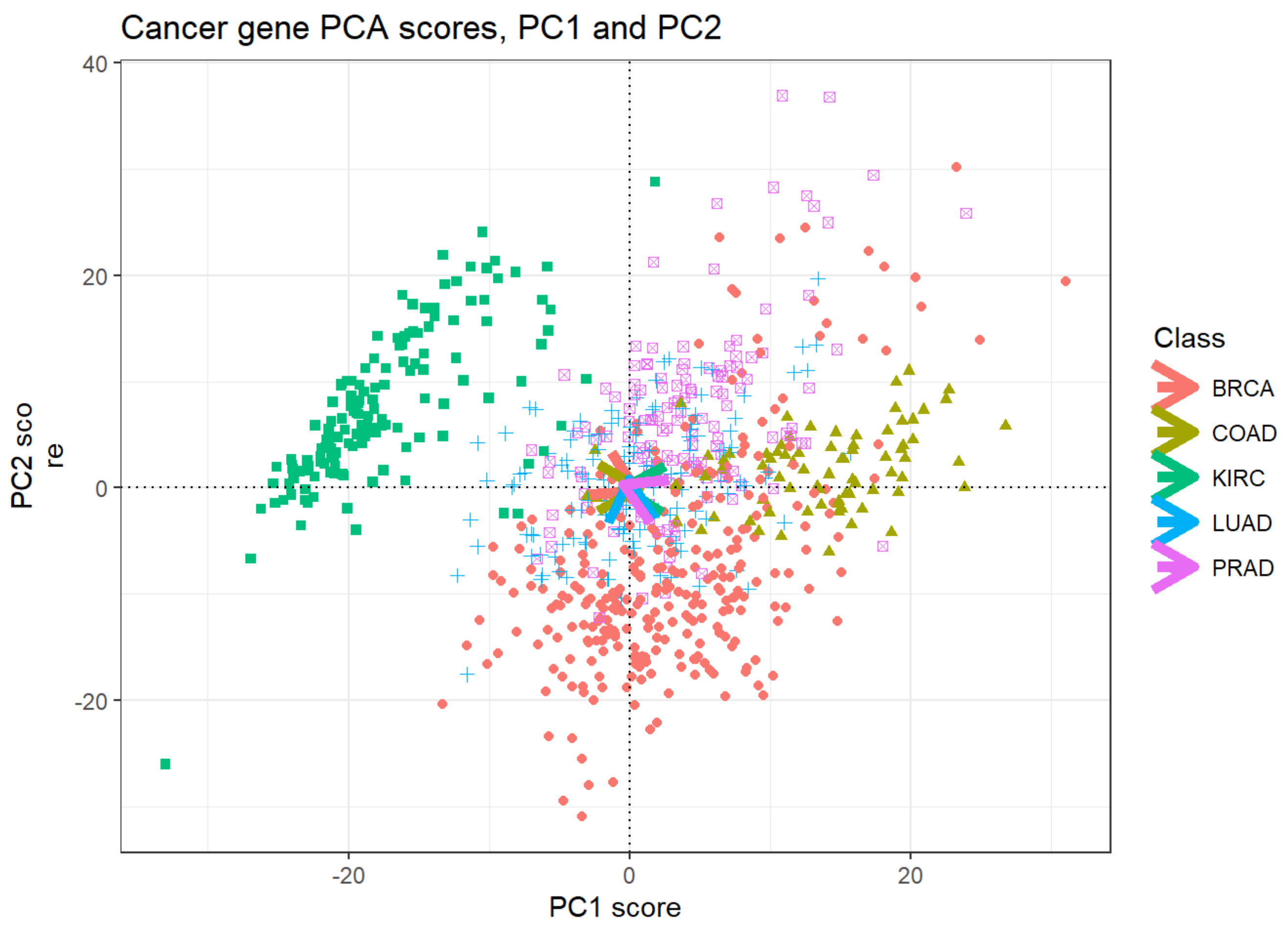
```
prad.mean = sapply(prad.score, mean)
message(sprintf("PRAD score mean: PC1=%s PC2=%s", prad.mean[1], prad.mean[2]))
```

```
## PRAD score mean: PC1=3.92530710471242 PC2=6.33918814900932
```

```
#pca1$x[1,]



(pca1$x[1,1]-pca1$center[1])/pca1$scale[1] * pca1$rotation[1,1] + (pca1$x[1,2]-pca1$
         center[2])/pca1$scale[2] * pca1$rotation[2,1] + (pca1$x[1,3]-pca1$center[3
         ])/pca1$scale[3] * pca1$rotation[3,1] + (pca1$x[1,4]-pca1$center[4])/pca1$s
         cale[4] * pca1$rotation[4,1] +
  (pca1$x[1,5]-pca1$center[5])/pca1$scale[5] * pca1$rotation[5,1] + (pca1$x[1,6]-pca
         1$center[6])/pca1$scale[6] * pca1$rotation[6,1] +
  (pca1$x[1,7]-pca1$center[7])/pca1$scale[7] * pca1$rotation[7,1] + (pca1$x[1,8]-pca
         1$center[8])/pca1$scale[8] * pca1$rotation[8,1] +
  (pca1$x[1,9]-pca1$center[9])/pca1$scale[9] * pca1$rotation[9,1] + (pca1$x[1,10]-pc
         a1$center[10])/pca1$scale[10] * pca1$rotation[10,1]
```

```
## PC1
##  NA
```

```
{
ggplot(pca.scores, aes(PC1, PC2, color = Class, shape = Class)) +
 geom_point() + theme_bw() +
 labs(title = "Cancer gene PCA scores, PC1 and PC2", x = "PC1 score", y = "PC2 sco
re") + geom_hline(yintercept =  0, linetype = "dotted") + geom_vline(xintercept = 0,
         linetype = "dotted") +
geom_segment(aes(x=0,y=0,xend=10*pca1$rotation[1,1],yend=10*pca1$rotation[1,2],color
         ="BRCA"),size=2,arrow = arrow())+
geom_segment(aes(x=0,y=0,xend=10*pca1$rotation[2,1],yend=10*pca1$rotation[2,2],color
         ="COAD"),size=2,arrow = arrow())+
geom_segment(aes(x=0,y=0,xend=10*pca1$rotation[3,1],yend=10*pca1$rotation[3,2],color
         ="KIRC"),size=2,arrow = arrow()) +
geom_segment(aes(x=0,y=0,xend=10*pca1$rotation[9,1],yend=10*pca1$rotation[9,2],color
         ="LUAD"),size=2,arrow = arrow()) +
geom_segment(aes(x=0,y=0,xend=10*pca1$rotation[10,1],yend=10*pca1$rotation[10,2],col
         or="PRAD"),size=2,arrow = arrow())
}
```


Cancer gene PCA scores, PC1 and PC2
40
20
0
-20
PC2 score
-20
0
20
PC1 score
Class
BRCA
COAD
KIRC
LUAD
PRAD