PCA and SVM Analysis on Gene Expression and Spam Data

Julia Jordan

Part 1: PCA and Sparse PCA Analysis of Gene Expression Data Set

Data set: https://archive.ics.uci.edu/dataset/401/gene+expression+cancer+rna+seq

Description: "This collection of data is part of the RNA-Seq (HiSeq) PANCAN data set, it is a random extraction of gene expressions of patients having different types of tumor: BRCA, KIRC, COAD, LUAD and PRAD."

"labels.csv" contains the cancer type for each sample, and "data.csv" contains the "gene expression profile" (i.e., expression measurements of a set of genes) for each sample. Each sample is for a subject and is stored in a row of "data.csv". The data set contains the gene expression profiles for 801 subjects, each with a cancer type, where each gene expression profile contains the gene expressions for the same set of 20531 genes.

Data processing

```r
# import data files
labels = read.csv("labels.csv")
data = read.csv("data.csv")
```

```r
set.seed(123)

# filter out columns with at least 300 zero values
gexp2 = data[, colSums(data == 0) < 300]

# randomly select 1000 columns
gexp3 = gexp2[, sample(2:dim(gexp2)[2], size = 1000)]

# standardize
stdgexpProj2 = scale(gexp3, center = TRUE, scale = TRUE)
```

Identifying patterns and low-dimensional structures

PCA

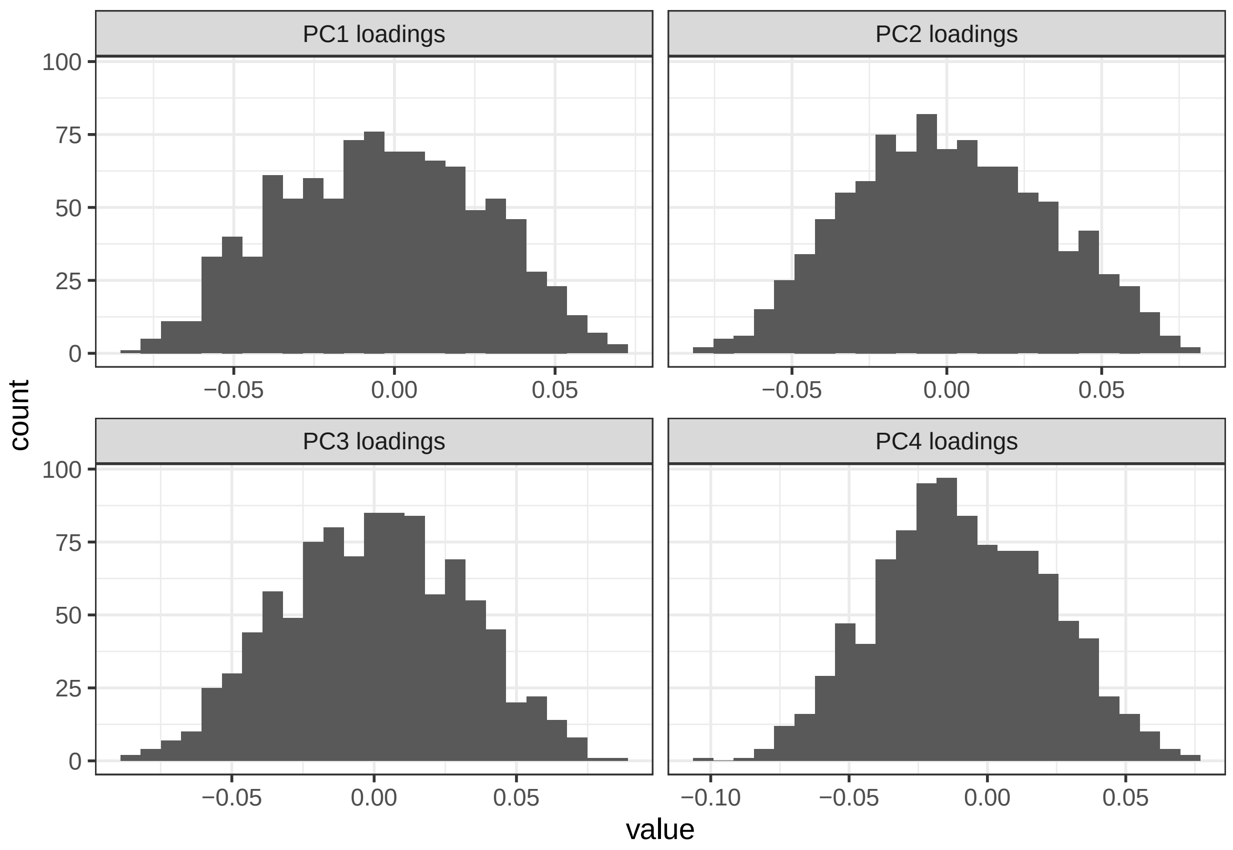
```r
library(ggplot2)
library(reshape2)
```

```
## Warning: package 'reshape2' was built under R version 4.4.3
```

```r
# apply PCA
pca = prcomp(stdgexpProj2, rank. = nrow(stdgexpProj2)-1)

# loading vectors
FourLD = pca$rotation[,1:4]
```

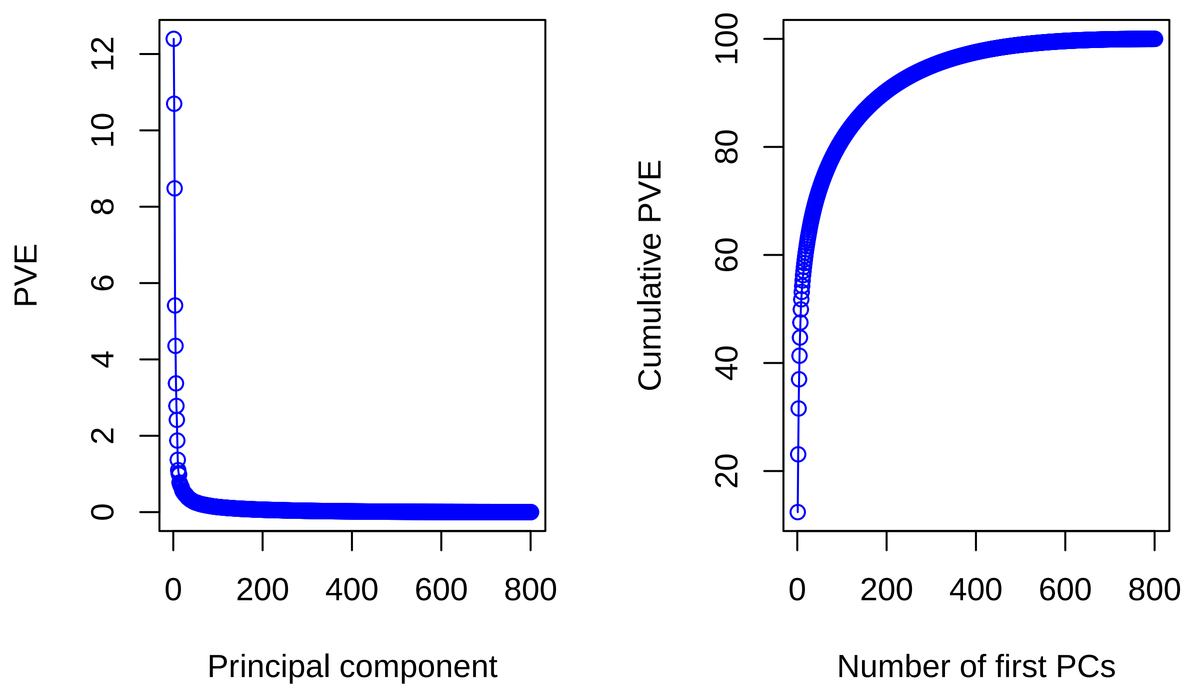
```r
colnames(FourLD) = paste(colnames(FourLD), "loadings", sep = " ")
dstack = melt(FourLD)
lvPlot = ggplot(dstack, aes(x = value)) + geom_histogram(bins = 25) +
  facet_wrap(~Var2, scales = "free_x") + theme_bw()
lvPlot
```



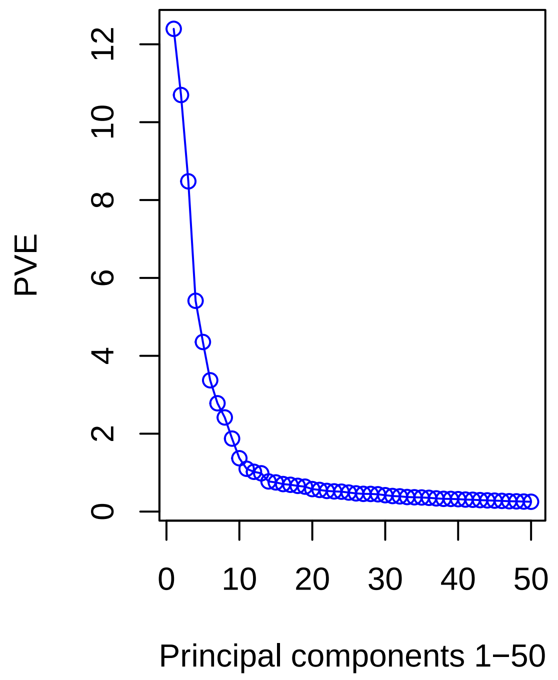
These plots of the first 4 PC loading vectors are spread over a fairly wide distribution with a large amount of nonzero entries, suggesting no dominant features. The first 3 are centered around 0, while the PC4 loadings are slightly skewed to the left of 0, indicating that PC4 has a slight negative correlation with the data.

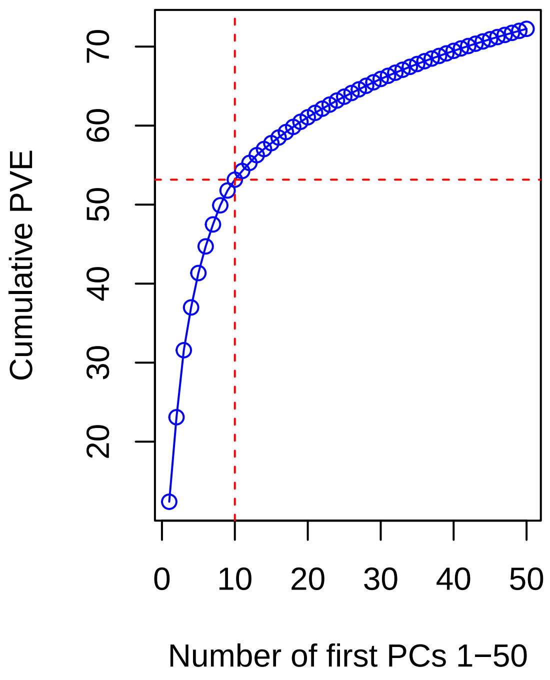
```r
# pve and cpve
pve = 100*pca$sdev^2/sum(pca$sdev^2)
cpve = cumsum(pve)

par(mfrow = c(1,2))
plot(1:length(pve), pve, type = "o", ylab = "PVE",
     xlab = "Principal component", col = "blue")
plot(cpve, type = "o", ylab = "Cumulative PVE",
     xlab = "Number of first PCs", col = "blue")
```



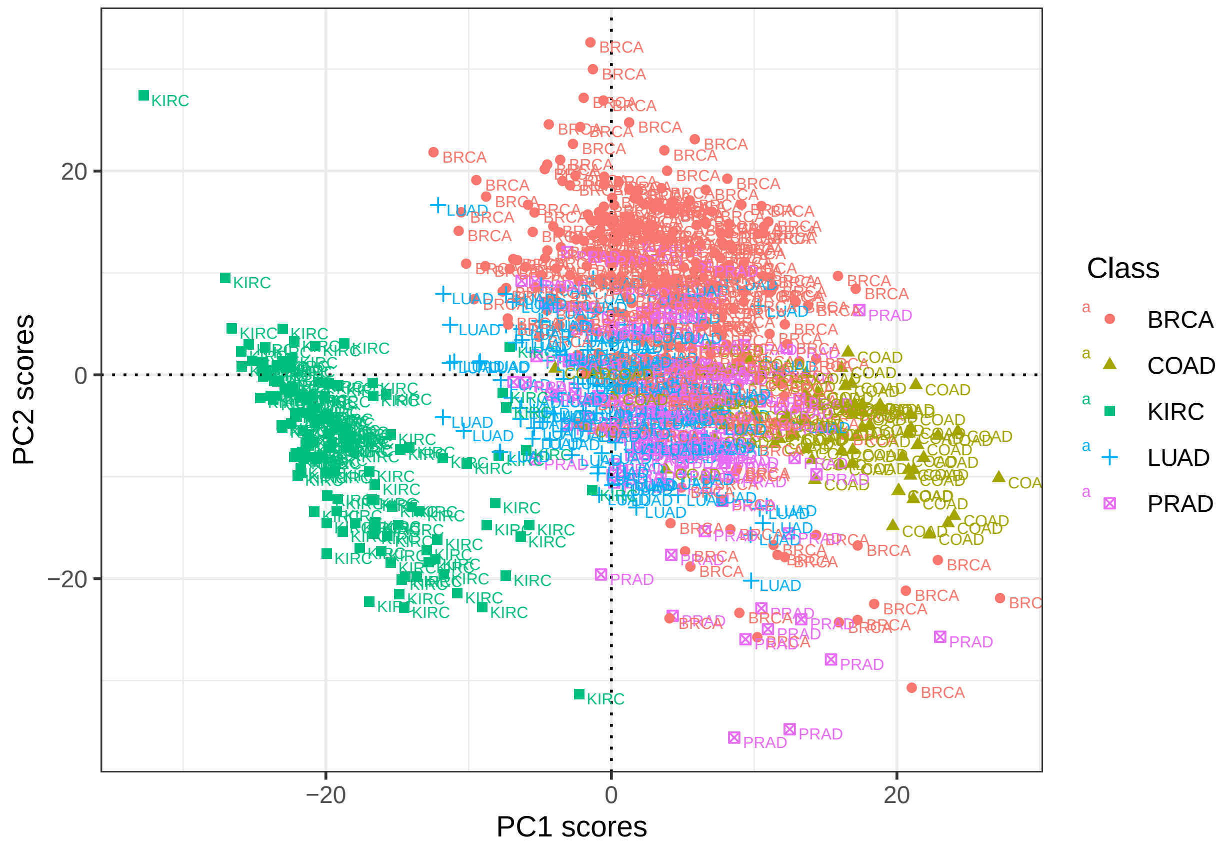
```
par(mfrow = c(1,2))
plot(1:50, pve[1:50], type = "o", ylab = "PVE",
     xlab = "Principal components 1-50", col = "blue")
plot(cpve[1:50], type = "o", ylab = "Cumulative PVE",
     xlab = "Number of first PCs 1-50", col = "blue")
abline(h = cpve[10], col = "red", lty = 2)
abline(v = 10, col = "red", lty = 2)
```





These PVE and CPVE plots show that the first few principal components explain most of the variance in the data. Looking closer at just the first 50 principal components, in the left plot, there is an elbow after about the tenth PC. In correspondence, the second plot shows that the first 10 PCs explain about 53% of the variance in the data. This suggests that we can focus on the first 10 PCs.

```r
# score vectors
scoresMat = as.data.frame(pca$x[,1:2])
scoresMat$Class = labels$Class
svPlot = ggplot(scoresMat, aes(PC1, PC2, color = Class, shape = Class)) +
  geom_point() + theme_bw() + xlab("PC1 scores") + ylab("PC2 scores") +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_vline(xintercept = 0, linetype = "dotted") +
  geom_text(aes(label = Class),hjust=-0.2, vjust = 0.9, size = 2)
svPlot
```



This plot of the first 2 PC score vectors shows a large portion of the points centered around 0. The KIRC class cluster is distinctly separate from the rest of the classes, clustered diagonally, indicating correlation with both PC1 and PC2. In addition, PC1 scores appear to be indicative of the COAD class due to its shape in following the PC1 axis. While the LUAD points are tightly clustered around 0, BRCA and PRAD have more variance as their points are more spread out.

Sparse PCA

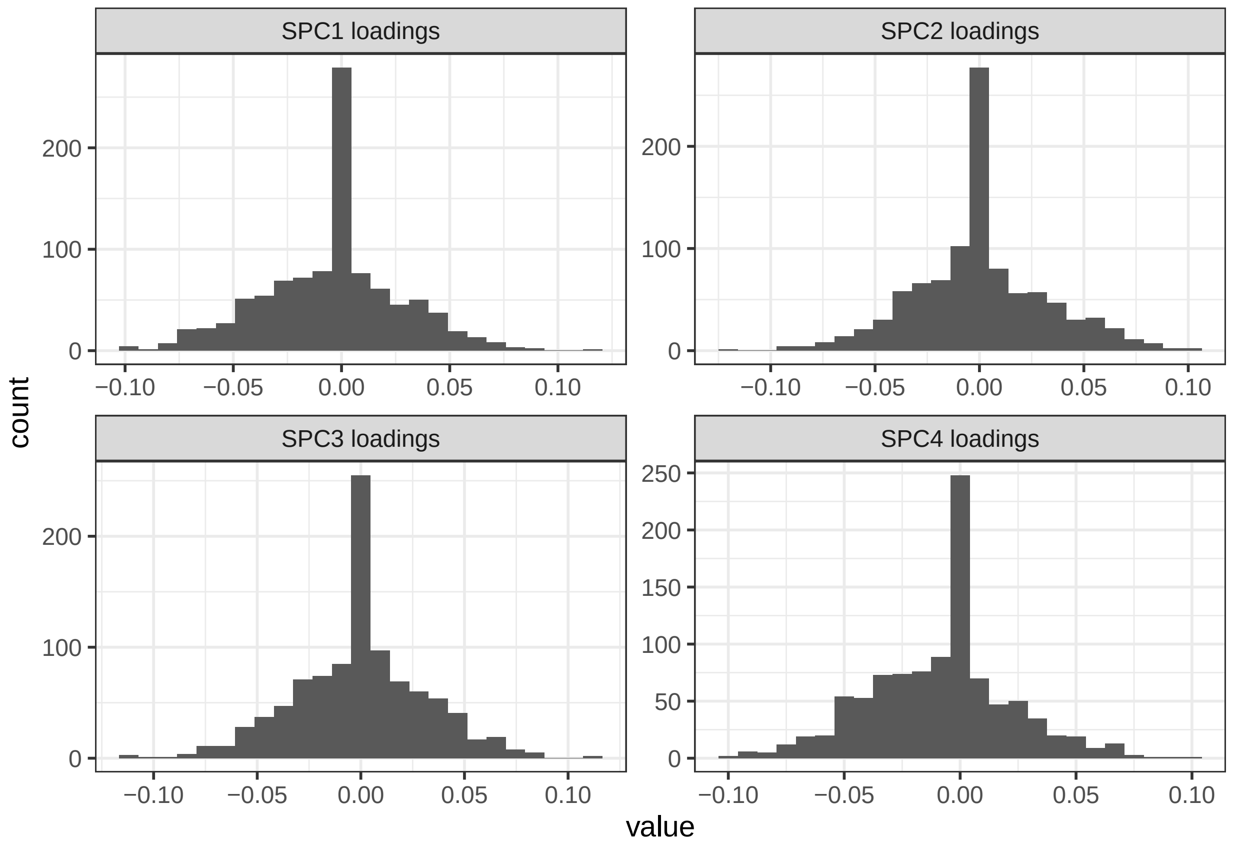
```
library(elasticnet)
```

```
## Loading required package: lars

## Loaded lars 1.3
```

```
# apply SPCA
spca = elasticnet::spca(stdgexpProj2, K = 10, para = rep(1e-2, 10),
                        type = c("predictor"), sparse = c("penalty"),
                        lambda = 1e-2, max.iter = 50, eps.conv = 1e-1)
```

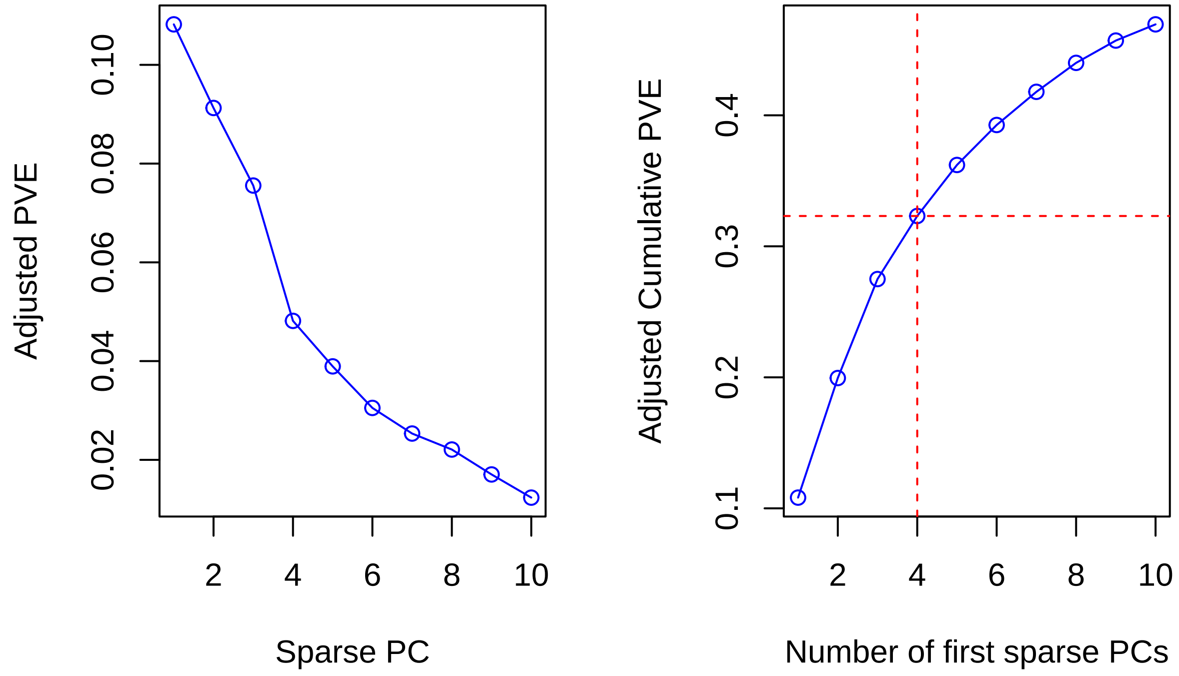
```
# loading vectors
FourLD = spca$loadings[,1:4]
colnames(FourLD) = paste("SPC", 1:4, sep = "")
colnames(FourLD) = paste(colnames(FourLD), "loadings", sep = " ")
dstack = melt(FourLD)
lvPlot = ggplot(dstack, aes(x = value)) + geom_histogram(bins = 25) +
  facet_wrap(~Var2, scales = "free") + theme_bw()
lvPlot
```



These plots of the first 4 sparse loading vectors have a high peak at 0, meaning a smaller number of nonzero loadings which suggests that there are dominant features.

```
# pve and cpve
pve = spca$pev
cpve = cumsum(pve)

par(mfrow = c(1,2))
plot(pve, type = "o", ylab = "Adjusted PVE",
     xlab = "Sparse PC", col = "blue")
plot(cpve, type = "o", ylab = "Adjusted Cumulative PVE",
     xlab = "Number of first sparse PCs", col = "blue")
abline(h = cpve[4], col = "red", lty = 2)
abline(v = 4, col = "red", lty = 2)
```



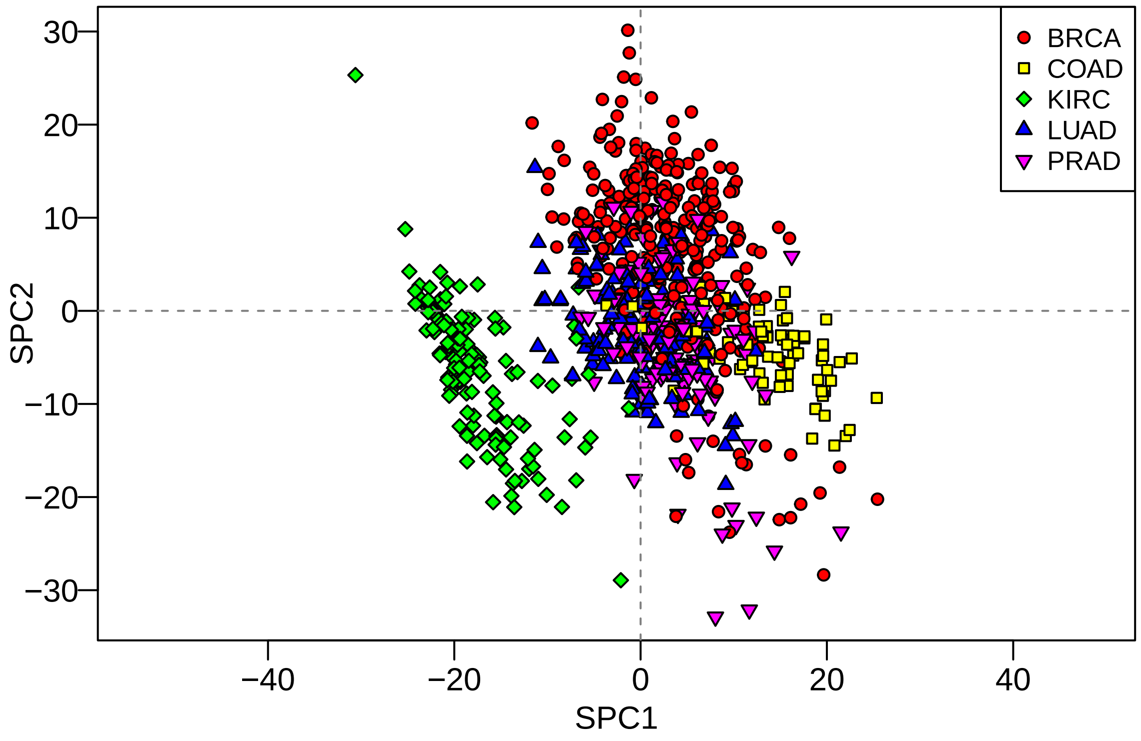
The adjusted PVE in the left plot flattens its slope at about PC 4 to 5. The second plot shows that the first 4 sparse PCs explains about 33% of the data.

```
# score vectors
sLVTwo = spca$loadings[,1:2]
sSVTwo = stdgexpProj2%*%sLVTwo

pch.group = rep(21, nrow(sSVTwo))
pch.group[labels$Class == 'BRCA'] = 21
pch.group[labels$Class == 'COAD'] = 22
pch.group[labels$Class == 'KIRC'] = 23
pch.group[labels$Class == 'LUAD'] = 24
pch.group[labels$Class == 'PRAD'] = 25

col.group = rep("blue", nrow(sSVTwo))
col.group[labels$Class == 'BRCA'] = "red"
col.group[labels$Class == 'COAD'] = "yellow"
col.group[labels$Class == 'KIRC'] = "green"
col.group[labels$Class == 'LUAD'] = "blue"
col.group[labels$Class == 'PRAD'] = "magenta"

par(mar = c(5, 4.5, 1, 1), mgp = c(1.5, 0.5, 0))
plot(sSVTwo[,1], sSVTwo[,2], xlab = "SPC1", ylab = "SPC2", col = "black",
     pch = pch.group, bg = col.group, las = 1, asp = 1, cex = 0.8)
abline(v = 0, lty = 2, col = "grey50")
abline(h = 0, lty = 2, col = "grey50")
legend("topright", legend = c("BRCA", "COAD", "KIRC", "LUAD", "PRAD"),
       pch = c(21, 22, 23, 24, 25),
       pt.bg = c("red", "yellow", "green", "blue", "magenta"), cex = 0.8)
```



The score vectors plot for PCA and sparse PCA look very similar, showing the same patterns for the cancer types.

Part 2: Analysis of Spam Emails Data Set

Data set: https://hastie.su.domains/CASI_files/DATA/SPAM.html

Description: The column "spam" contains the status for each email (whether it is a spam email or not), and the rest contain measurements of features. The first 1813 rows of the data set are for spam emails, and the rest for non-spam emails.

Data processing

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
```

```
##     intersect, setdiff, setequal, union
```

```r
# import data file
spam = read.csv("SPAM.csv")

# remove rows with missing values
spam = na.omit(spam)

# check for high correlation
corr = cor(spam[,-(1:2)])
which(abs(corr) > 0.8 & abs(corr) < 1, arr.ind = TRUE)
```

```
##        row col
## X415    34  32
## direct  40  32
## X857    32  34
## direct  40  34
## X857    32  40
## X415    34  40
```

```r
# remove highly correlated features
spam = spam %>% select(-X415, -direct, -X857)
```

Features “X415”, “direct”, and “X857” are highly correlated with correlation coefficients above 0.8. I will omit these features from the analysis in order to avoid collinearity and create a more accurate model.

Classification via SVM

```r
set.seed(123)

# set training and testing
rTrain = sample(1:nrow(spam), size = 300, replace = FALSE)
train = spam[rTrain,]
spam1 = spam[-rTrain,]
rTest = sample(1:nrow(spam1), size = 100, replace = FALSE)
test = spam[rTest,]

save(train, file = "train.RData")
save(test, file = "test.RData")
```

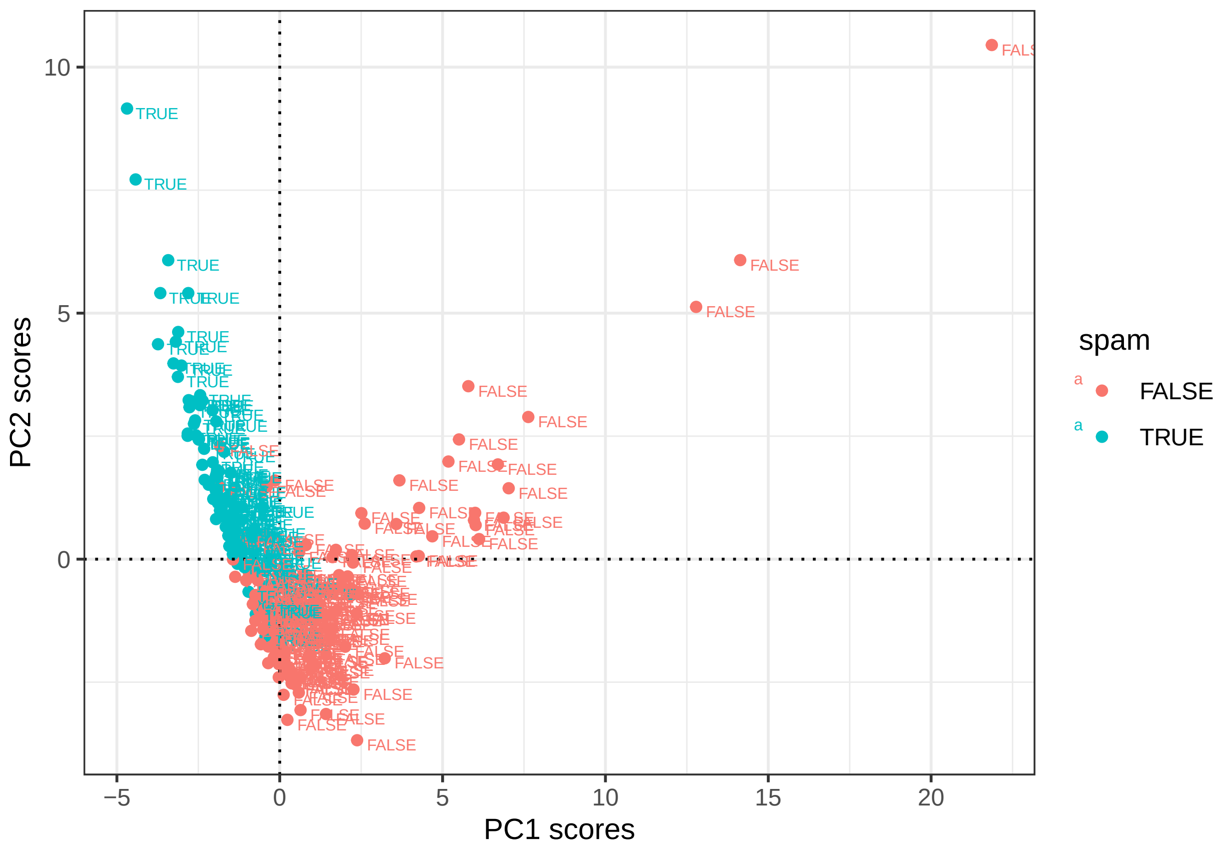
PCA on training data

```r
load("train.RData")
load("test.RData")

# standardize
train = cbind(train[,1:2], scale(train[,3:ncol(train)], center = TRUE, scale = TRUE))

# apply pca
pcaTrain = prcomp(train)

# score vectors
scoresMat = as.data.frame(pcaTrain$x[,1:2])
scoresMat$spam = train$spam
svPlot = ggplot(scoresMat, aes(PC1, PC2, color = spam)) +
  geom_point() + theme_bw() + xlab("PC1 scores") + ylab("PC2 scores") +
  geom_hline(yintercept = 0, linetype = "dotted") +
  geom_vline(xintercept = 0, linetype = "dotted") +
  geom_text(aes(label = spam),hjust=-0.2, vjust = 0.9, size = 2)
svPlot
```



This plot shows a notable difference between the FALSE (non-spam) and TRUE (spam) clusters. They are both somewhat clustered around 0, but trail into distinct directions, with FALSE going up and right and TRUE going up and left. Since they go upward along the PC2 axis about the same amount but differ horizontally along PC1, we can infer that PC1 is effective at capturing the difference between the two classes. Overall, the plot suggests that there are patterns that can be used to distinguish between spam and non-spam emails.

SVM model with linear kernel

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.4.3
```

```
set.seed(123)

train = train %>% select(-testid)
train$spam = as.factor(train$spam)

# cross validation for C
tune.out = tune(svm, spam~., data = train, kernel = "linear",
                ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 50)))
```

```r
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##   0.1
##
## - best performance: 0.1033333
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.1166667 0.08050765
## 2  0.10 0.1033333 0.06929985
## 3  1.00 0.1066667 0.06440612
## 4  5.00 0.1033333 0.05317105
## 5 10.00 0.1100000 0.06095941
## 6 50.00 0.1400000 0.05621827
```

```r
# optimal cost = 0.1

# svm model with linear kernel
svmLinear = tune.out$best.model

# apply svm model to test set
predLinear = predict(svmLinear, test[,3:ncol(test)])

# classification table
table(predicted = predLinear, truth = test$spam)
```

```
##          truth
## predicted FALSE TRUE
##     FALSE     3    0
##     TRUE     53   44
```

SVM model with radial kernel

```r
set.seed(123)

# cross validation for C
tune.out = tune(svm, spam~., data = train, kernel = "radial",
                ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 50),
                              gamma = c(0.5, 1, 2, 3, 4)))
summary(tune.out)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     5   0.5
##
## - best performance: 0.32
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1   0.01   0.5 0.3866667 0.05921294
## 2   0.10   0.5 0.3866667 0.05921294
## 3   1.00   0.5 0.3400000 0.05163978
## 4   5.00   0.5 0.3200000 0.05921294
## 5  10.00   0.5 0.3200000 0.05921294
## 6  50.00   0.5 0.3233333 0.05454639
## 7   0.01   1.0 0.3866667 0.05921294
## 8   0.10   1.0 0.3866667 0.05921294
## 9   1.00   1.0 0.3433333 0.05223404
## 10  5.00   1.0 0.3400000 0.05163978
## 11 10.00   1.0 0.3433333 0.04981447
## 12 50.00   1.0 0.3433333 0.04981447
## 13  0.01   2.0 0.3866667 0.05921294
## 14  0.10   2.0 0.3866667 0.05921294
## 15  1.00   2.0 0.3533333 0.05708992
## 16  5.00   2.0 0.3500000 0.05270463
## 17 10.00   2.0 0.3533333 0.05018484
## 18 50.00   2.0 0.3533333 0.05018484
## 19  0.01   3.0 0.3866667 0.05921294
## 20  0.10   3.0 0.3866667 0.05921294
## 21  1.00   3.0 0.3533333 0.05708992
## 22  5.00   3.0 0.3566667 0.05454639
## 23 10.00   3.0 0.3566667 0.05454639
## 24 50.00   3.0 0.3566667 0.05454639
## 25  0.01   4.0 0.3866667 0.05921294
## 26  0.10   4.0 0.3866667 0.05921294
## 27  1.00   4.0 0.3533333 0.05708992
## 28  5.00   4.0 0.3566667 0.05454639
## 29 10.00   4.0 0.3566667 0.05454639
## 30 50.00   4.0 0.3566667 0.05454639
```

```r
# optimal cost = 5
# optimal gamma = 0.5

# svm model with radial kernel
svmRadial = tune.out$best.model

# apply svm model to test set
predRadial = predict(svmRadial, test[,3:ncol(test)])

# classification table
table(predicted = predRadial, truth = test$spam)
```

```
##          truth
## predicted FALSE TRUE
##     FALSE    56   44
##     TRUE      0    0
```

Classification results

The SVM with linear kernel and optimal cost C = 0.1 misclassified 53 observations in the test set, resulting in a 47% accuracy.

The SVM with radial kernel, optimal cost C = 5, and optimal $\gamma = 0.5$ misclassified 44 observations, resulting in a 56% accuracy.

Both of these models did not have high accuracy percentages, only correctly classifying about half of the time. In addition, although the radial SVM had a higher accuracy percentage, this was due to predicting FALSE for every observation in the test set. On the other hand, the linear SVM mostly predicted TRUE (97/100). This suggests that these models are not very effective at fitting the data to predict spam vs non-spam emails. Further, they each seem to favor towards one class.