Stat 437 Project 2

Samantha Gregoryk (11559189)

```
## Loading required package: ggplot2

##
## 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

## corrplot 0.84 loaded

## Loading required package: lars

## Loaded lars 1.2

## Registered S3 method overwritten by 'elasticnet':
##   method     from
##   print.spca sparsepca

##
## Attaching package: 'elasticnet'

## The following object is masked from 'package:sparsepca':
##
##     spca
```

Task A: Analysis of gene expression data

For this task, you need to use PCA and Sparse PCA.

Data set and its description

Please download the data set "TCGA-PANCAN-HiSeq-801x20531.tar.gz" from the website https://archive.ics.uci.edu/ml/machine-learning-databases/00401/. A brief description of the data set is given at https://archive.ics.uci.edu/ml/datasets/gene+expression+cancer+RNA-Seq.

You need to decompress the data file since it is a .tar.gz file. Once uncompressed, the data files are "labels.csv" that contains the cancer type for each sample, and "data.csv" that contains the "gene expression profile" (i.e., expression measurements of a set of genes) for each sample. Here each sample is for a subject and is stored in a row of "data.csv". In fact, 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. The cancer types are: "BRCA", "COAD", "KIRC", "LUAD" and "PRAD". In both files "labels.csv" and "data.csv", each row name records which sample a label or observation is for.

Data processing

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.

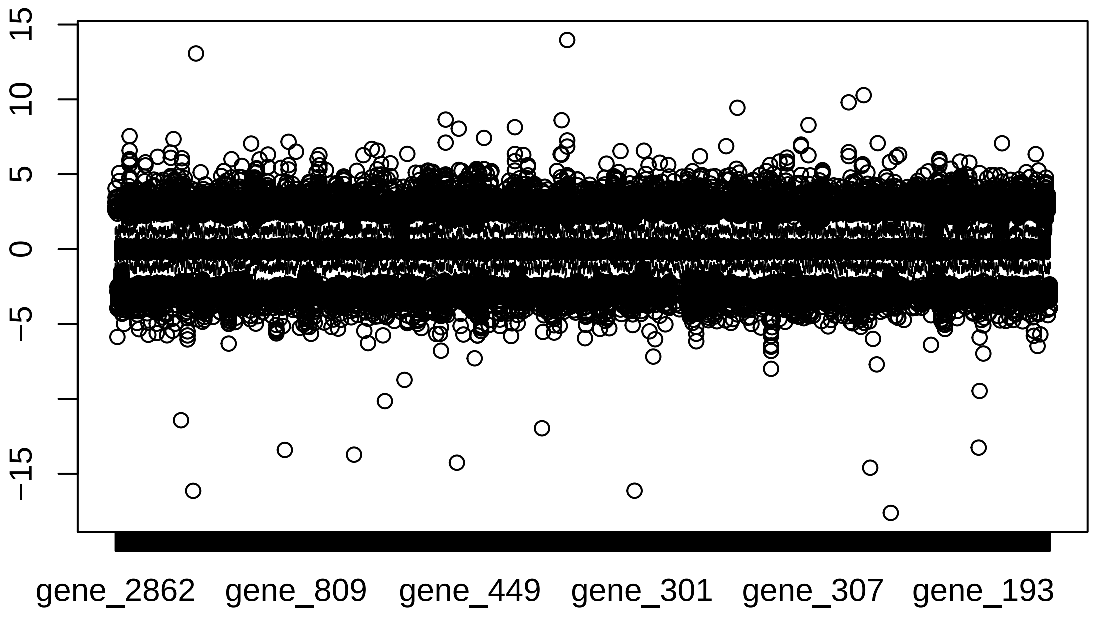
```r
set.seed(123)

labels <- read.csv("labels.csv")
data <- read.csv("data.csv")
data <- na.omit(data)

gexp2 <- data[, (colSums(data == 0, na.rm = TRUE) < 300), drop = TRUE]
gexp3 <- sample(gexp2, 1000)
stdgexpProj2 <- scale(gexp3)
```

Questions to answer when doing data analysis

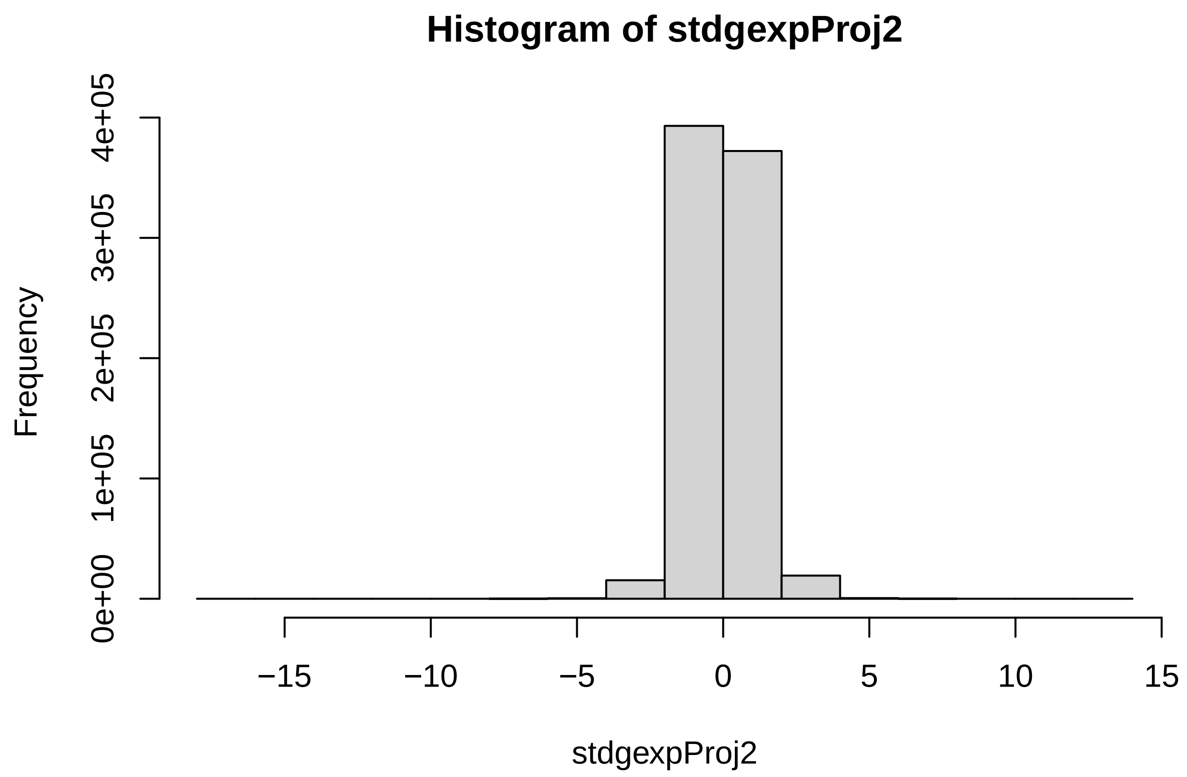
Please also investigate and address the following when doing data analysis:

(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.

```
boxplot(stdgexpProj2)
```



```
hist(stdgexpProj2)
```

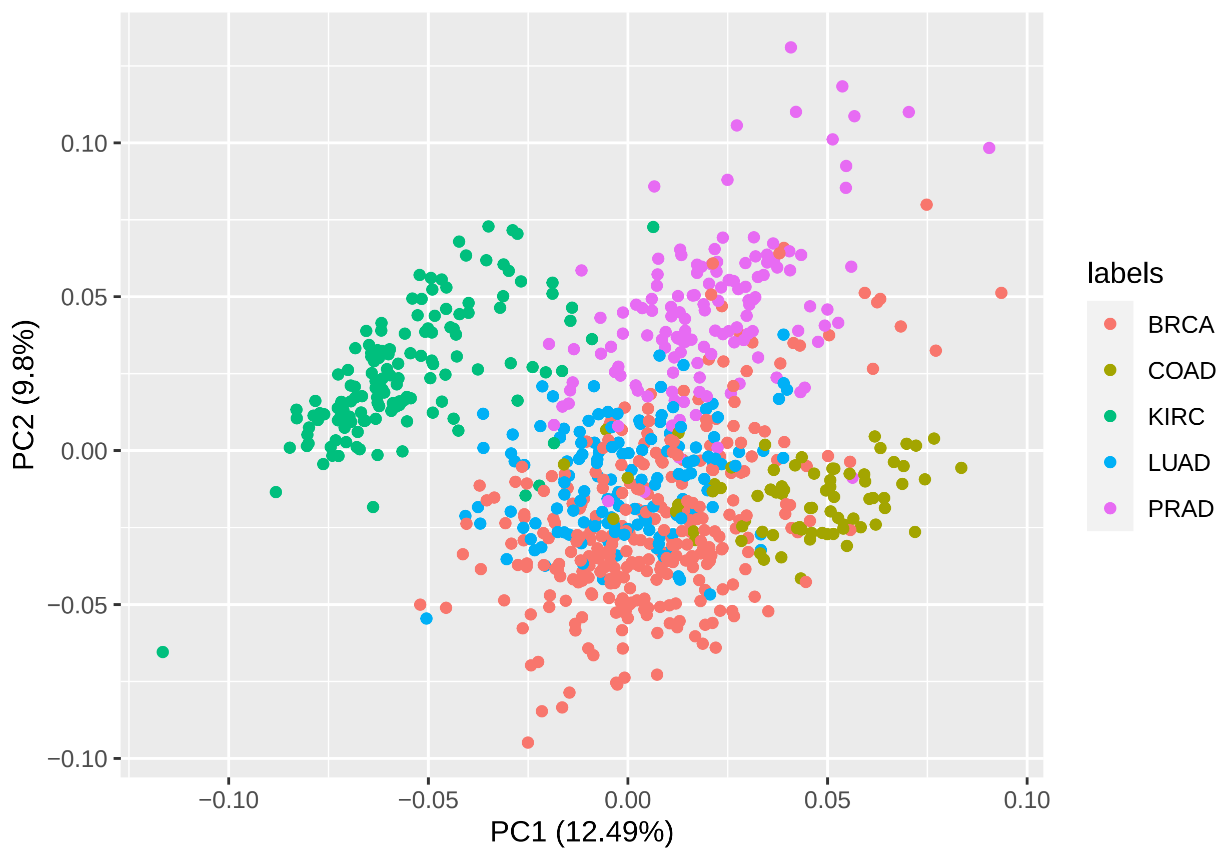


- There are no specific genes for which linear combinations of their expressions explain a significant proportion of the variation of gene expressions in the data set. The variation of the genes are normally distributed according to the boxplot and histogram of the data.

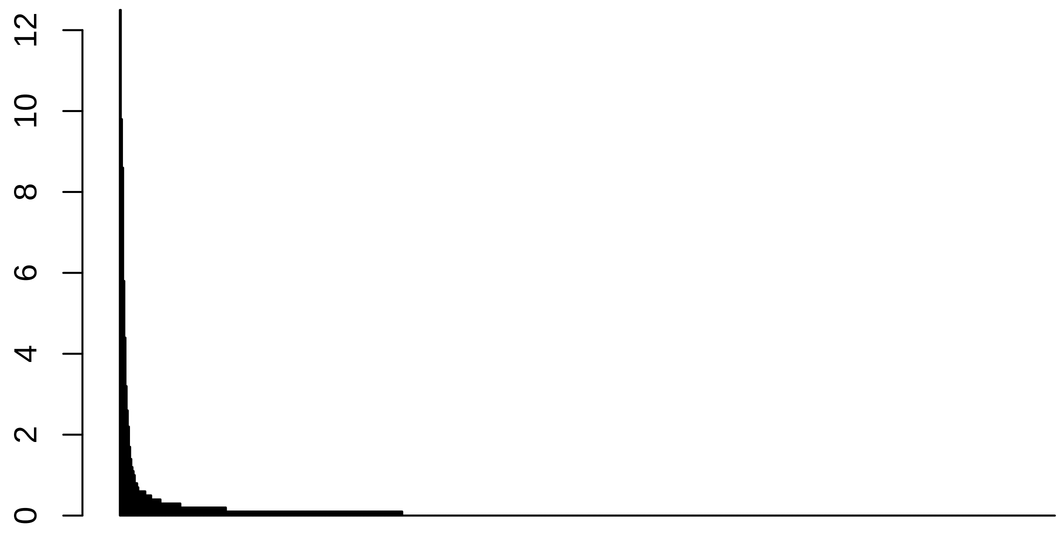
(1.b) Ideally, a type of cancer should have its "signature", i.e., a pattern in the gene expressions that is specific to this cancer type. From the "labels.csv", you will know which expression measurements belong to which cancer type. Identify the signature of each cancer type (if any) and visualize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

```r
stdgexpProj2 <- as.data.frame(stdgexpProj2)
stdgexpProj2$labels <- labels[, 2]
stdgexpProj2 <- stdgexpProj2[, c(1001, 1:1000)]

# PCA
stdgexpProj2.df <- stdgexpProj2[2:1001]
pca <- prcomp(stdgexpProj2.df)
autoplot(pca, data = stdgexpProj2, colour = "labels")
```

```
## Warning: 'select_()' is deprecated as of dplyr 0.7.0.
## Please use 'select()' instead.
## This warning is displayed once every 8 hours.
## Call 'lifecycle::last_warnings()' to see where this warning was generated.
```



```r
pca.var <- pca$sdev^2
pca.per <- round(pca.var/sum(pca.var) * 100, 1)
barplot(pca.per)
```



```
# poisson distribution

# SPCA
s.pca <- elasticnet::spca(stdgexpProj2.df, K = 5, para = rep(1e-06,
    5), type = c("predictor"), sparse = c("penalty"), lambda = 1e-06,
    max.iter = 200, eps.conv = 0.001)
dim(s.pca$loadings)
```

```
## [1] 1000    5
```

(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 5 principal components obtained either from PCA do not explain a dominant proportion of variability in the data set, and do not serve as the signatures of the 5 cancer types. The percentage of PC1 and PC2 have very low percentage of variances so it is difficult to determine if there are any dominant proportions in the data set.

Identify patterns and low-dimensional structures

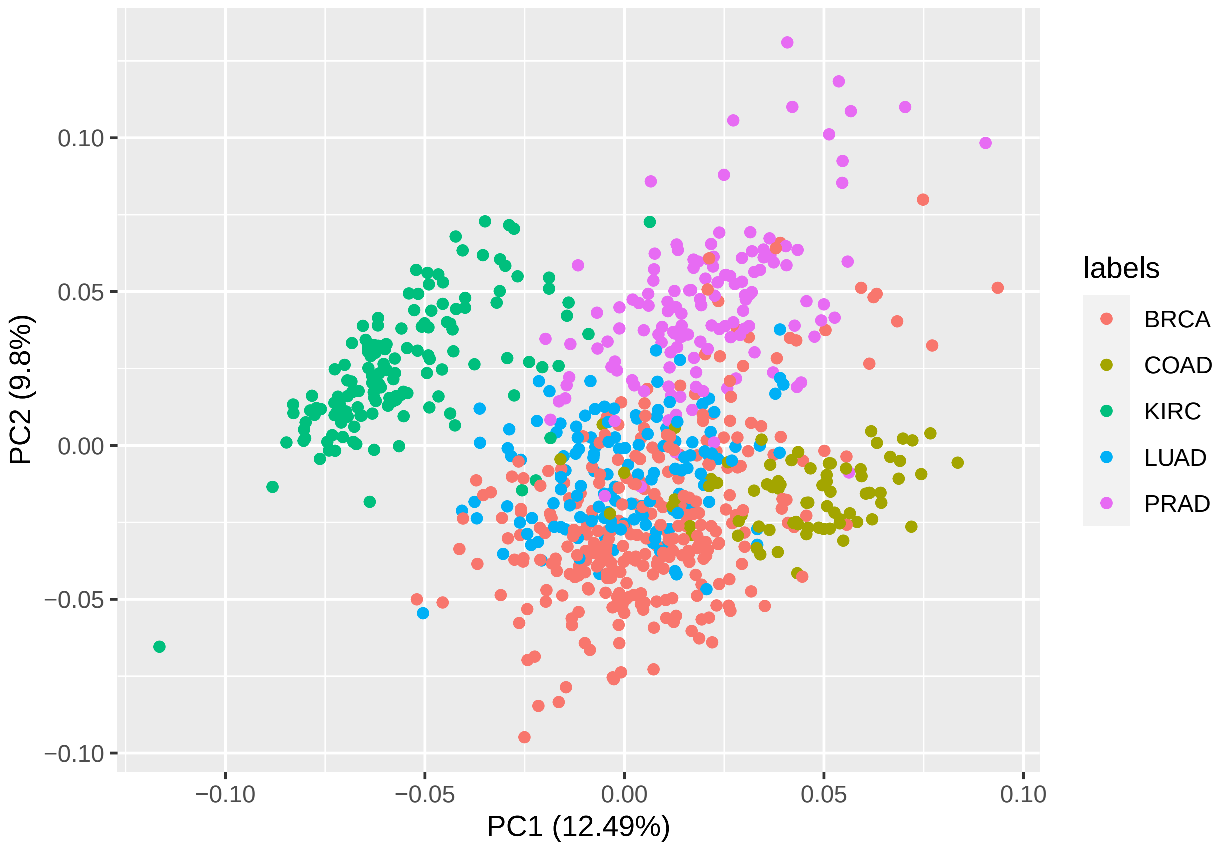
Please implement the following:

(2.a) Apply PCA, determine the number of principal components, provide visualizations of low-dimensional structures, and report your findings. Note that you need to use "labels.csv" for the task of discoverying patterns such as if different cancer types have distinct transformed gene expressions (that are represented by principal components). For PCA or Sparse PCA, low-dimensional structures are usually represented by the linear space spanned by some principal components.

```
P2sd = scale(stdgexpProj2.df, center = TRUE, scale = TRUE)
dim(P2sd)
```

```
## [1]  801 1000
```

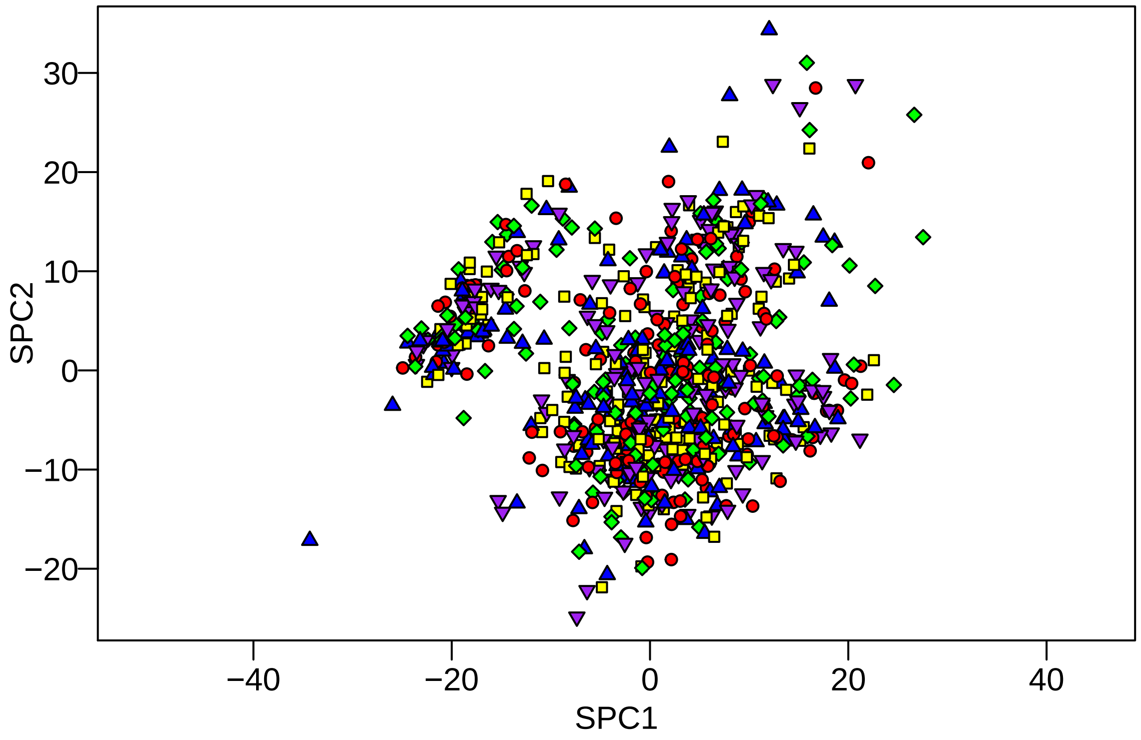
```
pca.sd <- prcomp(P2sd)
autoplot(pca.sd, data = stdgexpProj2, colour = "labels")
```



- PCA has 801 principle components with PC1 having 12.49% variability and PC2 having 9.8% variability. Cancer types like KIRC has distinct transformed gene expressions and partially BRCA, PRAD and COAD.

(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 discoverying 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.

```
nci.labs = unique(stdgexpProj2$labels)
nci.labs1 = nci.labs[nci.labs %in% c("BRCA", "COAD", "KIRC",
    "LUAD", "PRAD")]

sLVTwo = s.pca$loadings[, 1:2]
sSVTwo = P2sd %*% sLVTwo

pch.group = rep(21, nrow(sSVTwo))
pch.group[nci.labs1 == "BRCA"] = 21
pch.group[nci.labs1 == "COAD"] = 22
pch.group[nci.labs1 == "KIRC"] = 23
pch.group[nci.labs1 == "LUAD"] = 24
pch.group[nci.labs1 == "PRAD"] = 25

col.group = rep("blue", nrow(sSVTwo))
col.group[nci.labs1 == "BRCA"] = "red"
col.group[nci.labs1 == "COAD"] = "yellow"
col.group[nci.labs1 == "KIRC"] = "green"
col.group[nci.labs1 == "LUAD"] = "blue"
col.group[nci.labs1 == "PRAD"] = "purple"

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)
```



- It is difficult to determine patterns withing the sparse PCA because all points are clustered on top of one another.

(2.c) Do PCA and Sparse PCA reveal different low-dimensional structures for the gene expressions for different cancer types?

- PCA reveals a low-dimensional structure for gene expressions for different cancer types better than Sparse PCA.

Task B: analysis of SPAM emails data set

For this task, you need to use PCA and SVM.

Dataset and its description

The spam data set "SPAM.csv' ' is attached and also can be downloaded from https://web.stanford.edu/~hastie/CASI_files/DATA/SPAM.html. More information on this data set can be found at: https://archive.ics.uci.edu/ml/datasets/Spambase. The column "testid" in "SPAM.csv" was used to train a model when the data set was used by other analysts and hence should not be used as a feature or the response, the column "spam" contains the true status for each email, and the rest contain measurements of features. Here each email is represented by a row of features in the .csv file, and a “feature” can be regarded as a “predictor”. Also note that the first 1813 rows, i.e., observations, of the data set are for spam emails, and that the rest for non-spam emails.

Data processing

Please do the following:

- Remove rows that have missing values. For a .csv file, usually a blank cell is treated as a missing value.

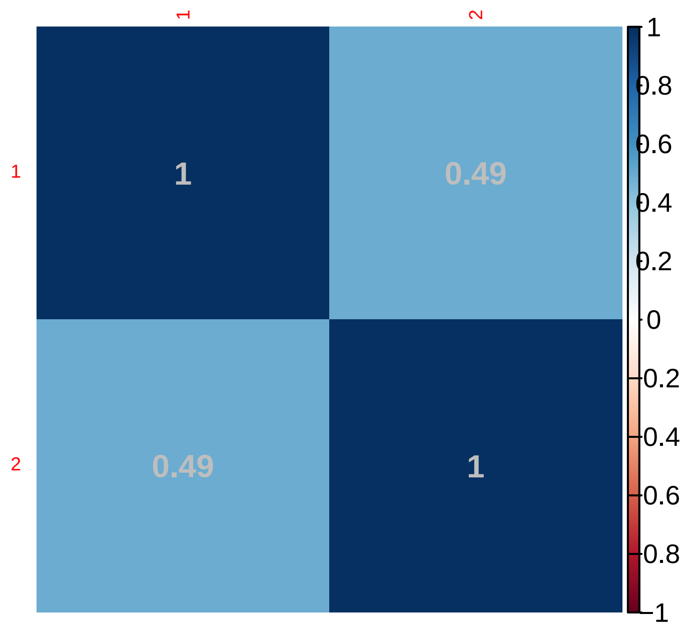
```r
spam <- read.csv("SPAM.csv")
spam <- na.omit(spam)

dim(spam)
```

```
## [1] 4601   59
```

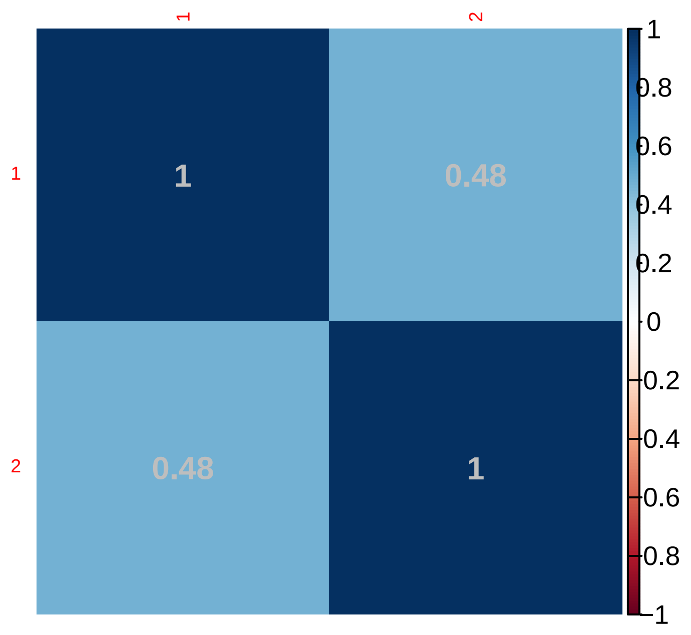
```r
spam <- transform(spam, testid = as.numeric(testid))
spam <- transform(spam, spam = as.numeric(spam))
```

- Check for highly correlated features using the absolute value of sample correlation. Think about if you should include all or some of highly correlated features into an SVM model. For example, “crl.ave” (average length of uninterrupted sequences of capital letters), “crl.long” (length of longest uninterrupted sequence of capital letters) and “crl.tot” (total number of capital letters in the e-mail) may be highly correlated. Whethere you choose to remove some highly correlated features from subsequent analysis or not, you need to provide a justification for your choice.

```r
s1 <- cbind(spam$crl.ave, spam$crl.long)
corMatFeature = cor(s1)
x <- abs(corMatFeature)
corrplot(x, method = "shade", addCoef.col = "grey", tl.cex = 0.57,
    mar = c(4.5, 1, 1, 1))
```



```
s2 <- cbind(spam$crl.long, spam$crl.tot)
corMatFeature = cor(s2)
x <- abs(corMatFeature)
corrplot(x, method = "shade", addCoef.col = "grey", tl.cex = 0.57,
    mar = c(4.5, 1, 1, 1))
```



```r
spam.df <- subset(spam, select = -c(57:59))
```

- I chose to remove crl.ave, crl.long, and crl.tot from the data set because of their high correlation to one another. crl.ave has a high correlation of 0.49 with crl.long and crl.long has a high correlation of 0.48 with crl.tot.

Note that each feature is stored in a column of the original data set and each observation in a row. You will analyze the processed data set.

Classifiction via SVM

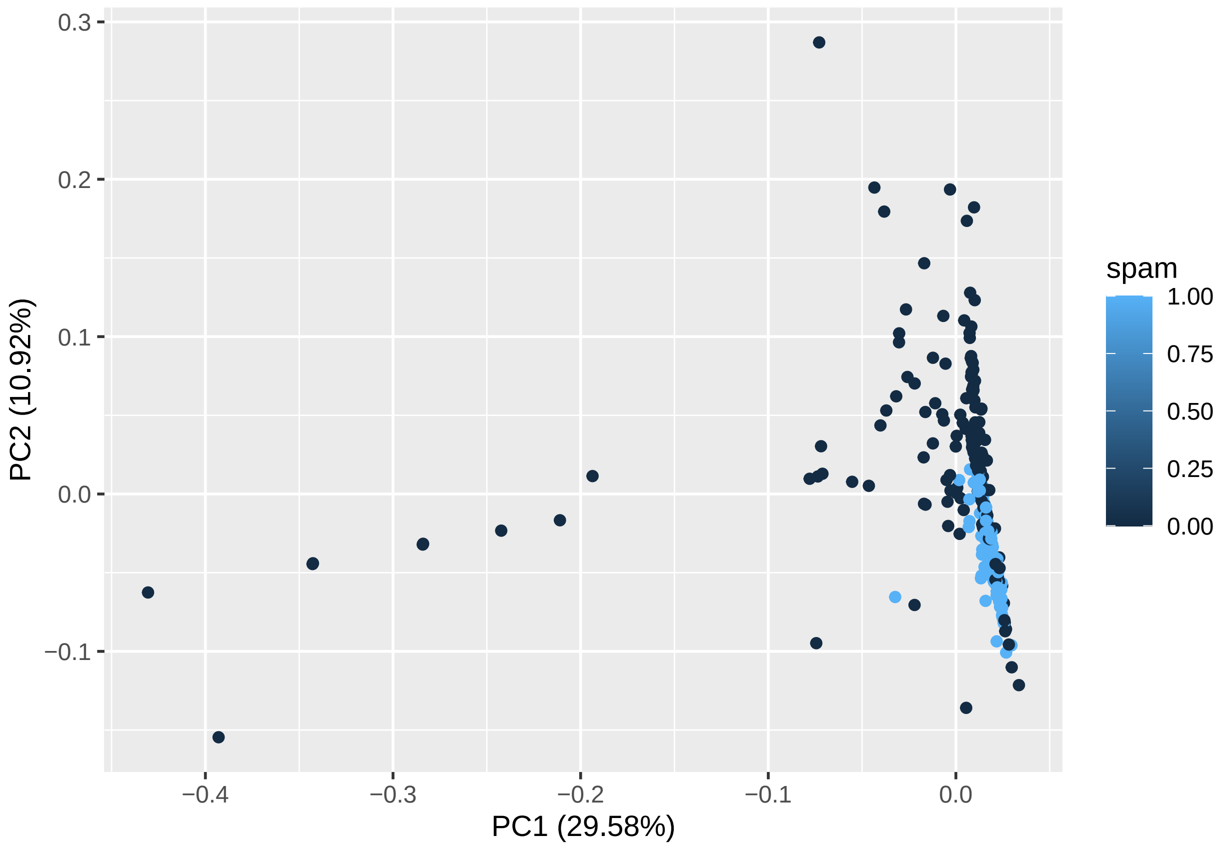
Please do the following:

(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.

```r
set.seed(123)

train.RData <- spam.df[sample(nrow(spam.df), 300, replace = F),
    ]
test.RData <- spam.df[sample(nrow(spam.df), 100, replace = F),
    ]
```

(3.b) Apply PCA to the training data "train.RData" and see if you find any pattern that can be used to approximately tell a spam email from a non-spam email.

```
pca2 <- prcomp(train.RData)
autoplot(pca2, data = train.RData, colour = "spam")
```



- The points on the graph are very similar in the most dense areas so it is difficult to determine whether emails are spam or not spam at this point.

(3.c) Use "train.RData" to build an SVM model with linear kernel, whose `cost` parameter is determined by 10-fold cross-validation, for which the features are predictors, the status of email is the response, and `cost` ranges in `c(0.01,0.1,1,5,10,50)`. Apply the obtained optimal model to "test.RData", and report via a 2-by-2 table on spams that are classified as spams or non-spams and on non-spams that are classified as non-spams or spams.

```r
set.seed(123)

tune.out1 <- tune(svm, spam ~ ., data = train.RData, kernel = "linear",
    ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 50)))
tune.out1$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = spam ~ ., data = train.RData, ranges = list(cost = c(0.01
##     0.1, 1, 5, 10, 50)), kernel = "linear")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  linear
##        cost:  0.01
##       gamma:  0.01818182
##     epsilon:  0.1
##
##
## Number of Support Vectors:  226
```

```r
summary(tune.out1)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost
##  0.01
##
## - best performance: 0.1184093
##
## - Detailed performance results:
##    cost     error dispersion
## 1  0.01 0.1184093 0.05314349
## 2  0.10 0.1447579 0.06668861
## 3  1.00 0.1532318 0.08903048
## 4  5.00 0.1528652 0.08874452
## 5 10.00 0.1508608 0.08630392
## 6 50.00 0.1511603 0.08547582
```

```r
predCV1 <- predict(tune.out1$best.model, test.RData, ranges = list(cost = c(0.01,
    0.1, 1, 5, 10, 50)))

dff <- data.frame(predCV1)
dff$pred[dff$predCV1 < 0] <- 0
dff$pred[dff$predCV1 > 0] <- 1

table(predicted = dff$pred, truth = test.RData$spam)
```

```
##          truth
## predicted  0  1
##         0 15  0
##         1 41 44
```

- 226 support vectors

(3.d) Use "train.RData" to build an SVM model with radial kernel, whose "cost" parameter is determined by 10-fold cross-validation, for which the features are predictors, the status of email is the response, `cost` ranges in `c(0.01,0.1,1,5,10,50)`, and `gamma=c(0.5,1,2,3,4)`. Report the number of support vectors. Apply the obtained optimal model to "test.RData", and report via a 2-by-2 table on spams that are classified as spams or non-spams and on non-spams that are classified as non-spams or spams.

```r
set.seed(123)

tune.out2 <- tune(svm, spam ~ ., data = train.RData, kernel = "radial",
    ranges = list(cost = c(0.01, 0.1, 1, 5, 10, 50), gamma = c(0.5,
        1, 2, 3, 4)))
tune.out2$best.model
```

```
##
## Call:
## best.tune(method = svm, train.x = spam ~ ., data = train.RData, ranges = list(cost = c(0.01
##     0.1, 1, 5, 10, 50), gamma = c(0.5, 1, 2, 3, 4)), kernel = "radial")
##
##
## Parameters:
##    SVM-Type:  eps-regression
##  SVM-Kernel:  radial
##        cost:  5
##       gamma:  0.5
##     epsilon:  0.1
##
##
## Number of Support Vectors:  297
```

```r
out1 = svm(spam ~ ., data = train.RData, kernel = "linear", cost = c(0.01,
    0.1, 1, 5, 10, 50))
summary(tune.out2)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  cost gamma
##     5   0.5
##
## - best performance: 0.2198271
##
## - Detailed performance results:
##     cost gamma     error dispersion
## 1   0.01   0.5 0.3489576 0.05319032
## 2   0.10   0.5 0.3285366 0.05003838
## 3   1.00   0.5 0.2212871 0.02042293
## 4   5.00   0.5 0.2198271 0.01799384
## 5  10.00   0.5 0.2198271 0.01799384
## 6  50.00   0.5 0.2198271 0.01799384
## 7   0.01   1.0 0.3491361 0.05320574
## 8   0.10   1.0 0.3302444 0.05015151
## 9   1.00   1.0 0.2309150 0.01863146
## 10  5.00   1.0 0.2291029 0.01501976
## 11 10.00   1.0 0.2291029 0.01501976
## 12 50.00   1.0 0.2291029 0.01501976
## 13  0.01   2.0 0.3492184 0.05322890
## 14  0.10   2.0 0.3310311 0.05036532
## 15  1.00   2.0 0.2356658 0.01881585
## 16  5.00   2.0 0.2340554 0.01441352
## 17 10.00   2.0 0.2340554 0.01441352
## 18 50.00   2.0 0.2340554 0.01441352
## 19  0.01   3.0 0.3492429 0.05324095
## 20  0.10   3.0 0.3312647 0.05048327
## 21  1.00   3.0 0.2371274 0.01922422
## 22  5.00   3.0 0.2356033 0.01455333
## 23 10.00   3.0 0.2356033 0.01455333
## 24 50.00   3.0 0.2356033 0.01455333
## 25  0.01   4.0 0.3492529 0.05324702
## 26  0.10   4.0 0.3313617 0.05054095
## 27  1.00   4.0 0.2377553 0.01949107
## 28  5.00   4.0 0.2362737 0.01470160
## 29 10.00   4.0 0.2362737 0.01470160
## 30 50.00   4.0 0.2362737 0.01470160
```

```
predCV2 <- predict(tune.out2$best.model, test.RData, ranges = list(cost = c(0.01,
    0.1, 1, 5, 10, 50)))

dfff <- data.frame(predCV2)
dfff$pred[dfff$predCV2 < 0] <- 0
dfff$pred[dfff$predCV2 > 0] <- 1

table(predicted = dfff$pred, truth = test.RData$spam)
```

```
##          truth
## predicted  0  1
##         1 56 44
```

- 297 support vectors

(3.e) Compare and comment on the classification results obtained by (3.c) and (3.d).

- The classification results obtained from the linear kernel show that the table is 59% accurate. The classification results obtained from the radial kernel show that the table is 56% accurate.