STA 380 Homework 2: Daan Rozenbroek, Daniel Lindquist, Kayla Torres

Kayla

August 15, 2016

Question 1

This is a plot of the mean departure delay (in minutes) for each day of the month/Day of the week. As well as for months in a year

```
ABIA <- read.csv('ABIA.csv')
dim(ABIA)
```

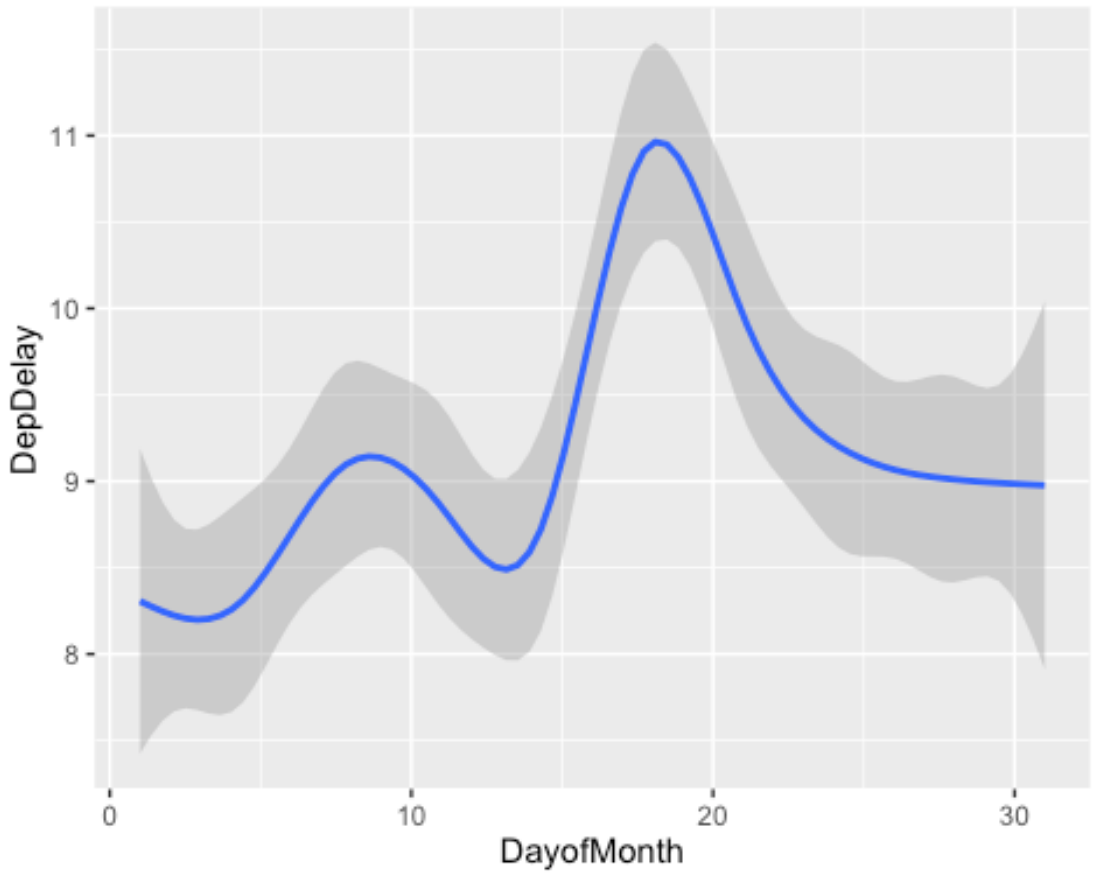
```
## [1] 99260    29
```

```
names(ABIA)
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay"
```

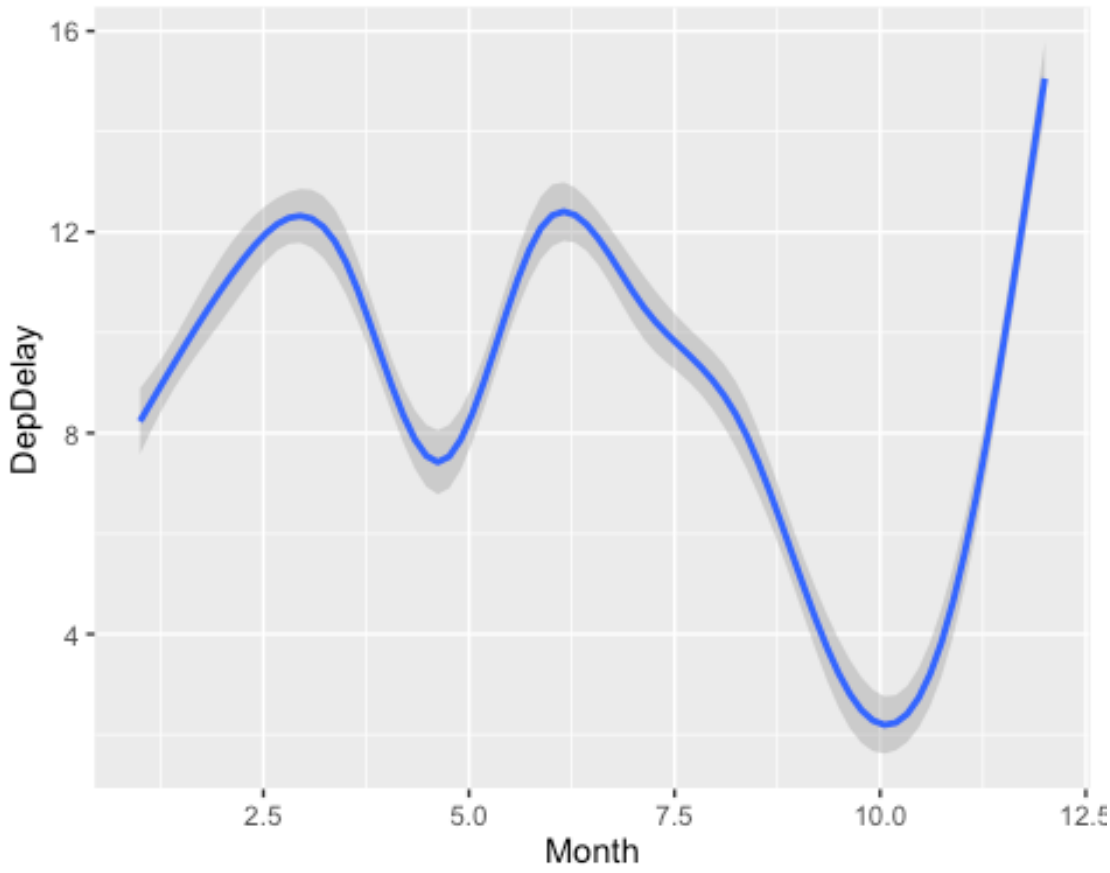
```
library(ggplot2)
ggplot(ABIA, aes(DayofMonth, DepDelay)) + geom_smooth()
```

```
## Warning: Removed 1413 rows containing non-finite values (stat_smooth).
```



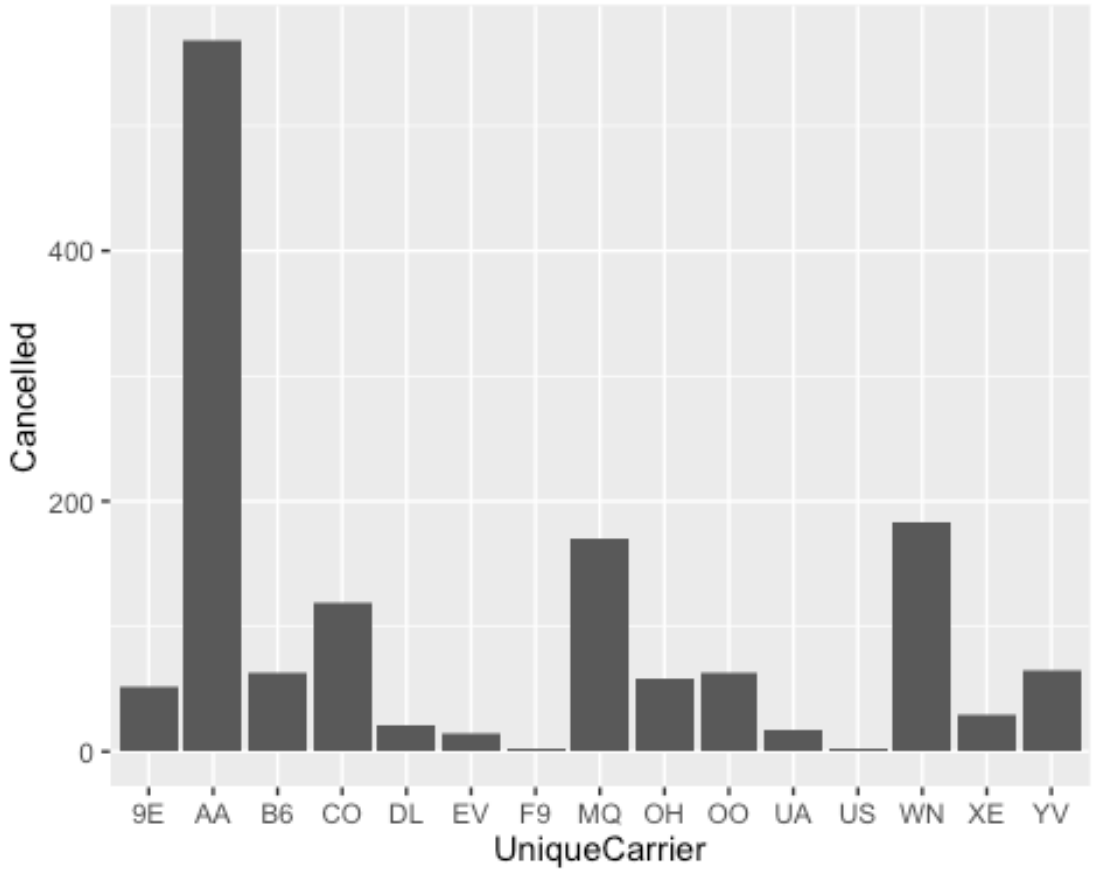

```
ggplot(ABIA, aes(Month, DepDelay)) + geom_smooth()
```

```
## Warning: Removed 1413 rows containing non-finite values (stat_smooth).
```



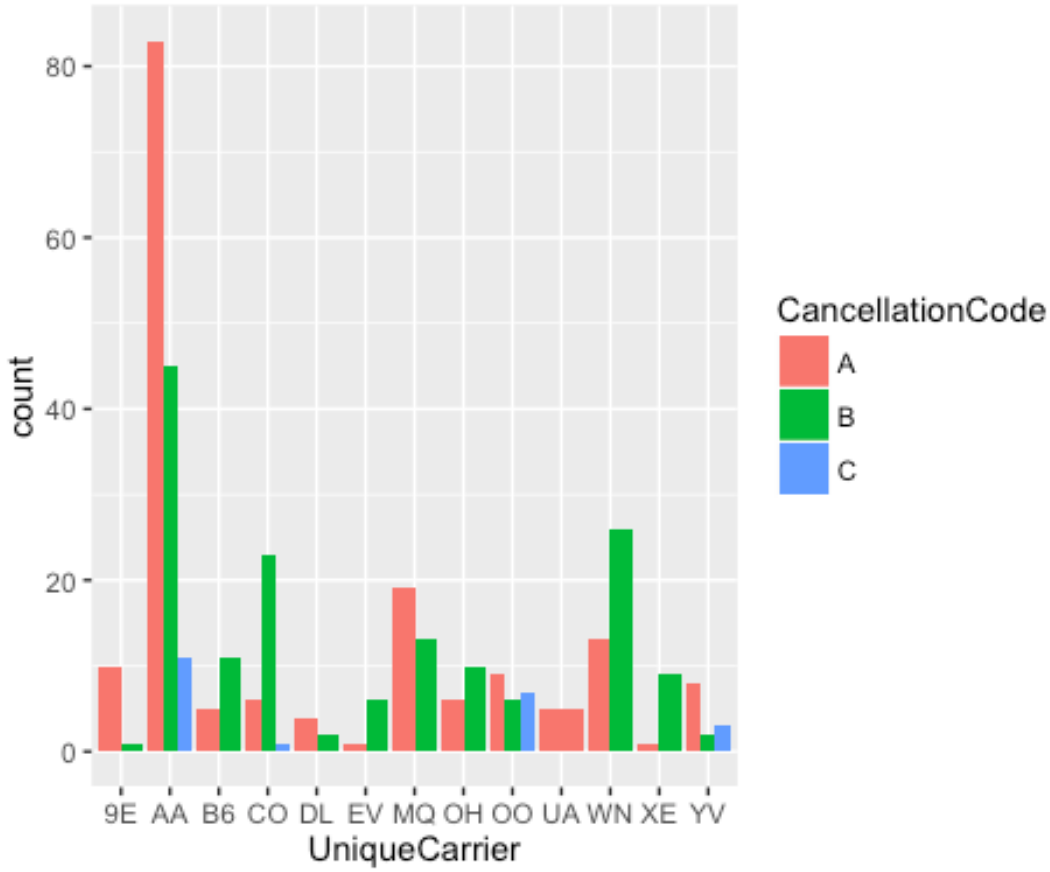
This is a plot of the number of cancelled flights by the flight carrier.

```
df1 <- ABIA[,c('UniqueCarrier', 'Cancelled')]
only_cancellations <- df1[df1$Cancelled == 1,]
ggplot(only_cancellations, aes(x = UniqueCarrier, y = Cancelled)) +
geom_bar(stat='identity')
```

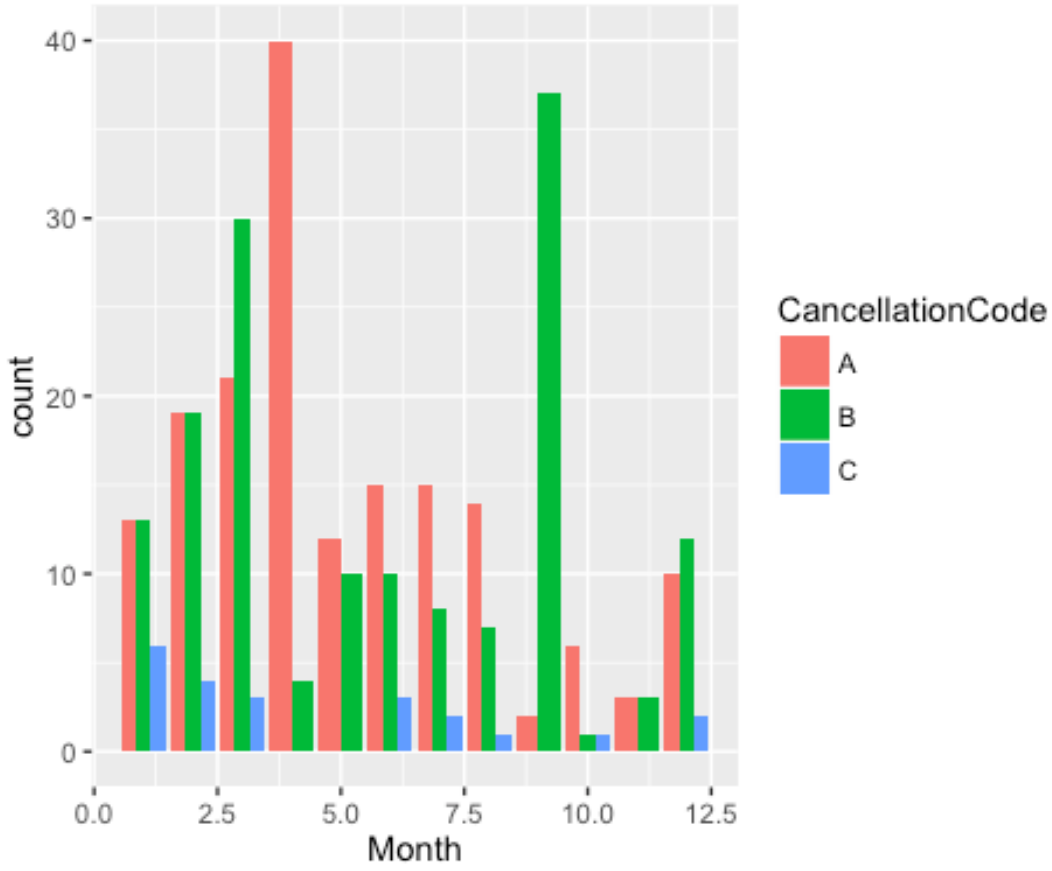


This is a plot of the different flight carriers and the count of cancellations and the specific reasons.

```
df2 <- ABIA[,c('UniqueCarrier', 'CancellationCode')]
df2_A <- df2[df2$CancellationCode == c('A','B','C','D'),]
plot <- ggplot(df2_A, aes(x=UniqueCarrier, fill=CancellationCode)) +
geom_bar(position='dodge')
plot
```

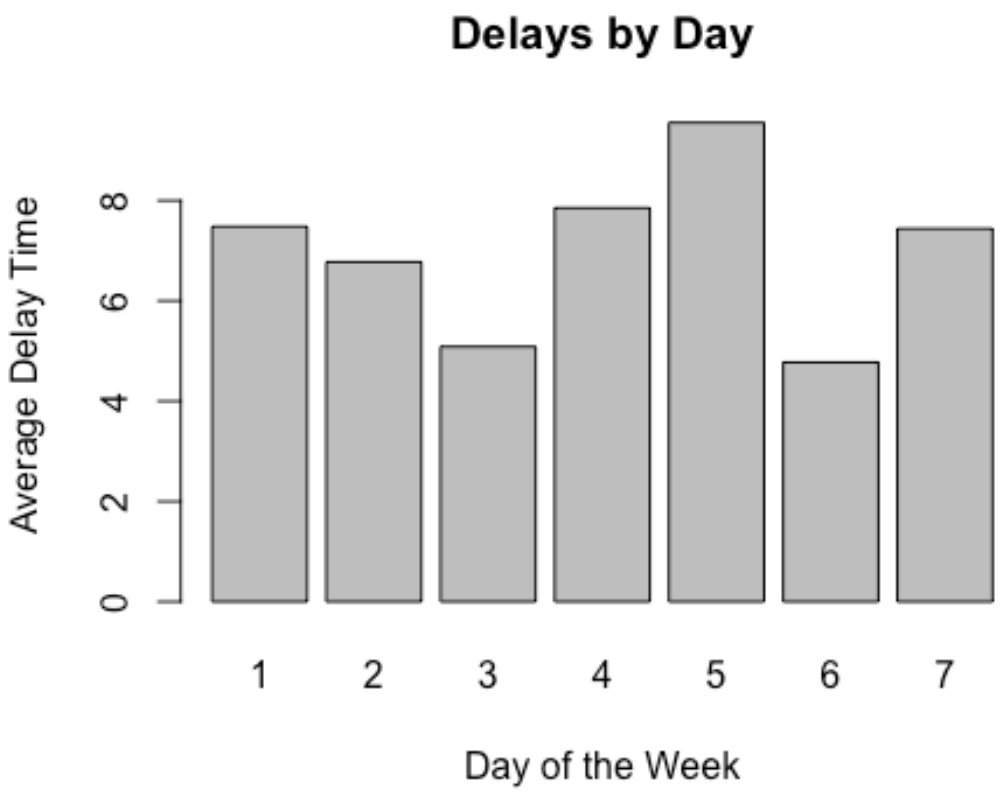



```
df3 <- ABIA[,c('Month', 'CancellationCode')]
df3_A <- df3[df3$CancellationCode == c('A','B','C','D'),]
plot2 <- ggplot(df3_A, aes(x=Month, fill=CancellationCode)) +
geom_bar(position='dodge')
plot2
```



This is a plot of the average delay time for every day of the week.

```
df <- ABIA[, c("ArrDelay","DayOfWeek")]
head(df)

##   ArrDelay DayOfWeek
## 1      339         2
## 2       -9         2
## 3       -1         2
## 4      -23         2
## 5       -6         2
## 6        2         2

a <- aggregate(ArrDelay ~ DayOfWeek, data=df, FUN=mean)
barplot(a$ArrDelay, names.arg=a$DayOfWeek, ylab = "Average Delay Time", xlab
= "Day of the Week", main = "Delays by Day")
```



Question 2

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(caret)
```

```
## Loading required package: lattice
```

```
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-5
```

```
library(SnowballC)
library(class)

readerPlain = function(fname){
                readPlain(elem=list(content=readLines(fname)),
                            id=fname, language='en') }

knitr::opts_chunk$set(echo = TRUE)

author_dirs1 = Sys.glob('ReutersC50/c50train/*')
author_dirs2 = Sys.glob('ReutersC50/c50test/*')
author_dirs = c(author_dirs1,author_dirs2)
file_list = NULL
labels = NULL

for(author in author_dirs) {
    author_name = substring(author, first=29)
    files_to_add = Sys.glob(paste0(author, '/*.txt'))
    file_list = append(file_list, files_to_add)
    labels = append(labels, rep(author_name, length(files_to_add)))
}

all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list


my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything
lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove
numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) #
remove punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove
excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords),
stopwords("SMART"))

my_corpus <- tm_map(my_corpus, stemDocument)

DTM = DocumentTermMatrix(my_corpus)

inspect(DTM[1:10,1:20])

## <<DocumentTermMatrix (documents: 10, terms: 20)>>
## Non-/sparse entries: 0/200
## Sparsity           : 100%
## Maximal term length: 7
```

```
## Weighting          : term frequency (tf)
##
##                                                    Terms
## Docs                                                aaa aaeu aah aama
##   ReutersC50/c50train/AaronPressman/106247newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt    0    0   0    0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt   0    0   0    0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt   0    0   0    0
##                                                    Terms
## Docs                                                aaminus aampa aampw
##   ReutersC50/c50train/AaronPressman/106247newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt        0     0     0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt       0     0     0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt       0     0     0
##                                                    Terms
## Docs                                                aan aaron aart aba
##   ReutersC50/c50train/AaronPressman/106247newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt    0     0    0   0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt   0     0    0   0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt   0     0    0   0
##                                                    Terms
## Docs                                                abadon abandon abat
##   ReutersC50/c50train/AaronPressman/106247newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt       0       0    0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt      0       0    0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt      0       0    0
```

```
##                                                     Terms
## Docs                                                 abb abba abbey abbott
##   ReutersC50/c50train/AaronPressman/106247newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt    0    0     0      0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt   0    0     0      0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt   0    0     0      0
##                                                     Terms
## Docs                                                 abbrevi abc
##   ReutersC50/c50train/AaronPressman/106247newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/120600newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/120683newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/136958newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/137498newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/14014newsML.txt        0   0
##   ReutersC50/c50train/AaronPressman/156814newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/182596newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/186392newsML.txt       0   0
##   ReutersC50/c50train/AaronPressman/193495newsML.txt       0   0
```

```
DTM = removeSparseTerms(DTM, 0.975)

X = as.data.frame(data.matrix(DTM))
names = row.names(X)
cat_col = c()
for (i in names){

  str = i

  list = strsplit(str, '/', fixed=FALSE)
  list= as.vector(list)

  list1 = matrix(unlist(list), ncol=4, byrow=TRUE)
  cat_col = c(cat_col,list1[1,3])

}
X[,"Category of Author"] <- cat_col
```

Using a KNN model

```
train <- X[1:2500,-1277]
test <- X[2501:5000,-1277]
cl <- X[, "Category of Author"]

knn.pred = knn(train, test, cl[1:2500]) #predicting the author names
```

```r
conf.mat <- table("Predictions" = knn.pred, Actual = cl[2501:5000])
#confusion matrix

accuracy <- sum(diag(conf.mat))/length(test) * 100
accuracy
```

```
## [1] 77.42947
```

```r
#using X, perform KNN
library(class)

# Split data by rownumber into two equal portions
set.seed(3)
train_sample = sample(1:2500,2500)
test_sample = sample(2501:5000, 2500)

train <- X[train_sample,] #don't include the column for category --- train
rows
test <- X[test_sample,]

# Isolate classifier
cl <- X[,"Category of Author"]

cat_train = cl[train_sample] #instead of cl[1:2500]
cat_test = cl[test_sample] #instead of cl[2501:5000]
#fit knn

knn.pred = knn(train[,-1277], test[,-1277], cat_train) #predicting the author
names

#knn.pred

conf.mat <- table("Predictions" = knn.pred, Actual = cat_test) #confusion
matrix

#conf.mat

accuracy <- sum(diag(conf.mat))/length(test) * 100
accuracy #percent accuracy
```

```
## [1] 77.13391
```

There is an accuracy of 77% when we predicted it on the out-of-sample data using the KNN model.

pca

```
dir1= Sys.glob('ReutersC50/C50train/*')
dir2=  Sys.glob('ReutersC50/C50test/*')



file_list_train = NULL #list of all the different text documents
labels_train = NULL #labels is the names of the authors
for(author in dir1) {
  author_name_train = substring(author, first=29)
  files_to_add_train = Sys.glob(paste0(author, '/*.txt'))
  file_list_train = append(file_list_train, files_to_add_train)
  labels_train = append(labels_train, rep(author_name_train,
length(files_to_add_train)))
}
file_list_test = NULL
labels_test = NULL
for(author in dir2) {
  author_name_test = substring(author, first=28)
  files_to_add_test = Sys.glob(paste0(author, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add_test)
  labels_test = append(labels_test, rep(author_name_test,
length(files_to_add_test)))
}

all_file <- append(file_list_train, file_list_test)
all_label <- unique(append(labels_train, labels_test))

all_docs = lapply(all_file, readerPlain)
names(all_docs) = all_file
names(all_docs) = sub('.txt', '', names(all_docs))

my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = all_label



my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything
lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove
numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) #
remove punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove
excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords),
stopwords("en"))
```

```
DTM <- DocumentTermMatrix(my_corpus)
DTM
```

```
## <<DocumentTermMatrix (documents: 5000, terms: 44565)>>
## Non-/sparse entries: 987658/221837342
## Sparsity           : 100%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

```
DTM <- removeSparseTerms(DTM, 0.975) #this only includes the top 97.5 of word
frequency

x <- as.matrix(DTM)
x_train <- x[1:2500,] #this splits the data in half
x_test <- x[2501:5000,]
smoothing <- 1/nrow(x_train)

pca_authors <- prcomp(x)
transformed <- pca_authors$x[,1:300]
transformed_train <- transformed[1:2500,]
transformed_test <- transformed[2501:5000,]
logistical_model <- glmnet(y = rep(1:50, each=50), x = transformed_train,
family='multinomial', alpha=0.1)
author_prediction <- as.data.frame(predict(logistical_model, newx =
transformed_train, type='class', s=0.05))
author_prediction$actual_authors <- rep(1:50, each =50)
confusionMatrix(author_prediction$`1`,author_prediction$actual_authors)
```

```
## Warning in confusionMatrix.default(author_prediction$`1`,
author_prediction
## $actual_authors): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##         1  46  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         2   0 45  0  0  0  0  0  0  0  1  0  0  0  2  1  0  0  0  3  0  0
##         3   0  0 44  0  2  0  0  0  0  0  0  0  1  0  0  0  0  3  0  0  0
##         4   0  0  0 38  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0
##         5   0  0  0  0 33  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         6   0  0  0  0  0 38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         7   0  0  0  0  0  0 43  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         8   0  0  0  0  0  1  0 39  0  0  0  0  0  0  0  0  0  0  0  0  0
##         9   0  0  0  0  0  0  0  0 36  0  0  0  0  0  0  0  0  1  1  0
##         10  0  0  0  0  0  0  0  0  0 37  0  0  0  0  0  0  0  0  0  0  0
##         11  0  0  0  2  0  0  0  0  0  0 50  0  0  0  0  0  0  0  0  0  0
##         12  0  0  0  0  0  0  0  0  0  0  0 48  0  0  1  0  0  0  0  0  0
##         13  0  0  0  0  1  0  0  0  0  0  0  0 43  0  0  0  0  0  0  0  0
##         14  0  2  0  0  0  0  0  0  0  0  0  0  0 36  0  0  0  0  5  0  0
```

```
##          15  0  0  0  3  0  0  0  0  0  0  0  0  0  0 36  0  0  0  0  0  1
##          16  0  0  0  0  1  0  0  2  2  0  0  0  0  0  0 50  0  0  0  0  0
##          17  0  0  2  0  0  0  0  0  3  0  0  0  0  0  0  0 46  1  0  3  0
##          18  0  0  1  0  1  0  1  0  0  0  0  0  0  0  0  0  1 40  0  0  1
##          19  0  2  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0 41  0  0
##          20  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0 42  0
##          21  2  0  0  0  0  0  0  0  0  1  0  0  3  0  0  0  0  1  0  0 44
##          22  0  0  1  0  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  1  0
##          23  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          24  0  0  0  0  7  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##          25  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          26  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0
##          27  0  0  0  0  0  0  2  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##          28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          29  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          31  0  0  0  0  3  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##          32  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          33  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  1
##          34  0  0  0  0  0  0  0  0  0  0  0  1  1  0  0  0  0  1  0  0  0
##          35  0  0  0  5  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0
##          36  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
##          37  0  0  1  0  0  2  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
##          38  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
##          39  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##          40  0  0  0  0  1  2  3  2  0  0  0  0  0  0  0  0  1  0  0  0  0
##          41  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          42  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##          43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  2
##          44  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0
##          46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          47  1  0  0  0  0  1  0  0  0  3  0  1  0  0  0  0  0  0  0  0  0
##          48  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  1  0  0  3  0
##          49  0  0  0  0  0  3  0  7  3  0  0  0  0  0  0  0  0  0  0  0  0
##          50  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0
##            Reference
## Prediction 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42
##          1   0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          3   1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          4   0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  1  0  0  0  0
##          5   0  0  5  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##          6   0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##          7   0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0
##          8   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          9   1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##          10  0  1  0  0  0  0  0  0  0  0  2  0  0  0  1  1  0  0  0  0  3
##          11  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##          12  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
```

```
##          13  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          14  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          15  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##          16  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  1  0  0  0
##          17  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          18  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##          19  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          20  0  2  0  1  0  0  0  0  1  0  0  0  1  0  0  0  0  0  0  0  0
##          21  0  2  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0  0  1  0  0
##          22 38  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          23  0 31  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0
##          24  0  0 43  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
##          25  5  0  0 34  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
##          26  0  1  0  0 47  0  0  0  0  0  1  0  0  0  0  0  0  0  0  1  1
##          27  0  0  0  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          28  0  0  0  0  0  0 49  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##          29  0  0  0  0  0  0  0 50  0  0  0  0  0  0  0  0  0  0  0  0  0
##          30  0  0  0  0  0  0  0  0 48  0  0  0  0  0  0  0  0 13  0  0  0
##          31  0  0  0  0  0  0  0  0  0 48  0  0  0  1  0  0  0  0  0  0  0
##          32  0  2  0  0  1  0  0  0  0  0 43  0  0  0  1  1  0  1  0  0  0
##          33  0  0  0  0  0  1  0  0  0  0  0 50  0  0  0  1  0  0  0  0  0
##          34  0  1  0  0  0  0  0  0  0  0  0  0 43  0  0  0  0  0  0  0  1
##          35  0  0  0  0  0  0  0  0  0  0  0  0  0 37  0  1  1  0  0  0  0
##          36  4  0  0  1  0  0  0  0  0  0  0  0  1  0 45  0  0  0  0  1  0
##          37  1  3  0  0  0  0  0  0  0  0  1  0  1  0  0 41  0  0  1  0  1
##          38  0  0  0  0  0  0  1  0  0  0  0  0  0  2  0  0 46  0  0  0  0
##          39  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 35  0  0  0
##          40  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 48  0  0
##          41  0  0  0  0  0  0  0  0  0  0  1  0  0  0  2  0  0  0  0 46  0
##          42  0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 40
##          43  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          44  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0
##          45  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##          46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0
##          47  0  0  1  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  1  3
##          48  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0
##          49  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          50  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##            Reference
## Prediction 43 44 45 46 47 48 49 50
##          1   0  0  1  0  0  0  0  0
##          2   0  1  0  0  0  0  0  0
##          3   0  0  2  0  0  0  0  0
##          4   0  8  0  0  0  0  0  5
##          5   0  0  0  0  0  0  0  0
##          6   1  0  0  0  1  0  0  0
##          7   0  0  0  0  0  0  0  0
##          8   0  0  0  0  0  0 11  0
##          9   0  0  0  1  0  0  0  0
##          10  0  0  0  0  4  0  0  0
```

```
##         11  0  0  0  0  0  0  0  1
##         12  1  1  0  1  0  0  0  3
##         13  0  0  0  0  0  0  0  0
##         14  0  0  0  0  0  0  0  0
##         15  0  6  0  1  0  0  0  2
##         16  0  0  0  1  1  2  0  0
##         17  0  0  1  0  0  1  0  0
##         18  0  0  3  0  0  0  0  0
##         19  0  0  0  0  0  0  0  0
##         20  0  0  0  0  0  0  0  0
##         21  0  1  0  0  0  1  2  0
##         22  0  0  1  0  0  0  0  0
##         23  0  0  1  0  0  0  1  0
##         24  0  0  0  0  0  0  0  0
##         25  0  0  1  0  0  1  0  0
##         26  0  1  0  0  4  0  0  0
##         27  0  0  0  0  0  0  0  0
##         28  0  0  0  1  0  0  0  0
##         29  0  0  1  1  0  0  0  1
##         30  0  0  0  0  0  0  0  0
##         31  0  0  0  1  0  0  0  0
##         32  0  0  0  0  0  0  0  0
##         33  0  1  0  0  0  0  0  0
##         34  0  0  1  0  0  0  0  0
##         35  0  9  0  1  0  0  0  1
##         36  0  0  0  0  0  0  0  0
##         37  1  0  0  0  0  1  0  0
##         38  2  4  0 21  0  0  0  0
##         39  0  0  0  0  0  0  0  0
##         40  1  0  0  0  0  0  0  1
##         41  0  0  0  0  0  0  0  0
##         42  0  0  0  0  2  0  0  0
##         43 43  0  0  1  0  0  0  4
##         44  0  9  0  0  0  0  0  0
##         45  0  0 38  0  0  1  0  0
##         46  1  1  0 20  0  0  0  1
##         47  0  0  0  0 38  0  0  0
##         48  0  0  0  0  0 43  0  0
##         49  0  0  0  0  0  0 36  0
##         50  0  8  0  0  0  0  0 31
##
## Overall Statistics
##
##                Accuracy : 0.8156
##                  95% CI : (0.7998, 0.8306)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.8118
##  Mcnemar's Test P-Value : NA
```

```
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.9200   0.9000   0.8800   0.7600   0.6600   0.7600
## Specificity            0.9988   0.9967   0.9963   0.9910   0.9971   0.9976
## Pos Pred Value         0.9388   0.8491   0.8302   0.6333   0.8250   0.8636
## Neg Pred Value         0.9984   0.9980   0.9975   0.9951   0.9931   0.9951
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0184   0.0180   0.0176   0.0152   0.0132   0.0152
## Detection Prevalence   0.0196   0.0212   0.0212   0.0240   0.0160   0.0176
## Balanced Accuracy      0.9594   0.9484   0.9382   0.8755   0.8286   0.8788
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.8600   0.7800   0.7200    0.7400    1.0000
## Specificity            0.9980   0.9951   0.9976    0.9951    0.9984
## Pos Pred Value         0.8958   0.7647   0.8571    0.7551    0.9259
## Neg Pred Value         0.9971   0.9955   0.9943    0.9947    1.0000
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0172   0.0156   0.0144    0.0148    0.0200
## Detection Prevalence   0.0192   0.0204   0.0168    0.0196    0.0216
## Balanced Accuracy      0.9290   0.8876   0.8588    0.8676    0.9992
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.9600    0.8600    0.7200    0.7200    1.0000
## Specificity             0.9967    0.9992    0.9971    0.9943    0.9951
## Pos Pred Value          0.8571    0.9556    0.8372    0.7200    0.8065
## Neg Pred Value          0.9992    0.9971    0.9943    0.9943    1.0000
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0192    0.0172    0.0144    0.0144    0.0200
## Detection Prevalence    0.0224    0.0180    0.0172    0.0200    0.0248
## Balanced Accuracy       0.9784    0.9296    0.8586    0.8571    0.9976
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.9200    0.8000    0.8200    0.8400    0.8800
## Specificity             0.9931    0.9963    0.9951    0.9971    0.9935
## Pos Pred Value          0.7302    0.8163    0.7736    0.8571    0.7333
## Neg Pred Value          0.9984    0.9959    0.9963    0.9967    0.9975
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0184    0.0160    0.0164    0.0168    0.0176
## Detection Prevalence    0.0252    0.0196    0.0212    0.0196    0.0240
## Balanced Accuracy       0.9565    0.8982    0.9076    0.9186    0.9367
##                      Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity             0.7600    0.6200    0.8600    0.6800    0.9400
## Specificity             0.9976    0.9984    0.9959    0.9959    0.9951
## Pos Pred Value          0.8636    0.8857    0.8113    0.7727    0.7966
## Neg Pred Value          0.9951    0.9923    0.9971    0.9935    0.9988
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0152    0.0124    0.0172    0.0136    0.0188
## Detection Prevalence    0.0176    0.0140    0.0212    0.0176    0.0236
## Balanced Accuracy       0.8788    0.8092    0.9280    0.8380    0.9676
##                      Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity             0.8800    0.9800    1.0000    0.9600    0.9600
```

```
## Specificity             0.9988    0.9992    0.9984    0.9947    0.9976
## Pos Pred Value          0.9362    0.9608    0.9259    0.7869    0.8889
## Neg Pred Value          0.9976    0.9996    1.0000    0.9992    0.9992
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0176    0.0196    0.0200    0.0192    0.0192
## Detection Prevalence    0.0188    0.0204    0.0216    0.0244    0.0216
## Balanced Accuracy       0.9394    0.9896    0.9992    0.9773    0.9788
##                      Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity             0.8600    1.0000    0.8600    0.7400    0.9000
## Specificity             0.9967    0.9976    0.9976    0.9910    0.9963
## Pos Pred Value          0.8431    0.8929    0.8776    0.6271    0.8333
## Neg Pred Value          0.9971    1.0000    0.9971    0.9947    0.9980
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0172    0.0200    0.0172    0.0148    0.0180
## Detection Prevalence    0.0204    0.0224    0.0196    0.0236    0.0216
## Balanced Accuracy       0.9284    0.9988    0.9288    0.8655    0.9482
##                      Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity             0.8200    0.9200    0.7000    0.9600    0.9200
## Specificity             0.9943    0.9873    0.9992    0.9951    0.9984
## Pos Pred Value          0.7455    0.5974    0.9459    0.8000    0.9200
## Neg Pred Value          0.9963    0.9983    0.9939    0.9992    0.9984
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0164    0.0184    0.0140    0.0192    0.0184
## Detection Prevalence    0.0220    0.0308    0.0148    0.0240    0.0200
## Balanced Accuracy       0.9071    0.9537    0.8496    0.9776    0.9592
##                      Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity             0.8000    0.8600    0.1800    0.7600    0.4000
## Specificity             0.9980    0.9967    0.9992    0.9971    0.9980
## Pos Pred Value          0.8889    0.8431    0.8182    0.8444    0.8000
## Neg Pred Value          0.9959    0.9971    0.9835    0.9951    0.9879
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0160    0.0172    0.0036    0.0152    0.0080
## Detection Prevalence    0.0180    0.0204    0.0044    0.0180    0.0100
## Balanced Accuracy       0.8990    0.9284    0.5896    0.8786    0.6990
##                      Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity             0.7600    0.8600    0.7200    0.6200
## Specificity             0.9947    0.9955    0.9939    0.9947
## Pos Pred Value          0.7451    0.7963    0.7059    0.7045
## Neg Pred Value          0.9951    0.9971    0.9943    0.9923
## Prevalence              0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0152    0.0172    0.0144    0.0124
## Detection Prevalence    0.0204    0.0216    0.0204    0.0176
## Balanced Accuracy       0.8773    0.9278    0.8569    0.8073
```

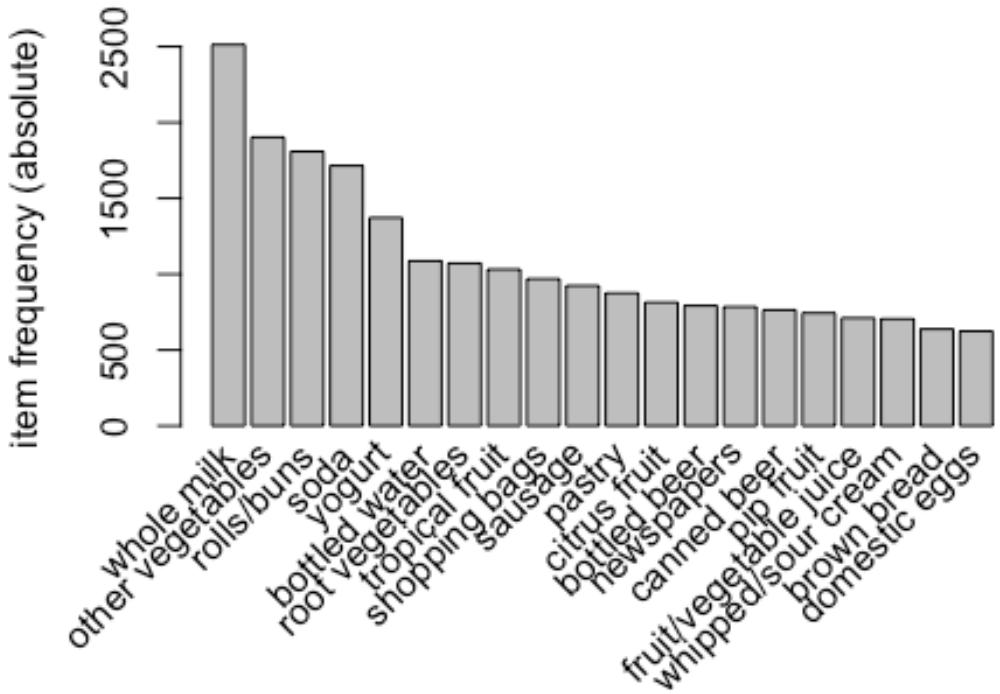
Using pca we got a accuracy prediction of 81.6%. There are some authors, such as the 38th author, who had 21 predictions that the document belonged to the 46th author. This could because the 38th author and 46th author have similar writing styles or documents. They could have worked together as well. We would prefer the pca model because we got a stronger prediction accuracy.

Question 3

```
library(arules)

##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:base':
##
##     abbreviate, write

library(arulesViz)

## Loading required package: grid

library(datasets)

groceries <- read.transactions(file='groceries.txt', format='basket',
sep=',')
inspect(groceries[1:5])

##   items
## 1 {citrus fruit,
##    margarine,
##    ready soups,
##    semi-finished bread}
## 2 {coffee,
##    tropical fruit,
##    yogurt}
## 3 {whole milk}
## 4 {cream cheese,
##    meat spreads,
##    pip fruit,
##    yogurt}
## 5 {condensed milk,
##    long life bakery product,
##    other vegetables,
##    whole milk}
```

this is an item frequency plot of the top 20 items

```
itemFrequencyPlot(groceries, topN=20, type='absolute')
```



Defining the rules; association rule mining. Given a set of transactions, find rules that will predict the occurance of an item based on the occurances of other items. Support of 0.01 is a 0.001 fraction of transactions that contain an itemset. The frequency of each item set that we are evaluating is bigger than 0.001. Confidence us the measure of how often items in Y appear in transactions that contain X. Which we are setting at 0.8, so we only want transactions that have a likelihood of greater than 80%.

```
rules <- apriori(groceries, parameter = list(support = 0.001, confidence =
0.8, maxlen=4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.8    0.1    1 none FALSE            TRUE   0.001      1      4
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
```

```
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.01s].
## writing ... [258 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
options(digits=2)
inspect(rules[1:5])
```

```
##   lhs                          rhs            support confidence lift
## 1 {liquor,red/blush wine} => {bottled beer} 0.0019  0.90       11.2
## 2 {cereals,curd}          => {whole milk}   0.0010  0.91        3.6
## 3 {cereals,yogurt}        => {whole milk}   0.0017  0.81        3.2
## 4 {butter,jam}            => {whole milk}   0.0010  0.83        3.3
## 5 {bottled beer,soups}    => {whole milk}   0.0011  0.92        3.6
```

Output of this reads that for example if someone buys cereal and yogurt, there is a 81% likeliness that they'll also purchase whole milk. This combination has a .0017 chance of existing. We also made the rules more concise so that there aren't lists that are excessively long by putting a max length on the rules.

```
summary(rules)
```

```
## set of 258 rules
##
## rule length distribution (lhs + rhs):sizes
##   3   4
##  29 229
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##     3.0     4.0     4.0     3.9     4.0     4.0
##
## summary of quality measures:
##     support          confidence        lift
##  Min.   :0.00102   Min.   :0.80   Min.   : 3.1
##  1st Qu.:0.00102   1st Qu.:0.82   1st Qu.: 3.3
##  Median :0.00122   Median :0.85   Median : 3.6
##  Mean   :0.00127   Mean   :0.86   Mean   : 3.9
##  3rd Qu.:0.00132   3rd Qu.:0.90   3rd Qu.: 4.3
##  Max.   :0.00285   Max.   :1.00   Max.   :11.2
##
## mining info:
##       data ntransactions support confidence
##  groceries          9835   0.001        0.8
```

we see that there are 258 rules generated. We also see that 4 is the highest rule length means they are 4 items long. We can see the ranges in support, lift, and confidence. We can see that the total data that was mined was 9835.

sorting the rules by most likely rules.

```
rules<-sort(rules, by="confidence", decreasing=TRUE)
options(digits=2)
inspect(rules[1:5])
```

```
##   lhs                     rhs          support confidence lift
## 1 {rice,
##    sugar}              => {whole milk}  0.0012          1  3.9
## 2 {canned fish,
##    hygiene articles}   => {whole milk}  0.0011          1  3.9
## 3 {butter,
##    rice,
##    root vegetables}    => {whole milk}  0.0010          1  3.9
## 4 {flour,
##    root vegetables,
##    whipped/sour cream} => {whole milk}  0.0017          1  3.9
## 5 {butter,
##    domestic eggs,
##    soft cheese}        => {whole milk}  0.0010          1  3.9
```

remove rududant repetative rules

```
subset.matrix <- is.subset(rules, rules)
subset.matrix[lower.tri(subset.matrix, diag=T)] <- NA
redundant <- colSums(subset.matrix, na.rm=T) >= 1
rules.pruned <- rules[!redundant]
rules<-rules.pruned
```

now we can target items, we will find out what items are most likely to be bought before purchasing a certain item. We also answer the question of what customers are likely to buy after they purchase a specific item.

These items are most likely to be purchsed before a customer buys soda.

```
rules<-apriori(data=groceries, parameter=list(supp=0.001,conf = 0.08,
minlen=3),
               appearance = list(default="lhs",rhs="soda"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##   lhs                   rhs    support confidence lift
## 1 {coffee,
##    misc. beverages} => {soda}  0.0010       0.77  4.4
## 2 {bottled water,
##    newspapers,
##    rolls/buns,
##    yogurt}          => {soda}  0.0010       0.77  4.4
## 3 {bottled beer,
##    bottled water,
```

```
##    sausage}         => {soda}  0.0011       0.73  4.2
## 4 {sausage,
##    shopping bags,
##    white bread}     => {soda}  0.0010       0.67  3.8
## 5 {bottled water,
##    chocolate,
##    rolls/buns}      => {soda}  0.0013       0.65  3.7
```

This is what customers are likely to buy after soda. We can see that the most common items bought after buying soda are whole milk, rolls/buns, other vegetables, bottled water and yogurt. These are the top 5 most likely items that would be purchsed.

```
rules<-apriori(data=groceries, parameter=list(supp=0.001,conf =
0.15,minlen=2),
               appearance = list(default="rhs",lhs="soda"),
               control = list(verbose=F))
rules<-sort(rules, decreasing=TRUE,by="confidence")
inspect(rules[1:5])
```

```
##   lhs       rhs                support confidence lift
## 5 {soda} => {whole milk}       0.040   0.23       0.90
## 3 {soda} => {rolls/buns}       0.038   0.22       1.20
## 4 {soda} => {other vegetables} 0.033   0.19       0.97
## 1 {soda} => {bottled water}    0.029   0.17       1.50
## 2 {soda} => {yogurt}           0.027   0.16       1.12
```

This makes sense to me because people are always buying other beverages when they buy soda and other items that would be purchased for a cook-out or BBQ.

here is a visualization of the rules, showing for each item what is the most likely next items to purchase.

```
plot(rules,method="graph",interactive=TRUE,shading=NA)
```