Solutions for Exercises 2

Vijayakumar Ganapathy

August 18, 2015

Question 1: Flights at ABIA

Objective: To create a figure, or set of related figures, that tell an interesting story about flights into and out of Austin.

```
# Reading the data
abia <- read.csv("ABIA.csv", header = TRUE)

# Extracting the useful columns
abia <-
abia[,c("Month","DayofMonth","DayOfWeek","DepTime","ArrTime","ArrDelay","DepD
elay","Origin","Dest","Cancelled","CancellationCode")]

# Manipulating the data and creating derived columns
abia$arr_dep[abia$Origin=="AUS"]="Departure"
abia$arr_dep[abia$Dest=="AUS"]="Arrival"
abia$delay[abia$Origin=="AUS"]=abia$DepDelay[abia$Origin=="AUS"]
abia$delay[abia$Dest=="AUS"]=abia$ArrDelay[abia$Dest=="AUS"]
abia$hour<-NA
abia$hour[abia$Origin=="AUS"]=ceiling(abia$DepTime[abia$Origin=="AUS"]/100)
abia$hour[abia$Dest=="AUS"]=ceiling(abia$ArrTime[abia$Dest=="AUS"]/100)
```

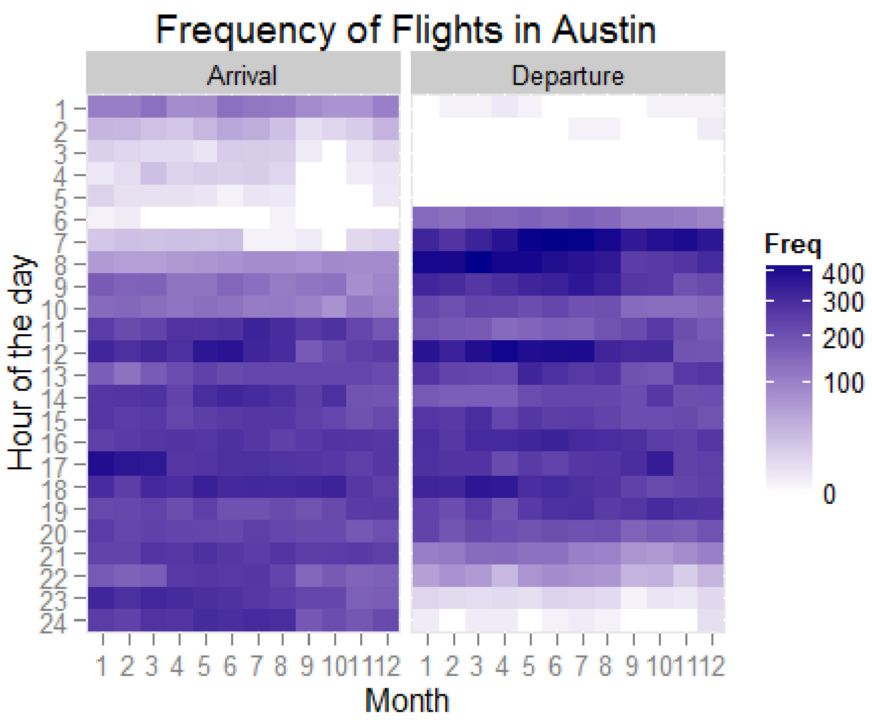
Let us first look at the frequency plot- how many flights fly per each hour on a monthly basis.

```
abia1 <- na.omit(abia[,c("arr_dep","delay","hour", "Month")])

# creating the pivot table
abia_tab <- xtabs(~hour + Month + arr_dep, abia1)
abia_tab <- as.data.frame.table(abia_tab)

library(ggplot2)

# plotting the frequency across months controlled for time of the day
plt <- ggplot(abia_tab, aes(Month, hour, fill = Freq)) + facet_grid(~arr_dep)
+ geom_tile() + ylab("Hour of the day") + ggtitle("Frequency of Flights in
Austin") + scale_fill_gradient( trans="sqrt", low = "white", high="dark
blue")
plt  + scale_y_discrete(limits=rev(levels(abia_tab$hour)))
```



The above figure shows the frequency of flights in Austin varying across months controlled for hour of the day.

Now let us look at the number of flights cancelled due to a particular reason on a monthly basis. Based on Wiki the 3 types of cancellations mean the following:

- **Carrier Delay:** Carrier delay is within the control of the air carrier. Examples of occurrences that may determine carrier delay are: aircraft cleaning, aircraft damage, awaiting the arrival of connecting passengers or crew, baggage, bird strike, cargo loading, catering, computer, outage-carrier equipment, crew legality (pilot or attendant rest), damage by hazardous goods, engineering inspection, fueling, handling disabled passengers, late crew, lavatory servicing, maintenance, oversales, potable water servicing, removal of unruly passenger, slow boarding or seating, stowing carry-on baggage, weight and balance delays.
- **NAS Delay:** Delay that is within the control of the National Airspace System (NAS) may include: non-extreme weather conditions, airport operations, heavy traffic volume, air traffic control, etc. Delays that occur after Actual Gate Out are usually attributed to the NAS and are also reported through OPSNET.
- **Weather Delay:** Weather delay is caused by extreme or hazardous weather conditions that are forecasted or manifest themselves on point of departure, enroute, or on point of arrival.

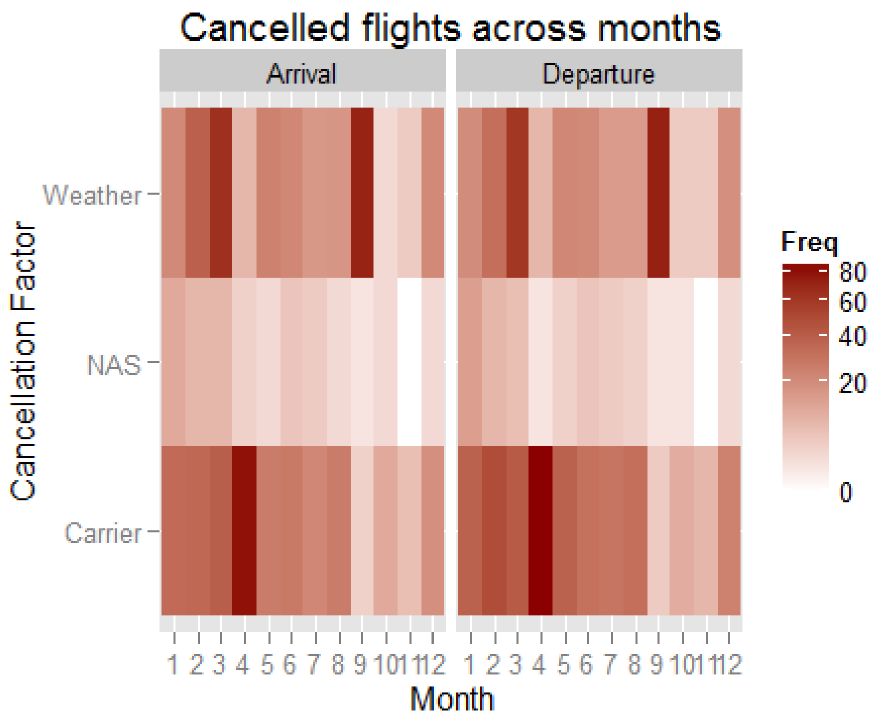
```
# Extracting the useful columns
abia2 <- na.omit(abia[,c("arr_dep","Cancelled","CancellationCode","Month")])
```

```r
# Removing all non-cancelled flights
abia2 <- abia2[abia2$Cancelled==1,]

# Creating pivot table
abia_tab <- xtabs(~CancellationCode + Month + arr_dep, abia2)
abia_tab <- as.data.frame.table(abia_tab)
abia_tab <- abia_tab[abia_tab$CancellationCode!="",]

# Adding Cancellation reasons
abia_tab$CancellationReason[abia_tab$CancellationCode=="A"] <- "Carrier"
abia_tab$CancellationReason[abia_tab$CancellationCode=="B"] <- "Weather"
abia_tab$CancellationReason[abia_tab$CancellationCode=="C"] <- "NAS"

# plotting the number of cancelled flights across months
plt <- ggplot(abia_tab, aes(Month, CancellationReason, fill = Freq)) +
facet_grid(~arr_dep) + geom_tile() + ylab("Cancellation Factor") +
ggtitle("Cancelled flights across months") + scale_fill_gradient(
trans="sqrt", low = "white", high="dark red")
plt
```



The above figure shows the number of cancelled flights in Austin across months due to different factors

Conclusion:

I have created 2 figures - one showing the frequency of flights across months controlled for time of the day, the second one showing the number of cancelled flights in Austin across months due to various reasons. Both theses plots are straight forward and readable. The legends, axis labels and the titles along with the figures themselves should convey the takeaways from these plots.

Question 2: Author attribution

Objective: To build two separate models for predicting the author of an article on the basis of that article's textual content.

The first part of this exercise derives a lot of its content from NaiveBayes.R which was discussed in class.

Let us first create the readerPlain function which will help us read the files.

```r
library(tm)

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

Let us first create the training corpus

```r
author_dirs_train = Sys.glob('ReutersC50/C50train/*')
file_list_train = NULL
labels_train = NULL

for(author in author_dirs_train) {
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_train = append(file_list_train, files_to_add)
  author_name = substring(author, first=21)
  labels_train = append(labels_train, rep(author_name, length(files_to_add)))
}
```

In a very similar manner, let us first create the test corpus

```r
author_dirs_test = Sys.glob('ReutersC50/C50test/*')

file_list_test = NULL
labels_test = NULL

for(author in author_dirs_test) {
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add)
  author_name = substring(author, first=20)
  labels_test = append(labels_test, rep(author_name, length(files_to_add)))
}
```

To deal with words in the test set we never saw in the training set, I am combining both training and test datasets which will later be used to create a single document term matrix with all words from training and test.

```
file_lists = append(file_list_train,file_list_test)
labels = NULL
labels <- unique(append(labels_train, labels_test))

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

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

# Preprocessing
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"))

DTM = DocumentTermMatrix(my_corpus)
DTM

## <<DocumentTermMatrix (documents: 5000, terms: 44234)>>
## Non-/sparse entries: 858721/220311279
## Sparsity           : 100%
## Maximal term length: 45
## Weighting          : term frequency (tf)
```

We can see that the sparsity of the document is very high. Let us now remove all elements which have more than a sparse factor of 0.975.

```
inspect(DTM[1:10,1:5])

## <<DocumentTermMatrix (documents: 10, terms: 5)>>
## Non-/sparse entries: 0/50
## Sparsity           : 100%
## Maximal term length: 7
## Weighting          : term frequency (tf)
##
```

```
##                                                  Terms
## Docs                                              aaa aaeu aah aama aaminus
##   ReutersC50/C50train/AaronPressman/106247newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/120600newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/120683newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/136958newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/137498newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/14014newsML    0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/156814newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/182596newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/186392newsML   0    0   0    0       0
##   ReutersC50/C50train/AaronPressman/193495newsML   0    0   0    0       0
```

```
DTM = removeSparseTerms(DTM, 0.975)
DTM
```

```
## <<DocumentTermMatrix (documents: 5000, terms: 1386)>>
## Non-/sparse entries: 494509/6435491
## Sparsity           : 93%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

I am going to use Naive Bayes technique discussed in class to build the first model.

Naive Bayes:

Let us first convert the DTM to a data matrix.

```
X = as.matrix(DTM)
```

We can separate the training set and test set by just taking the first 2500 and the next 2500 entries in the DTM. This should work well simply because our file lists were correctly ordered based on train and test. We can see that here.

```
file_lists[2490:2510]
```

```
##  [1] "ReutersC50/C50train/WilliamKazer/257526newsML.txt"
##  [2] "ReutersC50/C50train/WilliamKazer/258689newsML.txt"
##  [3] "ReutersC50/C50train/WilliamKazer/264132newsML.txt"
##  [4] "ReutersC50/C50train/WilliamKazer/268647newsML.txt"
##  [5] "ReutersC50/C50train/WilliamKazer/278687newsML.txt"
##  [6] "ReutersC50/C50train/WilliamKazer/281216newsML.txt"
##  [7] "ReutersC50/C50train/WilliamKazer/28223newsML.txt"
##  [8] "ReutersC50/C50train/WilliamKazer/282935newsML.txt"
##  [9] "ReutersC50/C50train/WilliamKazer/287736newsML.txt"
## [10] "ReutersC50/C50train/WilliamKazer/289747newsML.txt"
## [11] "ReutersC50/C50train/WilliamKazer/304402newsML.txt"
## [12] "ReutersC50/C50test/AaronPressman/421829newsML.txt"
## [13] "ReutersC50/C50test/AaronPressman/424074newsML.txt"
## [14] "ReutersC50/C50test/AaronPressman/42764newsML.txt"
## [15] "ReutersC50/C50test/AaronPressman/43033newsML.txt"
## [16] "ReutersC50/C50test/AaronPressman/433558newsML.txt"
## [17] "ReutersC50/C50test/AaronPressman/436774newsML.txt"
## [18] "ReutersC50/C50test/AaronPressman/439561newsML.txt"
## [19] "ReutersC50/C50test/AaronPressman/450383newsML.txt"
## [20] "ReutersC50/C50test/AaronPressman/450788newsML.txt"
## [21] "ReutersC50/C50test/AaronPressman/466275newsML.txt"
```

So now let us go ahead and split the training and test data.

```
X_train <- X[1:2500,]
labels <- unique(labels)
```

Now let us calculate the term level weights for each author by applying the Laplace smoothing factor.

```
smooth_count = 1/nrow(X_train)
for(i in 1:50)
{
  w_name <- paste("w",labels[i], sep = "_")
  temp <- colSums(X_train[(50*(i-1)+1):(50*i),] + smooth_count)
  assign(w_name, temp/sum(temp))
}
```

Now using the above weight vectors, let us predict the author name of the test data. We do this by calculation the log probabilities of all documents across all authors. The author with the highest value will be the most probable author for that document.

```
X_test <- X[2501:5000,]

pred = matrix(, nrow = 2500, ncol = 51)
for(i in 1:2500)
{
  for(j in 1:50)
  {
    w_name <- paste("w",labels[j], sep = "_")
    pred[i,j] = sum(X_test[i,]*log(get(w_name)))
  }
}

pred[1:10,1:5]

##             [,1]       [,2]       [,3]       [,4]       [,5]
##  [1,]  -968.6253 -1205.9818 -1054.3074 -1213.7225 -1158.2985
##  [2,]  -591.0338  -741.1685  -681.2818  -670.1952  -762.9471
##  [3,] -2185.4947 -2835.3524 -2494.5284 -2845.0260 -2532.8456
##  [4,]  -604.7419  -902.3168  -800.7661  -775.4366  -781.8511
##  [5,] -1182.3740 -1663.7194 -1507.1808 -1531.8951 -1488.0430
##  [6,]  -942.7602 -1151.2090 -1032.4955 -1172.2059 -1101.3595
##  [7,] -1598.4550 -1676.2207 -1528.6123 -1707.6570 -1560.7165
##  [8,] -1130.7470 -1649.9562 -1462.2392 -1422.1669 -1461.4791
##  [9,]  -816.9542 -1241.0801 -1113.7767 -1071.7356 -1109.9958
## [10,]  -770.3144 -1018.6272  -847.8154 -1007.4177  -887.8821
```

Let us now create a list with the predicted authors for each document by finding the highest probable authors for each document.

```
for (i in 1:2500)
{
  pred[i,51] = which.max(pred[i,])
}

predicted = NULL
predicted = cbind((rep(1:50, each=50)),pred[,51])
predicted$actual_author <- rep(1:50, each=50)
predicted$pred_author <- pred[,51]
```

Let us now compare the predicted and actual authors using a confusion matrix

```
library(caret)
library(e1071)

confusionMatrix(predicted$pred_author,predicted$actual_author)
```

```
## 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 42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          2  0 25  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
##          3  0  0 20  0  2  0  0  0  3  0  0  0  0  0  0  0  3  5  0  2  0
##          4  0  0  0 11  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0
##          5  0  0  0  0 27  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          6  1  0  0  0  0 43  0  8  0  1  0  0  0  0  0  0  0  0  0  0  0
##          7  1  0  0  0  0  0 14  0  0  0  0  0  7  0  0  0  0  0  0  0  0
##          8  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  0
##          9  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  3  0  0  1  0
##         10  0  0  0  0  0  2  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0
##         11  0  0  0  0  0  0  0  0  0  0 49  0  0  0  0  0  0  0  0  0  0
##         12  0  0  0  2  0  0  0  0  0  0  0 39  0  0  2  0  0  0  0  0  0
##         13  0  0  0  0  3  0 36  0  0  0  0  0 17  0  0  0  0  0  0  0  0
##         14  0  8  0  0  0  0  0  0  0  0  0  0  1 26  0  0  0  0 11  0  0
##         15  0  0  0 13  0  0  0  1  0  0  0  3 12  0 18  0  0  0  0  0  0
##         16  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 50  0  0  0  0  0
##         17  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0 33  0  0  2  0
##         18  1  0 28  0  1  0  0  1  0  0  0  0  0  0  0  0  1 39  0  0  0
##         19  0 17  0  0  0  0  0  0  0  0  0  0  3 23  0  0  0  0 32  0  0
##         20  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0 37  0
##         21  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  2  0 47
##         22  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0
##         23  0  0  0  0  0  1  0  3  0  3  0  0  0  0  0  0  0  0  0  0  0
##         24  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         25  0  0  0  0  0  0  0  0  5  2  0  0  0  0  0  0  0  0  0  1  0
##         26  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##         27  0  0  0  0  0  0  0  0  0  0  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  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0
##         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  1  0  0  0  4  0  0  0  0  0  0  0  0
##         32  0  0  0  0  0  1  0  2  0  1  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  0  0  0  0  0  0  0  2
##         34  0  0  0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         35  0  0  0  3  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 10  0  0  0  0  0  0  0  0  0  0  0
##         37  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         38  0  0  0  3  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0
##         39  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  1  0  0
##         40  0  0  0  0  0  1  0  4  0  1  0  0  0  0  0  0  0  0  0  0  0
##         41  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0
##         42  3  0  1  0  0  1  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0
##         43  0  0  0  0  4  0  0  0  0  0  1  4  0  0  0  0  0  1  0  0  0
##         44  0  0  0  7  0  0  0  0  0  0  0  1  1  0 14  0  0  0  1  0  1
##         45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  2  0
##         46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         47  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
##         48  0  0  0  0  2  0  0  0  6  0  0  0  0  0  0  0  8  1  0  3  0
##         49  0  0  0  0  0  1  0 23  6  0  0  0  0  0  0  0  0  0  1  0  0
##         50  2  0  0 11  0  0  0  0  0  0  0  2  2  0  2  0  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  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  1
##         2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         3   0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##         4   0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         5   0  0  6  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0
##         6   0  6  0  0  0  5  0  0  0  0  3  0  0  0  1  4  0  0  2  0  0
##         7   0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         8   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         9   5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##         10  0  0  0  0 10  1  0  0  0  0  0  0  0  0  3  2  0  0  1  0  5
##         11  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         12  0  0  0  0  0  1  1  0  0  0  0  0  0  4  0  0  0  0  0  0  0
##         13  0  0  0  0  0  1  0  0  0  2  0  2  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  2  0  0  0  4  1  0  1  0  3  0  9  0  0  1  0  0  1  0
##         16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
##         17  1  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         18  0  0  0  1  0  0  0  0  1  1  0  0  4  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  1  0  0  2  0  0  0  0  5  0  0  0  0  0  0  1  0  1  2  0  0
##         21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         22 37  1  1  7  0  0  0  0  1  0  0  0  2  0  0  0  0  0  0  2  0
##         23  0 26  0  0  1  2  0  0  2  0  2  0  1  0  0  6  0  0  1  0  2
##         24  0  0 28  0  0  0  0  0  0  8  0  0  0  0  0  0  0  0  1  0  0
##         25  5  0  2 34  0  0  0  0  1  2  0  0  0  0  2  0  0  0  0  0  0
##         26  0  0  0  0 32  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1
##         27  0  0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         28  0  0  0  0  0  0 39  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         29  0  0  0  0  0  0  0 49  4  0  0  2  0  1  0  0  0  0  0  0  0
##         30  0  0  0  0  0  0  0  0 30  0  0  0  0  0  0  0  0 13  1  0  0
##         31  0  0  4  0  0  0  0  0  0 21  0  0  0  0  0  0  0  0  0  0  0
##         32  0  0  0  0  0  0  0  0  0  0 25  0  0  0  0  0  0  0  1  0  2
##         33  0  0  0  0  0  0  0  0  0  0  0 42  0  0  0  0  0  0  0  0  0
##         34  0  2  0  0  0  1  0  0  0  0  0  0 39  0  0  1  0  0  1  2  1
##         35  1  0  0  0  0  0  2  0  0  1  0  0  0 13  0  0  0  0  0  0  0
##         36  0  0  1  0  1  0  0  0  0  0  0  0  0  0 41  3  0  0  0  2  0
##         37  0 10  0  0  1  0  0  0  0  0  1  0  1  0  1 31  0  0  0  0  0
##         38  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0 33  0  0  0  0
##         39  0  0  0  0  0  0  0  0  3  2  0  0  0  1  0  0  0 34  0  0  0
##         40  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 40  0  0
##         41  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0 38  0
##         42  0  4  0  0  1  2  0  0  1  0  7  0  0  0  1  0  0  0  0  1 32
##         43  0  0  2  1  0  1  0  0  0  5  1  0  0  0  0  0  3  0  0  0  0
##         44  0  0  0  0  0  0  0  0  0  1  1  0  0  8  0  0  1  0  0  2  0
##         45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
##         46  0  0  0  0  0  0  1  0  0  0  0  1  0  1  0  0 11  0  0  0  0
##         47  0  1  0  0  4  0  0  0  2  0  7  0  1  0  0  1  0  0  0  0  6
##         48  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##         49  0  0  0  0  0  0  1  0  0  2  2  0  1  0  0  0  0  0  0  0  0
##         50  0  0  3  0  0  0  2  0  0  0  0  0  0  8  0  0  0  0  0  0  0
##           Reference
## Prediction 43 44 45 46 47 48 49 50
##          1  0  0  0  0  4  0  0  0
##          2  0  0  0  0  0  0  0  0
##          3  0  0 16  0  0  0  0  0
##          4  0  0  0  0  0  0  0  4
##          5  0  0  0  1  0  0  0  0
##          6  0  0  0  0  1  0  2  0
##          7  0  0  0  0  0  0  0  0
##          8  0  0  0  0  0  0 13  0
##          9  0  0  0  0  0  2  0  0
##         10  0  0  0  0  8  0  0  0
##         11  0  0  1  0  0  0  0  2
##         12  1  1  0  1  0  0  0  4
##         13  0  0  0  0  0  0  0  0
##         14  0  0  0  0  0  0  0  0
##         15  1 28  0  0  0  0  1  9
##         16  1  0  0  0  0  0  1  0
##         17  0  0  0  0  0  0  0  0
##         18  0  0  4  0  0  0  0  0
##         19  0  0  0  0  0  0  0  0
##         20  0  0  0  0  0  1  0  0
##         21  0  0  0  0  0  0  7  0
##         22  0  0  0  0  0  2  0  0
##         23  0  0  0  0  4  0  3  0
##         24  0  0  0  0  0  0  0  0
##         25  0  0  0  0  0  0  0  0
##         26  0  0  0  0  1  0  0  0
##         27  0  0  0  0  0  0  0  0
##         28  0  1  0  0  0  0  0  1
##         29  0  0  0  0  0  0  0  0
##         30  0  0  0  0  0  0  0  0
##         31  0  0  0  0  0  0  0  0
##         32  0  0  0  0  1  0  0  0
##         33  0  0  0  0  0  0  0  0
##         34  0  0  0  0  0  1  1  0
##         35  0  2  0  0  0  0  0  2
##         36  0  0  0  0  0  0  0  0
##         37  0  0  0  0  0  0  0  0
##         38 11  4  0 18  0  0  0  1
##         39  0  0  0  0  0  0  0  0
##         40  0  0  0  0  0  0  0  0
##         41  0  0  0  0  0  0  0  0
##         42  0  0  0  0  5  0  1  0
##         43 27  0  0  8  0  0  1  1
##         44  1 13  0  1  0  0  0  5
##         45  0  0 28  0  0  5  0  0
##         46  8  1  0 20  0  0  0  4
##         47  0  0  0  0 26  0  0  0
##         48  0  0  1  0  0 39  0  0
##         49  0  0  0  0  0  0 20  0
##         50  0  0  0  1  0  0  0 17
##
## Overall Statistics
##
##                Accuracy : 0.6024
##                  95% CI : (0.5829, 0.6217)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5943
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.8400   0.5000   0.4000   0.2200   0.5400   0.8600
## Specificity            0.9963   0.9996   0.9869   0.9939   0.9955   0.9861
## Pos Pred Value         0.8235   0.9615   0.3846   0.4231   0.7105   0.5584
## Neg Pred Value         0.9967   0.9899   0.9877   0.9842   0.9907   0.9971
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0168   0.0100   0.0080   0.0044   0.0108   0.0172
## Detection Prevalence   0.0204   0.0104   0.0208   0.0104   0.0152   0.0308
## Balanced Accuracy      0.9182   0.7498   0.6935   0.6069   0.7678   0.9231
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.2800   0.1400   0.4000    0.5000    0.9800
## Specificity            0.9947   0.9947   0.9951    0.9869    0.9988
## Pos Pred Value         0.5185   0.3500   0.6250    0.4386    0.9423
## Neg Pred Value         0.9854   0.9827   0.9878    0.9898    0.9996
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0056   0.0028   0.0080    0.0100    0.0196
## Detection Prevalence   0.0108   0.0080   0.0128    0.0228    0.0208
## Balanced Accuracy      0.6373   0.5673   0.6976    0.7435    0.9894
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.7800    0.3400    0.5200    0.3600    1.0000
## Specificity             0.9931    0.9820    0.9918    0.9633    0.9984
## Pos Pred Value          0.6964    0.2787    0.5652    0.1667    0.9259
## Neg Pred Value          0.9955    0.9865    0.9902    0.9866    1.0000
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0156    0.0068    0.0104    0.0072    0.0200
## Detection Prevalence    0.0224    0.0244    0.0184    0.0432    0.0216
## Balanced Accuracy       0.8865    0.6610    0.7559    0.6616    0.9992
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.6600    0.7800    0.6400    0.7400    0.9400
## Specificity             0.9967    0.9824    0.9824    0.9931    0.9959
## Pos Pred Value          0.8049    0.4756    0.4267    0.6852    0.8246
## Neg Pred Value          0.9931    0.9955    0.9926    0.9947    0.9988
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0132    0.0156    0.0128    0.0148    0.0188
## Detection Prevalence    0.0164    0.0328    0.0300    0.0216    0.0228
## Balanced Accuracy       0.8284    0.8812    0.8112    0.8665    0.9680
##                      Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity             0.7400    0.5200    0.5600    0.6800    0.6400
## Specificity             0.9927    0.9873    0.9943    0.9918    0.9984
## Pos Pred Value          0.6727    0.4561    0.6667    0.6296    0.8889
## Neg Pred Value          0.9947    0.9902    0.9910    0.9935    0.9927
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0148    0.0104    0.0112    0.0136    0.0128
## Detection Prevalence    0.0220    0.0228    0.0168    0.0216    0.0144
## Balanced Accuracy       0.8663    0.7537    0.7771    0.8359    0.8192
##                      Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity             0.6200    0.7800    0.9800    0.6000    0.4200
## Specificity             1.0000    0.9988    0.9959    0.9943    0.9951
## Pos Pred Value          1.0000    0.9286    0.8305    0.6818    0.6364
## Neg Pred Value          0.9923    0.9955    0.9996    0.9919    0.9882
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0156    0.0196    0.0120    0.0084
## Detection Prevalence    0.0124    0.0168    0.0236    0.0176    0.0132
## Balanced Accuracy       0.8100    0.8894    0.9880    0.7971    0.7076
##                      Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity             0.5000    0.8400    0.7800    0.2600    0.8200
## Specificity             0.9967    0.9992    0.9947    0.9939    0.9931
## Pos Pred Value          0.7576    0.9545    0.7500    0.4643    0.7069
## Neg Pred Value          0.9899    0.9967    0.9955    0.9850    0.9963
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0100    0.0168    0.0156    0.0052    0.0164
## Detection Prevalence    0.0132    0.0176    0.0208    0.0112    0.0232
## Balanced Accuracy       0.7484    0.9196    0.8873    0.6269    0.9065
##                       Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity             0.6200    0.6600    0.6800    0.8000    0.7600
## Specificity             0.9943    0.9829    0.9955    0.9976    0.9988
## Pos Pred Value          0.6889    0.4400    0.7556    0.8696    0.9268
## Neg Pred Value          0.9923    0.9930    0.9935    0.9959    0.9951
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0132    0.0136    0.0160    0.0152
## Detection Prevalence    0.0180    0.0300    0.0180    0.0184    0.0164
## Balanced Accuracy       0.8071    0.8214    0.8378    0.8988    0.8794
##                       Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity             0.6400    0.5400    0.2600    0.5600    0.4000
## Specificity             0.9873    0.9865    0.9816    0.9951    0.9890
## Pos Pred Value          0.5079    0.4500    0.2241    0.7000    0.4255
## Neg Pred Value          0.9926    0.9906    0.9848    0.9911    0.9878
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0128    0.0108    0.0052    0.0112    0.0080
## Detection Prevalence    0.0252    0.0240    0.0232    0.0160    0.0188
## Balanced Accuracy       0.8137    0.7633    0.6208    0.7776    0.6945
##                       Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity             0.5200    0.7800    0.4000    0.3400
## Specificity             0.9902    0.9902    0.9849    0.9865
## Pos Pred Value          0.5200    0.6190    0.3509    0.3400
## Neg Pred Value          0.9902    0.9955    0.9877    0.9865
## Prevalence              0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0104    0.0156    0.0080    0.0068
## Detection Prevalence    0.0200    0.0252    0.0228    0.0200
## Balanced Accuracy       0.7551    0.8851    0.6924    0.6633
```

The accuracy of this Naive Bayes classification model is 60.24%.

There are authors whose works are similar and hence it is tough to distinguish them - these authors have similar "bag of words" and hence it is safe to assume that they write on similar topics. In fact, we can see a lot of such examples in the confusion matrix. Take author 14 and 19. 23 articles of author JanLopatka (author number 14) have been classified as written by JohnMastrini (author number 19). 11 of John's article have again been classified as written by Jan. This means both should be writing about similar stuff. Lets go ahead and verify that!

```
head(sort(w_JanLopatka, decreasing=TRUE),30)
```

```
##       percent        czech         year      billion         bank
##   0.025054098  0.024900411  0.014603364  0.013988615  0.011836993
##        crowns     minister   government         told        state
##   0.009992746  0.009685371  0.009377997  0.008916935  0.008455873
##         party       social       prague      foreign        banks
##   0.007687437  0.007533750  0.007533750  0.007226375  0.007072688
##         lower        added      cabinet       vaclav        prime
##   0.007072688   0.007072688   0.006457939   0.006457939   0.006304252
##        prices         years         stake         crown       central
##   0.005996877   0.005996877   0.005843190   0.005689503   0.005382128
## privatisation         house         month          news         trade
##   0.004921066   0.004767379   0.004767379   0.004767379   0.004767379
```

```
head(sort(w_JohnMastrini, decreasing=TRUE),30)
```

```
##       czech     percent        year      prague  government        bank
## 0.035472420 0.023840706 0.012926999 0.011921789 0.010772978 0.009911369
##       crown     foreign       party     billion     deficit      market
## 0.009049761 0.009049761 0.008906159 0.007613747 0.007470145 0.007182942
##       house       trade     central      crowns       lower        told
## 0.007039341 0.006752138 0.006464935 0.006034131 0.006034131 0.006034131
##    minister   president       banks     million    ministry      vaclav
## 0.005890530 0.005746928 0.005603327 0.005603327 0.005603327 0.005172523
##      social        vote     capital   political      growth    analysts
## 0.005028921 0.005028921 0.004885320 0.004741719 0.004598117 0.004310914
```

We can easily see that both authors seem to be writers of articles on Czech Government. Even their last names make sense now!

The second model I have tried is a random Forest classifier.

Random Forest:

Let us first convert the data matrix to a dataframe.

```
actual_author = rep(rep(1:50,each=50),2)
author = as.data.frame(X)
colnames(author) = make.names(colnames(author))
author$actual_author=actual_author
author$actual_author=as.factor(author$actual_author)
```

Let us now split the data into training and test similarly and then run the random forest model.

```
author_train=author[1:2500,]
author_test=author[2501:5000,]

library(randomForest)

set.seed(23432)
rf_author=randomForest(actual_author~.,data=author_train)
predicted_author=predict(rf_author,newdata=author_test)
confusionMatrix(predicted_author,author_test$actual_author)

## 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  1  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          2  0 36  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  3  0  0
##          3  0  0 15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  1  0
##          4  0  0  0 23  0  0  0  0  0  0  0  0  0  0 17  0  0  0  0  0  0
##          5  0  0  0  0 24  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          6  1  0  0  0  0 27  0  5  0  1  0  0  0  0  0  0  0  0  0  0  0
##          7  0  0  0  0  0  0 14  0  0  0  0  0 27  0  0  0  0  0  0  0  0
##          8  0  0  0  0  0  0  0  6  2  0  0  0  0  0  0  0  0  0  0  0  0
##          9  0  0  3  0  0  0  0  0 18  0  0  0  0  0  0  0  2  4  0  4  0
##         10  0  0  0  0  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0  0
##         11  0  0  0  0  0  0  0  0  0  1 50  0  0  0  0  0  0  0  0  0  0
##         12  0  0  0  2  0  0  0  0  0  0  0 48  0  0  2  0  0  0  0  0  0
##         13  0  0  1  0  2  1 36  0  0  0  0  0 19  0  0  0  0  0  0  0  0
##         14  0  2  0  0  0  0  0  0  0  0  0  0  0 26  0  0  0  0 10  0  0
##         15  0  0  0  5  0  0  0  0  0  0  0  1  0  0 14  0  0  0  0  0  0
##         16  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 50  0  0  0  0  0
##         17  0  0  2  0  0  0  0  0  6  0  0  0  0  0  0  0 42  0  0  1  0
##         18  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  0  2 29  0  0  0
##         19  0 12  0  0  0  0  0  0  0  0  0  0  0 20  0  0  0  0 35  0  0
##         20  0  0  1  0  0  0  0  0  3  0  0  0  0  0  0  0  1  2  0 34  0
##         21  1  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0 46
##         22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  1  0
##         23  0  0  0  0  0 10  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0
##         24  0  0  0  0 14  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         25  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         26  1  0  0  0  0  0  0  0  0  6  0  0  0  0  0  0  0  0  0  0  0
##         27  0  0  0  0  0  0  0  0  0  0  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  2  1  0
##         30  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##         31  0  0  0  0 10  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##         32  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         33  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4
##         34  0  0  0  0  0  1  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0
##         35  0  0  0  4  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0
##         36  0  0  0  0  0  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0
##         37  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         38  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         39  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
##         40  0  0  0  0  0  1  0  0  4  0  0  0  0  0  0  0  0  1  0  0  0
##         41  0  0  0  0  0  0  0  0  1  5  0  0  0  0  0  0  0  0  0  0  0
##         42  1  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0
##         43  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  4  0  0  0
##         44  0  0  0  3  0  0  0  0  0  0  0  0  0  0  9  0  0  0  0  0  0
##         45  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  9  0  1  0
##         46  0  0  0  1  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0
##         47  0  0  1  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0
##         48  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  2  0  0  7  0
##         49  0  0  0  0  0  2  0 36  8  0  0  0  0  0  0  0  0  0  0  0  0
##         50  0  0  0  8  0  0  0  0  0  0  0  0  1  0  3  0  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  2  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  3  1
##           2    0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           3    0  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  1  0  0  0  0  0  0  2  0  0  0  0  0  0  0
##           5    0  0  9  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           6    0  4  0  0  0  0  0  0  0  0  3  0  0  0  0  1  0  0  0  0  0
##           7    1  1  0  0  0 17  0  0  0  0  0  0  0  0  0  0  0  0  0  0  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  1  0  0  0  1  0  0
##           10   0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##           11   0  1  1  2  0  0  1  0  0  0  2  0  0  1  0  0  0  0  0  0  0
##           12   0  0  0  0  0  0  8  0  0  1  0  0  0  3  0  0  1  0  0  0  0
##           13   0  0  0  0  0  0  0  0  0  3  0  1  0  1  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  2  0  0  0  0  0  0  0
##           16   0  0  0  1  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0  2  0
##           17   0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           18   0  0  1  0  0  0  0  0  0  0  0  0  0  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   4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           21   0  2  0  0  0  0  0  1  0  0  1  3  3  0  0  0  0  0  0  0  0
##           22  39  0  0 10  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           23   0 28  0  0  0  0  0  0  0  0  1  0  5  0  0  1  0  0  0  0  0
##           24   0  0 31  0  0  0  0  0  0  2  1  0  0  0  0  0  0  0  0  0  0
##           25   0  0  0 21  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           26   0  0  0  1 44  1  0  0  0  0  3  0  1  0  1  2  0  0  0  0 14
##           27   0  0  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           28   0  0  0  0  0  0 39  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           29   0  0  0  0  0  0  0 49  1  1  0  0  0  0  0  0  0  0  0  0  0
##           30   0  0  0  1  0  0  0  0 46  0  0  1  0  0  0  2  0 27  2  0  0
##           31   1  0  4  0  0  1  0  0  0 42  0  0  0  0  1  1  0  0  0  0  0
##           32   0  0  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  1  0  0  1
##           33   0  0  0  0  0  0  0  0  0  0  0 45  1  0  1  0  0  0  0  0  0
##           34   0  1  0  0  0  0  0  0  0  0  1  0 32  0  0  0  0  0  3  1  0
##           35   0  0  0  0  0  0  0  0  0  1  0  0  0 20  0  0  1  0  0  2  0
##           36   0  0  2  4  1  0  0  0  0  0  1  0  0  0 44  0  0  0  0  0  1
##           37   2  5  0  0  0  0  0  0  0  0  1  0  0  0  0 40  0  1  0  0  0
##           38   0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0 42  0  0  0  0
##           39   0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0 20  0  0  0
##           40   1  6  0  0  0  0  0  0  0  0  0  0  1  0  0  3  0  0 44  0  0
##           41   0  0  0  0  0  0  0  0  0  0  2  0  2  0  0  0  0  0  0 41  0
##           42   0  1  0  0  2  0  0  0  1  0 10  0  0  0  1  0  0  0  0  0 23
##           43   0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##           44   0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0
##           45   0  0  0  1  0  0  0  0  0  0  0  0  2  0  0  0  0  1  0  0  0
##           46   0  0  0  1  0  0  0  0  0  0  0  0  4  0  0  5  0  0  0  0  0
##           47   0  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  9
##           48   1  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##           49   0  0  0  0  0  0  1  0  0  0  1  0  2  0  1  0  0  0  0  0  0
##           50   0  0  0  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  1  0
##             Reference
## Prediction 43 44 45 46 47 48 49 50
##         1   0  0  1  0  3  0  1  0
##         2   0  0  0  0  0  0  0  0
##         3   0  0  0  0  0  0  0  0
##         4   1 18  0  0  0  0  0  9
##         5   0  0  0  0  0  0  0  0
##         6   0  0  0  0  0  0  4  0
##         7   0  0  0  0  0  0  0  0
##         8   0  0  0  0  0  0 11  0
##         9   0  0  0  0  0  7  0  0
##         10  0  0  0  0  4  0  0  0
##         11  0  0  0  0  0  0  0  1
##         12  0  0  0  0  0  0  0  4
##         13  0  0  0  0  0  0  0  0
##         14  0  0  0  0  0  0  0  0
##         15  0  2  0  0  0  0  0  4
##         16  0  0  0  1  0  0  2  1
##         17  0  0  1  0  0  0  0  0
##         18  0  0  6  0  0  0  0  0
##         19  0  0  0  0  0  0  0  0
##         20  0  0  0  0  0  1  0  0
##         21  0  1  0  0  0  0  6  0
##         22  0  0  0  0  0  1  0  0
##         23  0  0  0  0  1  0  0  0
##         24  0  0  0  0  0  0  0  0
##         25  0  0  0  0  0  0  0  0
##         26  0  0  0  0  7  0  0  0
##         27  0  0  0  0  0  0  0  0
##         28  0  2  0  0  0  0  0  1
##         29  0  0  0  0  0  0  0  0
##         30  0  0  0  0  0  1  0  0
##         31  0  0  0  0  0  1  0  0
##         32  0  0  0  0  2  0  0  0
##         33  0  0  0  0  0  0  0  0
##         34  0  0  0  0  0  1  1  0
##         35  0  8  0  0  0  0  0  5
##         36  0  0  0  0  4  0  0  0
##         37  0  0  0  0  4  0  0  0
##         38 10  3  0 14  0  0  0  2
##         39  0  0  0  0  0  0  0  0
##         40  0  0  0  0  0  0  0  0
##         41  0  0  0  0  0  0  1  0
##         42  0  0  0  0  0  0  0  0
##         43 24  0  0 10  0  0  0  0
##         44  0 11  0  0  0  0  0  3
##         45  0  0 41  0  0  3  0  0
##         46 15  2  0 25  0  0  0  4
##         47  0  0  0  0 25  0  0  0
##         48  0  0  1  0  0 35  0  0
##         49  0  0  0  0  0  0 24  0
##         50  0  3  0  0  0  0  0 16
##
## Overall Statistics
##
##                Accuracy : 0.624
##                  95% CI : (0.6047, 0.643)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6163
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.9200   0.7200   0.3000   0.4600   0.4800   0.5400
## Specificity            0.9939   0.9971   0.9992   0.9804   0.9963   0.9922
## Pos Pred Value         0.7541   0.8372   0.8824   0.3239   0.7273   0.5870
## Neg Pred Value         0.9984   0.9943   0.9859   0.9889   0.9895   0.9906
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0184   0.0144   0.0060   0.0092   0.0096   0.0108
## Detection Prevalence   0.0244   0.0172   0.0068   0.0284   0.0132   0.0184
## Balanced Accuracy      0.9569   0.8586   0.6496   0.7202   0.7382   0.7661
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.2800   0.1200   0.3600    0.3200    1.0000
## Specificity            0.9812   0.9947   0.9902    0.9971    0.9959
## Pos Pred Value         0.2333   0.3158   0.4286    0.6957    0.8333
## Neg Pred Value         0.9852   0.9823   0.9870    0.9863    1.0000
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0056   0.0024   0.0072    0.0064    0.0200
## Detection Prevalence   0.0240   0.0076   0.0168    0.0092    0.0240
## Balanced Accuracy      0.6306   0.5573   0.6751    0.6586    0.9980
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.9600    0.3800    0.5200    0.2800    1.0000
## Specificity             0.9914    0.9816    0.9951    0.9943    0.9959
## Pos Pred Value          0.6957    0.2969    0.6842    0.5000    0.8333
## Neg Pred Value          0.9992    0.9873    0.9903    0.9854    1.0000
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0192    0.0076    0.0104    0.0056    0.0200
## Detection Prevalence    0.0276    0.0256    0.0152    0.0112    0.0240
## Balanced Accuracy       0.9757    0.6808    0.7576    0.6371    0.9980
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.8400    0.5800    0.7000    0.6800    0.9200
## Specificity             0.9951    0.9857    0.9869    0.9951    0.9922
## Pos Pred Value          0.7778    0.4531    0.5224    0.7391    0.7077
## Neg Pred Value          0.9967    0.9914    0.9938    0.9935    0.9984
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0168    0.0116    0.0140    0.0136    0.0184
## Detection Prevalence    0.0216    0.0256    0.0268    0.0184    0.0260
## Balanced Accuracy       0.9176    0.7829    0.8435    0.8376    0.9561
##                      Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity             0.7800    0.5600    0.6200    0.4200    0.8800
## Specificity             0.9947    0.9922    0.9931    1.0000    0.9849
## Pos Pred Value          0.7500    0.5957    0.6458    1.0000    0.5432
## Neg Pred Value          0.9955    0.9910    0.9923    0.9883    0.9975
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0156    0.0112    0.0124    0.0084    0.0176
## Detection Prevalence    0.0208    0.0188    0.0192    0.0084    0.0324
## Balanced Accuracy       0.8873    0.7761    0.8065    0.7100    0.9324
##                      Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity             0.6200    0.7800    0.9800    0.9200    0.8400
## Specificity             1.0000    0.9988    0.9980    0.9857    0.9918
## Pos Pred Value          1.0000    0.9286    0.9074    0.5679    0.6774
## Neg Pred Value          0.9923    0.9955    0.9996    0.9983    0.9967
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0156    0.0196    0.0184    0.0168
## Detection Prevalence    0.0124    0.0168    0.0216    0.0324    0.0248
## Balanced Accuracy       0.8100    0.8894    0.9890    0.9529    0.9159
##                      Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity             0.4000    0.9000    0.6400    0.4000    0.8800
## Specificity             0.9984    0.9971    0.9955    0.9910    0.9902
## Pos Pred Value          0.8333    0.8654    0.7442    0.4762    0.6471
## Neg Pred Value          0.9879    0.9980    0.9927    0.9878    0.9975
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0080    0.0180    0.0128    0.0080    0.0176
## Detection Prevalence    0.0096    0.0208    0.0172    0.0168    0.0272
## Balanced Accuracy       0.6992    0.9486    0.8178    0.6955    0.9351
##                      Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity             0.8000    0.8400    0.4000    0.8800    0.8200
## Specificity             0.9918    0.9857    0.9988    0.9931    0.9955
## Pos Pred Value          0.6667    0.5455    0.8696    0.7213    0.7885
## Neg Pred Value          0.9959    0.9967    0.9879    0.9975    0.9963
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0160    0.0168    0.0080    0.0176    0.0164
## Detection Prevalence    0.0240    0.0308    0.0092    0.0244    0.0208
## Balanced Accuracy       0.8959    0.9129    0.6994    0.9365    0.9078
##                      Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity             0.4600    0.4800    0.2200    0.8200    0.5000
## Specificity             0.9914    0.9927    0.9922    0.9927    0.9853
## Pos Pred Value          0.5227    0.5714    0.3667    0.6949    0.4098
## Neg Pred Value          0.9890    0.9894    0.9842    0.9963    0.9897
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0092    0.0096    0.0044    0.0164    0.0100
## Detection Prevalence    0.0176    0.0168    0.0120    0.0236    0.0244
## Balanced Accuracy       0.7257    0.7363    0.6061    0.9063    0.7427
##                      Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity             0.5000    0.7000    0.4800    0.3200
## Specificity             0.9931    0.9922    0.9792    0.9894
## Pos Pred Value          0.5952    0.6481    0.3200    0.3810
## Neg Pred Value          0.9898    0.9939    0.9893    0.9862
## Prevalence              0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0100    0.0140    0.0096    0.0064
## Detection Prevalence    0.0168    0.0216    0.0300    0.0168
## Balanced Accuracy       0.7465    0.8461    0.7296    0.6547
```

Random Forest gives a slightly better accuracy of ~62.4%

Conclusion:

Using the training and test corpus of Reuters datasets, I built 2 classifiers - Naive Bayes and Random Forest. Out of these 2, Random Forest gives slightly better prediction than Naive Bayes. Even though the accuracy is slightly higher for Random Forest, I would still choose Naive Bayes because it is much less complex than Random Forest. It is also less computationally intensive compared to Random Forest.

Question 3: Association rule mining

Objective: To use the data on grocery purchases and find some interesting association rules for the shopping baskets.

```
# Loading arules and reshape packages
library(arules)
library(reshape)
```

Let us read the data from the text file.

```
groc <- read.csv("groceries.txt", header = FALSE)
head(groc,8)

##                   V1                 V2             V3
## 1     citrus fruit semi-finished bread      margarine
## 2   tropical fruit             yogurt         coffee
## 3       whole milk
## 4        pip fruit             yogurt   cream cheese
## 5 other vegetables         whole milk condensed milk
## 6       whole milk             butter         yogurt
## 7 abrasive cleaner
## 8       rolls/buns
##                         V4
## 1             ready soups
## 2
## 3
## 4            meat spreads
## 5 long life bakery product
## 6                     rice
## 7
## 8
```

We can see that each row has only 4 columns. But we do have rows with more than 4 columns, for example elements in row number 7 and 8 from above are actually from row 6. So we need to find out the actual number of columns first and use that while reading the csv.

```
#Finding the maximum number of columns
max(count.fields("groceries.txt", sep = ','))

## [1] 32

groc <- read.csv("groceries.txt", header = FALSE, col.names =
paste0("V",seq_len(32)), fill = TRUE)
dim(groc)

## [1] 9835   32

head(groc,8)

##                  V1                  V2             V3
## 1      citrus fruit semi-finished bread      margarine
## 2    tropical fruit              yogurt         coffee
## 3        whole milk
## 4         pip fruit              yogurt   cream cheese
## 5  other vegetables          whole milk condensed milk
## 6        whole milk              butter         yogurt
## 7        rolls/buns
## 8  other vegetables            UHT-milk     rolls/buns
##                          V4                  V5 V6 V7 V8 V9 V10 V11 V12 V13
## 1              ready soups
## 2
## 3
## 4             meat spreads
## 5 long life bakery product
## 6                     rice   abrasive cleaner
## 7
## 8             bottled beer liquor (appetizer)
##   V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
##   V32
## 1
## 2
## 3
## 4
## 5
## 6
## 7
## 8
```

Now let us extract the number of transactions and then add it to the dataframe.

```
rows <- 1:nrow(groc)
groc <- cbind(rows,groc)
head(groc)
```

```
##   rows                V1                  V2             V3
## 1    1      citrus fruit semi-finished bread      margarine
## 2    2    tropical fruit              yogurt         coffee
## 3    3        whole milk                                   
## 4    4         pip fruit              yogurt  cream cheese 
## 5    5 other vegetables          whole milk condensed milk
## 6    6        whole milk              butter         yogurt
##                         V4               V5 V6 V7 V8 V9 V10 V11 V12 V13
## 1             ready soups                                               
## 2                                                                       
## 3                                                                       
## 4            meat spreads                                               
## 5 long life bakery product                                              
## 6                          rice abrasive cleaner                        
##   V14 V15 V16 V17 V18 V19 V20 V21 V22 V23 V24 V25 V26 V27 V28 V29 V30 V31
## 1                                                                        
## 2                                                                        
## 3                                                                        
## 4                                                                        
## 5                                                                        
## 6                                                                        
##   V32
## 1    
## 2    
## 3    
## 4    
## 5    
## 6    
```

As we can see above, each column has different number of levels. So we need to melt this dataframe into individual rows, add NA's and then remove them - this will make the number of levels for each column same, then get back the dataframe by using split function.

```
# unstacking the dataframe
groc1 <- melt(groc,id=c("rows"))
#ordering it by transactions
groc1 <- groc1[order(groc1$rows),]
#adding NA's
groc1[groc1==""] <- NA
#removing NA's
groc1 <- na.omit(groc1)
```

```
groc1$rows <- factor(groc1$rows)   #this is not a continuous variable, it
represents transaction id or number
head(groc1)
```

```
##       rows variable               value
## 1        1       V1        citrus fruit
## 9836     1       V2 semi-finished bread
## 19671    1       V3           margarine
## 29506    1       V4         ready soups
## 2        2       V1      tropical fruit
## 9837     2       V2              yogurt
```

As can be seen above the words are in the unstacked format (similar to how playlists.csv was stored)

Now let us create a list of baskets (analogous to bag of words)

```
# First split data into a list of items for each transaction
groc1 <- split(x=groc1$value, f=groc1$rows)

## Remove duplicates ("de-dupe")
groc1 <- lapply(groc1, unique)
head(groc1)
```

```
## $`1`
## [1] citrus fruit        semi-finished bread margarine
## [4] ready soups
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
##
## $`2`
## [1] tropical fruit yogurt         coffee
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
##
## $`3`
## [1] whole milk
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
##
## $`4`
## [1] pip fruit     yogurt        cream cheese  meat spreads
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
##
## $`5`
## [1] other vegetables         whole milk
## [3] condensed milk           long life bakery product
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
##
## $`6`
## [1] whole milk       butter           yogurt           rice
## [5] abrasive cleaner
## 170 Levels: abrasive cleaner artif. sweetener baby cosmetics ... baby food
```

We can now see the levels are the same in all the transactions and hence we can apply the apriori function to this.

```
## Cast this variable as a special arules "transactions" class.
groc_trans <- as(groc1, "transactions")

# Now run the 'apriori' algorithm
# Look at rules with support > .01 & confidence >.5 & length (# items) <= 4
groc_rules <- apriori(groc_trans, parameter=list(support=.01, confidence=.5,
maxlen=4))

# Look at the output
inspect(groc_rules)

##    lhs                        rhs                   support confidence
lift
## 1  {curd,
##     yogurt}                => {whole milk}       0.01006609  0.5823529
2.279125
## 2  {butter,
##     other vegetables}      => {whole milk}       0.01148958  0.5736041
2.244885
## 3  {domestic eggs,
##     other vegetables}      => {whole milk}       0.01230300  0.5525114
2.162336
## 4  {whipped/sour cream,
##     yogurt}                => {whole milk}       0.01087951  0.5245098
2.052747
## 5  {other vegetables,
##     whipped/sour cream} => {whole milk}       0.01464159  0.5070423
1.984385
## 6  {other vegetables,
##     pip fruit}             => {whole milk}       0.01352313  0.5175097
2.025351
## 7  {citrus fruit,
##     root vegetables}       => {other vegetables} 0.01037112  0.5862069
3.029608
## 8  {root vegetables,
##     tropical fruit}        => {other vegetables} 0.01230300  0.5845411
3.020999
## 9  {root vegetables,
##     tropical fruit}        => {whole milk}       0.01199797  0.5700483
2.230969
## 10 {tropical fruit,
##     yogurt}                => {whole milk}       0.01514997  0.5173611
2.024770
## 11 {root vegetables,
##     yogurt}                => {other vegetables} 0.01291307  0.5000000
2.584078
## 12 {root vegetables,
##     yogurt}                => {whole milk}       0.01453991  0.5629921
```

```
2.203354
## 13 {rolls/buns,
##     root vegetables}    => {other vegetables} 0.01220132  0.5020921
2.594890
## 14 {rolls/buns,
##     root vegetables}    => {whole milk}       0.01270971  0.5230126
2.046888
## 15 {other vegetables,
##     yogurt}             => {whole milk}       0.02226741  0.5128806
2.007235
```

With the cutoffs similar to playlists, we get 15 itemsets as result. But we can see that most of the results above contain associations between vegetables and milk products.

This makes sense because we have used a support of 0.01 and we totally have 9835 rows. But we can see even the itemset (root vegetables, yogurt, whole milk) which one would imagine to be one of the most frequently purchased grocery items has a support of only ~0.015, which is about 145 occurences of the itemset in the entire dataset. So keeping this in mind, we should reduce the support so that we identify more frequent itemsets.

Now let us play around with the cutoff values and zero-in on one set that makes sense

```
# First changing the support cutoff and keeping confidence cutoff constant
groc_rules <- apriori(groc_trans, parameter=list(support=.005, confidence=.5,
maxlen=4))

inspect(groc_rules)
```

The above gave 120 results so not including the output.

```
groc_rules <- apriori(groc_trans, parameter=list(support=.002, confidence=.8,
maxlen=4))

inspect(groc_rules)

##   lhs                      rhs                    support confidence     lift
## 1 {herbs,
##    tropical fruit}   => {whole milk}       0.002338587  0.8214286 3.214783
## 2 {herbs,
##    rolls/buns}       => {whole milk}       0.002440264  0.8000000 3.130919
## 3 {curd,
##    hamburger meat}   => {whole milk}       0.002541942  0.8064516 3.156169
## 4 {grapes,
##    tropical fruit,
##    whole milk}       => {other vegetables} 0.002033554  0.8000000 4.134524
## 5 {curd,
##    domestic eggs,
##    other vegetables} => {whole milk}       0.002846975  0.8235294 3.223005
## 6 {butter,
##    other vegetables,
##    pork}             => {whole milk}       0.002236909  0.8461538 3.311549
```

Again, nothing new. All that we would expect. I am decreasing the support even further and increasing the maxlen to 5.

```
groc_rules <- apriori(groc_trans, parameter=list(support=.0015,
confidence=.9, maxlen=5))

inspect(groc_rules)

##   lhs                           rhs                     support confidence
lift
## 1 {liquor,
##    red/blush wine}           => {bottled beer}     0.001931876  0.9047619
11.235269
## 2 {flour,
##    root vegetables,
##    whipped/sour cream}       => {whole milk}       0.001728521  1.0000000
3.913649
## 3 {cream cheese ,
##    other vegetables,
##    sugar}                    => {whole milk}       0.001525165  0.9375000
3.669046
## 4 {butter,
##    pip fruit,
##    whipped/sour cream}       => {whole milk}       0.001830198  0.9000000
3.522284
## 5 {domestic eggs,
##    tropical fruit,
##    whipped/sour cream}       => {whole milk}       0.001830198  0.9000000
3.522284
## 6 {fruit/vegetable juice,
##    tropical fruit,
##    whipped/sour cream}       => {other vegetables} 0.001931876  0.9047619
4.675950
## 7 {root vegetables,
##    sausage,
##    tropical fruit,
##    yogurt}                   => {whole milk}       0.001525165  0.9375000
3.669046
```

Now this is interesting. We have all what we expect, but we also have something that has not showed up until now which at the same time is very intuitive. I am talking about the item set (Liquor, red/blush wine/bottled beer).

This has a support of ~0.002 which is ~20 occurences of the itemsets in total. So we would not want to decrease the support any further.

Conclusion:

The best set so far is the one with support=0.0015 and confidence=0.9 and maxlen=5. I have chosen this as the best set because all of these sets make very good intuitive sense to me. Since my support is less and my confidence is high, it may be read as *most customers who buy itemset X will also buy itemset Y 90% of the times, even though they might not actually buy these sets as frequently as few others*.

- Customers who are in to purchase alcohol will tend to buy different sets of alcohols. So liquor and red/blush wine being associated with bottled beer makes total sense amd its confidence tells the same story!
- All the other items have very good associations as well. Whole milk goes with cream cheese/butter/yogurt/sugar and root/other vegetables/tropical fruits/domestic eggs. These all look like a regular grocery shopping.

So placing these above itemsets next to each other in the grocery store will make shopping very convenient for the customers.

Aside: I have designed my cutoffs in such a way that there are only few number of interesting associations found. We can play around with these numbers to find more number of association rules. For example, we can make the maxlen to be 2 and then increase the support cutoff to say 0.05

```
groc_rules <- apriori(groc_trans, parameter=list(support=.05, confidence=.2,
maxlen=2))

inspect(groc_rules)

##   lhs                   rhs                  support confidence     lift
## 1 {}                 => {whole milk}       0.25551601  0.2555160 1.000000
## 2 {yogurt}           => {whole milk}       0.05602440  0.4016035 1.571735
## 3 {whole milk}       => {yogurt}           0.05602440  0.2192598 1.571735
## 4 {rolls/buns}       => {whole milk}       0.05663447  0.3079049 1.205032
## 5 {whole milk}       => {rolls/buns}       0.05663447  0.2216474 1.205032
## 6 {other vegetables} => {whole milk}       0.07483477  0.3867578 1.513634
## 7 {whole milk}       => {other vegetables} 0.07483477  0.2928770 1.513634
```

So what the above results say is that 25% of the customers are definitely going to buy milk. So the store needs to make sure "Whole Milk"" is very accessible. And the combinations below are also very common pairs occuring more than 5% of times. So keep them next to each other at the same time make them very accessible too since a lot of people buy them!
