STA 380 Homework 2

Henry Chang, Joseph Chin, Tiffany Sung, Jeffrey Fulkerson

August 17, 2018

Problem 1: Flighs at ABIA

```
library(RColorBrewer)
library(gplots)
#setup

# Read in Data
data_raw <-
read.csv(url("https://raw.githubusercontent.com/jgscott/STA380/master/data/AB
IA.csv"))
# Clean
# --Year is the same for every row
# --Only interested in flights originating from Austin, TX
drops <- c("Year")
data <- data_raw[ data_raw$Origin == "AUS", !(names(data_raw) %in% drops)]
rm(data_raw)
```

For the purpose of this project, We decide to narrow our analysis to flights departing from AUS only.

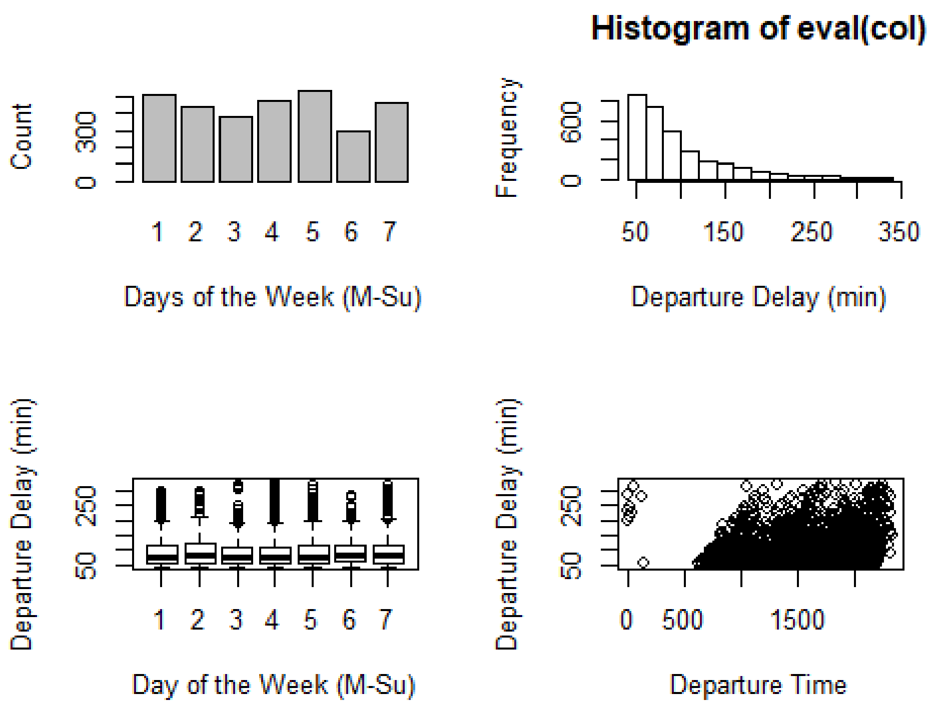
EDA

In the EDA process we are interested in how we can minimize delay.

We believe that a delay longer than 45 minutes is significant to a business traveler; thus, we decide to analyze flights meeting this standard in Austin-Bergstrom International Airport in 2008.

```
# Plotting functions
plot_delays <- function(dcol, col, colName){
  ## This function plots delay variables
  # --Model
  threshold = 45
  delays <- data[eval(dcol) > threshold,]
  # --Clean Delays data of high outliers and NA
  delays <- delays[eval(col) < unname(quantile(eval(col), 0.99, na.rm =
TRUE)),]
  delays <- delays[! is.na(eval(col)),]
  # --Display Results
  print(round(table(delays$DayOfWeek) / dim(delays)[1] * 100.0, 2))
  print(summary(eval(col)))
```

```
  par(mfrow=c(2,2))
  barplot(table(delays$DayOfWeek), xlab="Days of the Week (M-Su)",
ylab="Count")
  hist(eval(col), xlab = colName)
  boxplot(eval(col) ~ delays$DayOfWeek, xlab = "Day of the Week (M-Su)" ,
ylab = colName)
  plot(delays$DepTime, eval(col), xlab = "Departure Time" , ylab = colName)
}

##
##     1     2     3     4     5     6     7
## 16.34 14.36 12.39 15.18 17.35  9.50 14.88
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   46.00   58.00   78.00   96.13  116.00  328.00
```



Given the preliminary exploratory data analysis, we found out that Friday on average has the worst delay statistics, with most flights having longer than 45 minute delays. Separately, the graph on the bottom right showcases that departure time also affects the departure delay time of a flight.

Delay Heatmap

Given the summary above, we would like to create several delay heatmaps to help business travelers flying out of Austin minimize delay time by choosing the optimal time, day, and Airline combination.

We create heatmaps for the top 5 airlines with the most number of flights in AUS:

1. Southwest Airlines (WN)
2. American Airlines (AA)
3. Continental Airlines (CO)
4. Mesa Airlines (YV)
5. JetBlue Airways (B6)

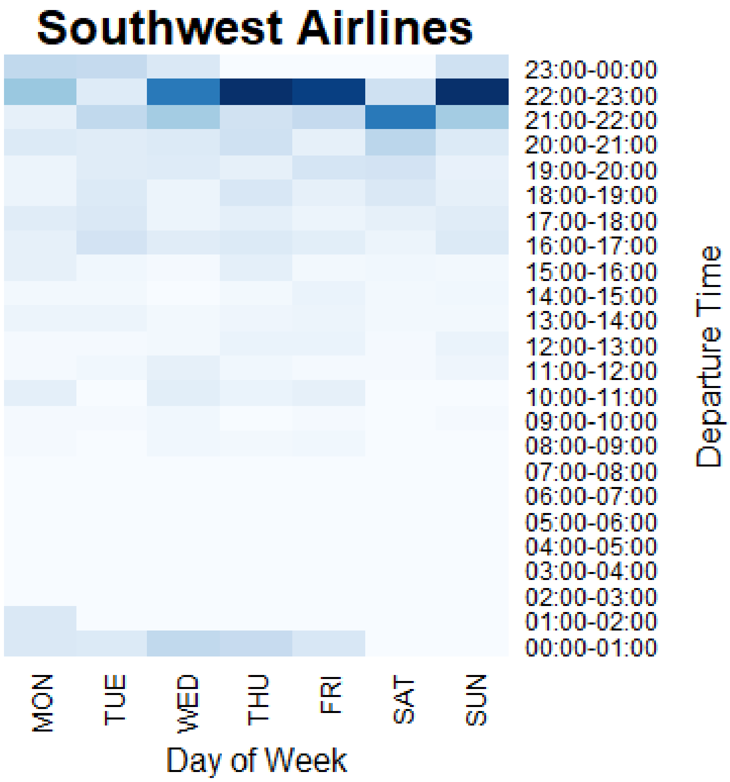
```
# CREATE A HEATMAP
# Create a Day of Week by Departure Time matrix of Average Delays (in minutes)
# for a particular airline
airlines <- c("WN", "AA", "CO", "YV", "B6")
carrierCodeLookup <- c("Southwest Airlines",
                       "American Airlines",
                       "Continental (UA)",
                       "Mesa Airlines",
                       "JetBlue")

airlineHeatMap <- function(cc, airlineName, timeIntervalInMinutes, thresholdInMinutes){

  # Setup
  timeInterval_converted <- timeIntervalInMinutes*(5.0/3.0) # Convertion from base 60 (time) to base 100 (0-2400)
  threshold <- thresholdInMinutes*(5.0/3.0) # How many minutes late are we counting, converted to base 100
  numRows <- 2400/(timeInterval_converted)
  values <- c()

  # Get Dataframe of just Carrier Code cc
  cc_data <- data[data$UniqueCarrier == cc, c("DayOfWeek", "DepTime", "DepDelay")] # TODO Limit this to just the necessary list of columns

  interval <- seq(from=0, to=2400, by=timeInterval_converted)
  l_interval <- length(interval)

  # For each day of the week...
  for (day in c(1,2,3,4,5,6,7)){
    cc_data_day <- cc_data[cc_data$DayOfWeek == day,]

    # For each time period...
    for (t in c(2:l_interval-1)){
      cc_data_day_t <- cc_data_day[interval[t] < cc_data_day$DepTime & cc_data_day$DepTime < interval[t+1],]

      # Get the Percent Chance of Delays * Avg. Duration of Delays in minutes
      totalFlights_t <- dim(cc_data_day_t)[1]
      delays <- cc_data_day_t[cc_data_day_t$DepDelay >= threshold,"DepDelay"]
      numDelays_t <- length(delays[!is.na(delays)])
      sumDelays_t <- sum(delays, na.rm=TRUE)
```

```
      #delayIntensity <-
(numDelays_t/totalFlights_t)*(sumDelays_t/numDelays_t)
      delayIntensity <- numDelays_t/totalFlights_t
      if (is.na(delayIntensity)){delayIntensity = 0}
      values <- c(values, delayIntensity) # TESTING: paste(day,
totalFlights_t, sep=':')
    }
  }

  # Create Matrix with Values inside
  return <- matrix(data=values, nrow=numRows, ncol=7)
}
```

Heatmaps

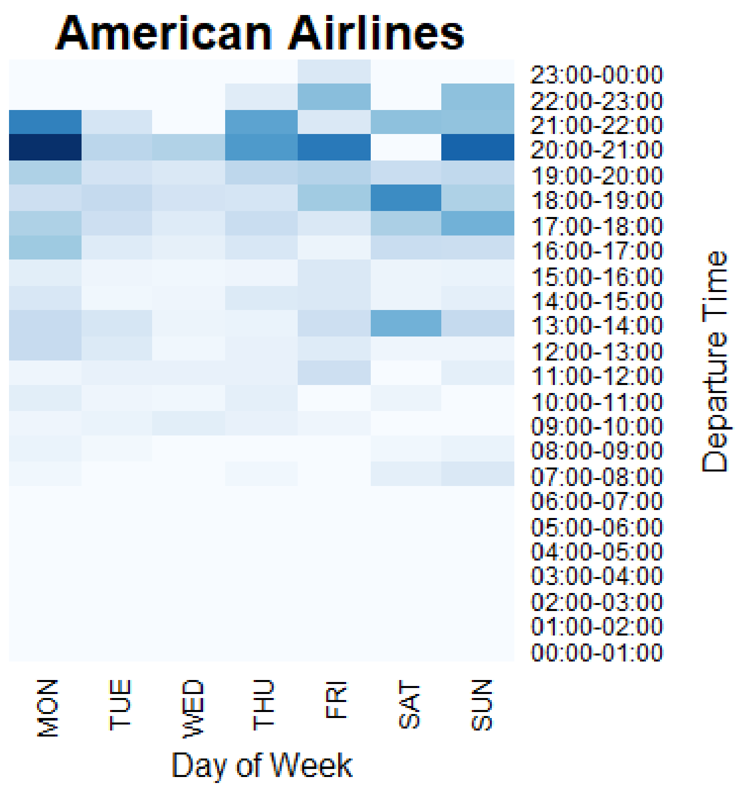
These heatmaps will show the possibility of a flight having a delay of more than 45 minutes, given the flight's departure time and day of week. The darker the color, the higher probability of delay.

```
#plot a heatmap for Southwest Airlines

heatmap(WN_m, Rowv=NA, Colv=NA,col= colorRampPalette(brewer.pal(9,
"Blues"))(100),xlab="Day of Week", ylab="Departure Time", main="Southwest
Airlines", scale = 'none', labCol = day, labRow = hour, margins = c(4,7),
cexRow=0.9,cexCol = 1)
```



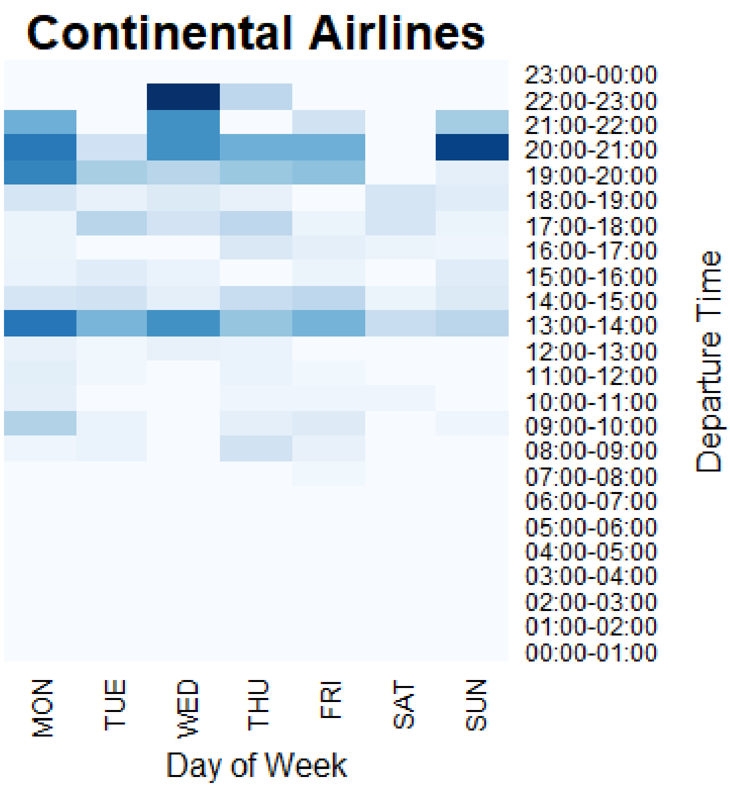
If you want to minimize delay, avoid Thursday, Friday, Sunday 22:00-23:00 flights from Southwest.

```
#plot a heatmap for American Airlines

heatmap(AA_m, Rowv=NA, Colv=NA,col= colorRampPalette(brewer.pal(9,
"Blues"))(100),xlab="Day of Week", ylab="Departure Time", main="American
Airlines", scale = 'none', labCol = day, labRow = hour, margins = c(4,7),
cexRow=0.9,cexCol = 1)
```



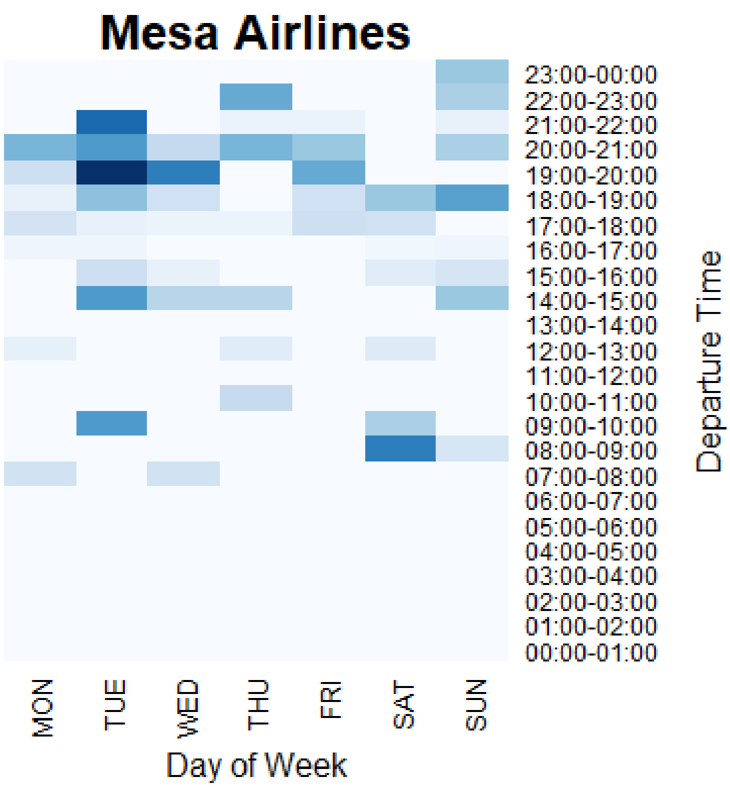
If you want to minimize delay, avoid Sunday and Monday 20:00 - 22:00 flights from American Airlines.

```
#plot a heatmap for Continental Airlines

heatmap(CO_m, Rowv=NA, Colv=NA,col= colorRampPalette(brewer.pal(9,
"Blues"))(100),xlab="Day of Week", ylab="Departure Time", main="Continental
Airlines", scale = 'none', labCol = day, labRow = hour, margins = c(4,7),
cexRow=0.9,cexCol = 1)
```



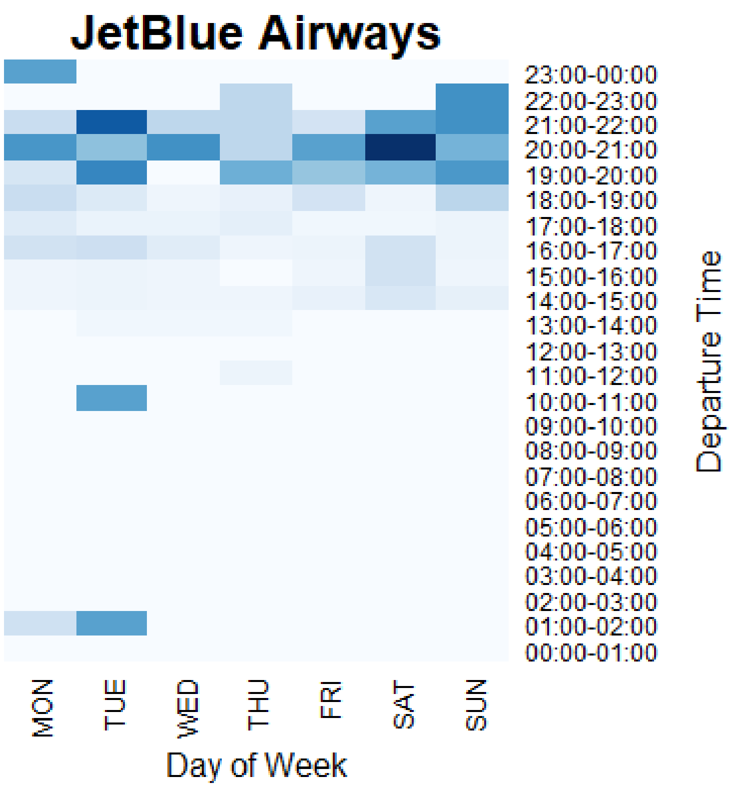
Try avoiding flights departing around 13:00-14:00 on Weekdays, as well as Monday, Wednesday, Sunday late night flights with Continental.

```
#plot a heatmap for Mesa Airlines

heatmap(YV_m, Rowv=NA, Colv=NA,col= colorRampPalette(brewer.pal(9,
"Blues"))(100),xlab="Day of Week", ylab="Departure Time", main="Mesa
Airlines", scale = 'none', labCol = day, labRow = hour, margins = c(4,7),
cexRow=0.9,cexCol = 1)
```



Mesa has several interesting delay blocks. In general, stay away from Mesa if you want to travel on a Tuesday night.

```
#plot a heatmap for Jetblue Airlines

heatmap(B6_m, Rowv=NA, Colv=NA,col= colorRampPalette(brewer.pal(9,
"Blues"))(100),xlab="Day of Week", ylab="Departure Time", main="JetBlue
Airways", scale = 'none', labCol = day, labRow = hour, margins = c(4,7),
cexRow=0.9,cexCol = 1)
```



If you want to minimize the possibility of delay, avoid Tuesday, Saturday, and Sunday 19:00-22:00 flights from Jetblue.

Conclusion

With these heatmaps, travelers will be able to minimize delay time by choosing the best day-time combination with one of the top five airlines.

Problem 2: Author attribution

Setup

By function “text_data_preprocess” we get a cleaned Corpus containing all of 2500 the document wiritten by each author. And the “get_y” function gives us a length 2500 list of authors whose order is the same as our Corpus object.

```
library(tm)
library(magrittr)
library(slam)
library(proxy)
library(glmnet)
library(caret)
library(dplyr)
library(naivebayes)
```

```r
library(randomForest)
library(e1071)
library(caret)

text_data_preprocess = function(pp){
  writer_list = list.files(pp)

  read_f_list = c()


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

  for ( i in writer_list){
    read_f_list = c(read_f_list,
                    Sys.glob(paste0(pp,i,'/*.txt')))
  }

  all_Doc = lapply(read_f_list, readerPlain)

  mynames = read_f_list %>%
  { strsplit(., '/', fixed=TRUE) } %>%
  { lapply(., tail, n=2) } %>%
  { lapply(., paste0, collapse = '') } %>%
    unlist


  names(all_Doc) = mynames



  documents_raw = Corpus(VectorSource(all_Doc))


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

```r
get_y = function(pp){
  writer_list = list.files(pp)
  y = c()
  for ( i in writer_list){
    y = c(y, rep(i,
                 times = length(list.files(paste0(pp,i)))))
  }
  return(y)
}
```

We build our "document term matrix" and prepare for PCA by removing sparse terms, sort columns by alphabetical order, and remove the zero-sum columns.

```r
my_documents = text_data_preprocess('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50train/')
y = get_y('C:/Users/Joseph/Desktop/jgscott git/data/ReutersC50/C50train/')
DTM_all = DocumentTermMatrix(my_documents)
DTM_all_d = removeSparseTerms(DTM_all, 0.95)
DTM_all_s = DTM_all_d[ ,order(DTM_all_d$dimnames$Terms)]

tfidf_all = weightTfIdf(DTM_all_d)
tfidf_matrix = as.matrix(tfidf_all)

scrub_cols = which(colSums(tfidf_matrix) == 0)
pre_pca_1 = tfidf_matrix[,-scrub_cols]
```

We then read the test data, ordered it, and take the intersection of words used in both test data and training data.

```r
test_documents = text_data_preprocess('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50test/')
y_test = get_y('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50test/')
DTM_test = DocumentTermMatrix(test_documents)
DTM_test_s = DTM_test[, order(DTM_test$dimnames$Terms)]
Term_inter = intersect(Terms(DTM_test_s), colnames(pre_pca_1))
```

We then run PCA using our training document term matrix, using the vocabulary intersection as columns. And we tried to fit a logistic regression using first 100 prinicle component.

```r
pre_pca_2 = pre_pca_1[,Term_inter]
pc_doc = prcomp(pre_pca_2, scale = TRUE)
X = (pc_doc$x)[, 1:100]
logit_model =
cv.glmnet(X,as.factor(y),family='multinomial',type.measure="class")
```

The last step is to predict y_test_hat using the lasso regression with the lambda having minimum error. We found that the accuracy of PCA + logistic regression is 55%

```
DTM_test_c = DTM_test_s[,Term_inter]
tfidf_test = weightTfIdf(DTM_test_c)
tfidf_test_matrix = as.matrix(tfidf_test)
tfidf_test_matrix_s = scale(tfidf_test_matrix)
pc_test = (tfidf_test_matrix_s %*% pc_doc$rotation)[, 1:100]

y_test_hat = predict(logit_model,   pc_test,
                     type = "class", s = logit_model$lambda.min)
pred2   = ifelse( y_test_hat == y_test,  1, 0)
mean(pred2)

## [1] 0.5524
```

DarrenSchuettler,DavidLawder,EdnaFernandes,BenjaminKangLim,JaneMacartney,William Kazer are the authors that PCA + logistic regression can't predict well.

```
author_p_result_df = data.frame(y_test, pred2)
author_p_result= author_p_result_df %>%
    group_by(y_test) %>%
    summarise(p_accu = mean(X1))
author_p_result_s = author_p_result[order(author_p_result$p_accu),]
author_p_result_s[c(1:10),]

## # A tibble: 10 x 2
##    y_test           p_accu
##    <fct>             <dbl>
##  1 DarrenSchuettler   0.18
##  2 EdnaFernandes      0.2
##  3 DavidLawder        0.22
##  4 JaneMacartney      0.24
##  5 BenjaminKangLim    0.26
##  6 WilliamKazer       0.28
##  7 MartinWolk         0.3
##  8 HeatherScoffield   0.32
##  9 MureDickie         0.34
## 10 JanLopatka         0.36
```

Naive Bayes

***Train/Test Set*

```
mycorpus = text_data_preprocess('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50train/')
labels = get_y('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50train/')
DTM = DocumentTermMatrix(mycorpus)
DTM = removeSparseTerms(DTM, 0.975)
tfidf_train = weightTfIdf(DTM)

X = as.matrix(tfidf_train)
```

```
mycorpus2 = text_data_preprocess('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50test/')
labels2 = get_y('C:/Users/Joseph/Desktop/jgscott
git/data/ReutersC50/C50test/')

DTM2=DocumentTermMatrix(mycorpus2)
DTM2=removeSparseTerms(DTM2,0.975)
tfidf_test = weightTfIdf(DTM2)

x2=as.matrix(tfidf_test)


words=colnames(X)
words2=colnames(x2)

W=words[!(words %in% words2)]
W2=words2[!(words2 %in% words)]

words_matrix=matrix(0,nrow=nrow(x2), ncol=length(W))
colnames(words_matrix)=W

words_matrix2=matrix(0,nrow=nrow(X), ncol=length(W2))
colnames(words_matrix2)=W2

train_matrix=cbind(X,words_matrix2)
test_matrix=cbind(x2,words_matrix)
```

***Predict Test Accuracy*

```
set.seed(1)
test_matrix=as.data.frame(test_matrix)
train_matrix=as.data.frame(train_matrix)

nb = naive_bayes(x=train_matrix,y=as.factor(labels),laplace=1)
predNB=predict(nb,test_matrix)

actual = rep(1:50,each=50)

TestTable = table(predNB,actual)
correct = 0
for (i in seq(1,50)){
    correct = correct + TestTable[i,i]
}

NB_accuracy = correct/2500
print(NB_accuracy)

## [1] 0.4412
```

The Naive Bayes model prediction accuracy is somewhat low, despite being much better than randomly guessing. A different model may have better predictive accuracy 0.4416

***Confusion Matrix of Naive Bayes* First, creat a confusion matrix to calculate the accuracy of the model in predicting the authors. Sensitivity column gives the accuracy % of predicting the documents under each of the authors correctly. Also, the accuracy of the model is the average of the accuracy measures for all the authors.

```
NB_confusion = confusionMatrix(table(predNB,labels))
NB_class= as.data.frame(NB_confusion$byClass)
NB_class[order(-NB_class$Sensitivity),][1]
```

```
##                          Sensitivity
## Class: LynnleyBrowning          0.84
## Class: MatthewBunce             0.74
## Class: RobinSidel               0.74
## Class: GrahamEarnshaw           0.68
## Class: BradDorfman              0.66
## Class: FumikoFujisaki           0.66
## Class: LynneO'Donnell           0.64
## Class: SarahDavison             0.64
## Class: NickLouth                0.62
## Class: LydiaZajc                0.60
## Class: SimonCowell              0.58
## Class: PeterHumphrey            0.56
## Class: JimGilchrist             0.54
## Class: KirstinRidley            0.54
## Class: AaronPressman            0.52
## Class: JoeOrtiz                 0.52
## Class: EricAuchard              0.50
## Class: TimFarrand               0.50
## Class: AlexanderSmith           0.48
## Class: JoWinterbottom           0.48
## Class: KeithWeir                0.48
## Class: JonathanBirt             0.46
## Class: PierreTran               0.46
## Class: MarcelMichelson          0.44
## Class: RogerFillion             0.44
## Class: BernardHickey            0.42
## Class: TheresePoletti           0.42
## Class: WilliamKazer             0.42
## Class: KarlPenhaul              0.40
## Class: SamuelPerry              0.40
## Class: HeatherScoffield         0.38
## Class: KevinDrawbaugh           0.38
## Class: TanEeLyn                 0.38
## Class: AlanCrosby               0.36
## Class: KevinMorrison            0.36
## Class: JohnMastrini             0.34
## Class: KouroshKarimkhany        0.34
```

```
## Class: MarkBendeich             0.34
## Class: MichaelConnor            0.32
## Class: MureDickie               0.32
## Class: JaneMacartney            0.30
## Class: MartinWolk               0.28
## Class: ScottHillis              0.28
## Class: ToddNissen               0.28
## Class: PatriciaCommins          0.22
## Class: JanLopatka               0.20
## Class: DavidLawder              0.16
## Class: EdnaFernandes            0.16
## Class: BenjaminKangLim          0.14
## Class: DarrenSchuettler         0.14
```

The model predict well for a few authors like LynnleyBrowning, MatthewBunce and RobinSidel. ###Random Forests### ***Predict Test Accuracy*

```
set.seed(1)
RF = randomForest(y=as.factor(labels), x=train_matrix,ntrees=500)
pr = predict(RF, test_matrix, type = "response")

TestTable2 = table(pr, actual)

correct2 = 0
for (i in seq(1,50)){
    correct2 = correct2 + TestTable2[i,i]
}

RF_accuracy = correct2/2500
print(RF_accuracy)

## [1] 0.6188
```

The random forest model was a good bit better at 0.6176

Confusion Matrix of Random Forest

```
RF_confusion = confusionMatrix(table(pr,labels))
RF_class= as.data.frame(RF_confusion$byClass)
RF_class[order(-RF_class$Sensitivity),][1]
##                           Sensitivity
## Class: FumikoFujisaki            1.00
## Class: JimGilchrist              0.98
## Class: LynnleyBrowning           0.98
## Class: GrahamEarnshaw            0.96
## Class: AaronPressman             0.94
## Class: KarlPenhaul               0.92
## Class: MatthewBunce              0.92
## Class: PeterHumphrey             0.90
## Class: JoWinterbottom            0.86
## Class: KouroshKarimkhany         0.86
```

```
## Class: NickLouth                0.86
## Class: RobinSidel               0.86
## Class: SimonCowell              0.84
## Class: MarcelMichelson          0.82
## Class: JohnMastrini             0.76
## Class: KeithWeir                0.76
## Class: LynneO'Donnell           0.76
## Class: MarkBendeich             0.76
## Class: MichaelConnor            0.76
## Class: RogerFillion             0.76
## Class: PatriciaCommins          0.74
## Class: TimFarrand               0.72
## Class: ToddNissen               0.72
## Class: BradDorfman              0.68
## Class: JonathanBirt             0.68
## Class: KevinDrawbaugh           0.64
## Class: SarahDavison             0.64
## Class: LydiaZajc                0.62
## Class: KevinMorrison            0.58
## Class: BernardHickey            0.52
## Class: JanLopatka               0.52
## Class: MureDickie               0.52
## Class: PierreTran               0.50
## Class: TheresePoletti           0.50
## Class: JoeOrtiz                 0.42
## Class: KirstinRidley            0.42
## Class: AlanCrosby               0.40
## Class: AlexanderSmith           0.40
## Class: HeatherScoffield         0.38
## Class: EricAuchard              0.36
## Class: JaneMacartney            0.34
## Class: SamuelPerry              0.34
## Class: TanEeLyn                 0.32
## Class: DarrenSchuettler         0.30
## Class: MartinWolk               0.30
## Class: WilliamKazer             0.30
## Class: BenjaminKangLim          0.28
## Class: EdnaFernandes            0.26
## Class: DavidLawder              0.14
## Class: ScottHillis              0.14
AccuracyRF = mean(RF_class$Sensitivity)
```

The model predict well for a few authors like FumikoFujisaki, JimGilchrist and LynnleyBrowning. And we can also see that random forest model, on average, have a better accuracy then the other model we did.

Problem 3: Association Rule Mining

Set up

```
library(tidyverse)
library(arules)
library(arulesViz)
```

We first read in our grocery list by letting each row of the data as a basket of one shopping list. And we seperate each row by comma as items in each basket.

```
groceries = read.transactions('C:/Users/Joseph/Desktop/jgscott
git/data/groceries.txt',
                              format = 'basket',
                              sep = ',',
                              rm.duplicates = FALSE)
grocery<- as(groceries, "transactions")
```

The apriori algorithm is used to identify the associations between the different products from the different baskets that were loaded. We first choose a stricter criteria of support value = 0.005 and a confidence = 0.5 and we could observe that the results are mostly 'whole milk' and 'other vegetables'.

Based on this result, we found that 'whole milk' and 'other vegetables' consists a significant portion of purchases from this grocery store. Therefore, we suggest that 'whole milk' and 'other vegatables' can be placed in the center of our store, which not only improves our costumor's shopping experience by getting what they needed quickly but also increases exposure of other products.

```
groc_rules <- apriori(grocery, parameter=list(support=.005, confidence=.5,
maxlen=6))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.5    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target   ext
##       6  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.01s].
```

```
## writing ... [120 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(groc_rules)
```

```
##       lhs                           rhs                   support
confidence     lift count
## [1]   {baking powder}            => {whole milk}       0.009252669
0.5229885 2.046793    91
## [2]   {oil,
##        other vegetables}         => {whole milk}       0.005083884
0.5102041 1.996760    50
## [3]   {onions,
##        root vegetables}          => {other vegetables} 0.005693950
0.6021505 3.112008    56
## [4]   {onions,
##        whole milk}               => {other vegetables} 0.006609049
0.5462185 2.822942    65
## [5]   {hygiene articles,
##        other vegetables}         => {whole milk}       0.005185562
0.5425532 2.123363    51
## [6]   {other vegetables,
##        sugar}                    => {whole milk}       0.006304016
0.5849057 2.289115    62
## [7]   {long life bakery product,
##        other vegetables}         => {whole milk}       0.005693950
0.5333333 2.087279    56
## [8]   {cream cheese,
##        yogurt}                   => {whole milk}       0.006609049
0.5327869 2.085141    65
## [9]   {chicken,
##        root vegetables}          => {other vegetables} 0.005693950
0.5233645 2.704829    56
## [10]  {chicken,
##        root vegetables}          => {whole milk}       0.005998983
0.5514019 2.157993    59
## [11]  {chicken,
##        rolls/buns}               => {whole milk}       0.005287239
0.5473684 2.142208    52
## [12]  {coffee,
##        yogurt}                   => {whole milk}       0.005083884
0.5208333 2.038359    50
## [13]  {frozen vegetables,
##        root vegetables}          => {other vegetables} 0.006100661
0.5263158 2.720082    60
## [14]  {frozen vegetables,
##        root vegetables}          => {whole milk}       0.006202339
0.5350877 2.094146    61
## [15]  {frozen vegetables,
##        rolls/buns}               => {whole milk}       0.005083884
```

```
0.5000000 1.956825    50
## [16]  {frozen vegetables,
##        other vegetables}         => {whole milk}       0.009659380
0.5428571 2.124552    95
## [17]  {beef,
##        yogurt}                   => {whole milk}       0.006100661
0.5217391 2.041904    60
## [18]  {beef,
##        rolls/buns}               => {whole milk}       0.006812405
0.5000000 1.956825    67
## [19]  {curd,
##        whipped/sour cream}       => {whole milk}       0.005897306
0.5631068 2.203802    58
## [20]  {curd,
##        tropical fruit}           => {yogurt}           0.005287239
0.5148515 3.690645    52
## [21]  {curd,
##        tropical fruit}           => {other vegetables} 0.005287239
0.5148515 2.660833    52
## [22]  {curd,
##        tropical fruit}           => {whole milk}       0.006507372
0.6336634 2.479936    64
## [23]  {curd,
##        root vegetables}          => {other vegetables} 0.005490595
0.5046729 2.608228    54
## [24]  {curd,
##        root vegetables}          => {whole milk}       0.006202339
0.5700935 2.231146    61
## [25]  {curd,
##        yogurt}                   => {whole milk}       0.010066090
0.5823529 2.279125    99
## [26]  {curd,
##        rolls/buns}               => {whole milk}       0.005897306
0.5858586 2.292845    58
## [27]  {curd,
##        other vegetables}         => {whole milk}       0.009862735
0.5739645 2.246296    97
## [28]  {pork,
##        root vegetables}          => {other vegetables} 0.007015760
0.5149254 2.661214    69
## [29]  {pork,
##        root vegetables}          => {whole milk}       0.006812405
0.5000000 1.956825    67
## [30]  {pork,
##        rolls/buns}               => {whole milk}       0.006202339
0.5495495 2.150744    61
## [31]  {frankfurter,
##        tropical fruit}           => {whole milk}       0.005185562
0.5483871 2.146195    51
## [32]  {frankfurter,
```

```
##        root vegetables}          => {whole milk}       0.005083884
0.5000000 1.956825    50
## [33]  {frankfurter,
##        yogurt}                   => {whole milk}       0.006202339
0.5545455 2.170296    61
## [34]  {bottled beer,
##        yogurt}                   => {whole milk}       0.005185562
0.5604396 2.193364    51
## [35]  {brown bread,
##        tropical fruit}           => {whole milk}       0.005693950
0.5333333 2.087279    56
## [36]  {brown bread,
##        root vegetables}          => {whole milk}       0.005693950
0.5600000 2.191643    56
## [37]  {brown bread,
##        other vegetables}         => {whole milk}       0.009354347
0.5000000 1.956825    92
## [38]  {domestic eggs,
##        margarine}                => {whole milk}       0.005185562
0.6219512 2.434099    51
## [39]  {margarine,
##        root vegetables}          => {other vegetables} 0.005897306
0.5321101 2.750028    58
## [40]  {margarine,
##        rolls/buns}               => {whole milk}       0.007930859
0.5379310 2.105273    78
## [41]  {butter,
##        domestic eggs}            => {whole milk}       0.005998983
0.6210526 2.430582    59
## [42]  {butter,
##        whipped/sour cream}       => {other vegetables} 0.005795628
0.5700000 2.945849    57
## [43]  {butter,
##        whipped/sour cream}       => {whole milk}       0.006710727
0.6600000 2.583008    66
## [44]  {butter,
##        citrus fruit}             => {whole milk}       0.005083884
0.5555556 2.174249    50
## [45]  {bottled water,
##        butter}                   => {whole milk}       0.005388917
0.6022727 2.357084    53
## [46]  {butter,
##        tropical fruit}           => {other vegetables} 0.005490595
0.5510204 2.847759    54
## [47]  {butter,
##        tropical fruit}           => {whole milk}       0.006202339
0.6224490 2.436047    61
## [48]  {butter,
##        root vegetables}          => {other vegetables} 0.006609049
0.5118110 2.645119    65
```

```
## [49]  {butter,
##        root vegetables}           => {whole milk}       0.008235892
0.6377953 2.496107    81
## [50]  {butter,
##        yogurt}                    => {whole milk}       0.009354347
0.6388889 2.500387    92
## [51]  {butter,
##        other vegetables}          => {whole milk}       0.011489578
0.5736041 2.244885   113
## [52]  {newspapers,
##        root vegetables}           => {other vegetables} 0.005998983
0.5221239 2.698417    59
## [53]  {newspapers,
##        root vegetables}           => {whole milk}       0.005795628
0.5044248 1.974142    57
## [54]  {domestic eggs,
##        whipped/sour cream}        => {other vegetables} 0.005083884
0.5102041 2.636814    50
## [55]  {domestic eggs,
##        whipped/sour cream}        => {whole milk}       0.005693950
0.5714286 2.236371    56
## [56]  {domestic eggs,
##        pip fruit}                 => {whole milk}       0.005388917
0.6235294 2.440275    53
## [57]  {citrus fruit,
##        domestic eggs}             => {whole milk}       0.005693950
0.5490196 2.148670    56
## [58]  {domestic eggs,
##        tropical fruit}            => {whole milk}       0.006914082
0.6071429 2.376144    68
## [59]  {domestic eggs,
##        root vegetables}           => {other vegetables} 0.007320793
0.5106383 2.639058    72
## [60]  {domestic eggs,
##        root vegetables}           => {whole milk}       0.008540925
0.5957447 2.331536    84
## [61]  {domestic eggs,
##        yogurt}                    => {whole milk}       0.007727504
0.5390071 2.109485    76
## [62]  {domestic eggs,
##        other vegetables}          => {whole milk}       0.012302999
0.5525114 2.162336   121
## [63]  {fruit/vegetable juice,
##        root vegetables}           => {other vegetables} 0.006609049
0.5508475 2.846865    65
## [64]  {fruit/vegetable juice,
##        root vegetables}           => {whole milk}       0.006507372
0.5423729 2.122657    64
## [65]  {fruit/vegetable juice,
##        yogurt}                    => {whole milk}       0.009456024
```

```
0.5054348 1.978094    93
## [66]  {pip fruit,
##        whipped/sour cream}       => {other vegetables} 0.005592272
0.6043956 3.123610    55
## [67]  {pip fruit,
##        whipped/sour cream}       => {whole milk}       0.005998983
0.6483516 2.537421    59
## [68]  {citrus fruit,
##        whipped/sour cream}       => {other vegetables} 0.005693950
0.5233645 2.704829    56
## [69]  {citrus fruit,
##        whipped/sour cream}       => {whole milk}       0.006304016
0.5794393 2.267722    62
## [70]  {sausage,
##        whipped/sour cream}       => {whole milk}       0.005083884
0.5617978 2.198679    50
## [71]  {tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.007829181
0.5661765 2.926088    77
## [72]  {tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.007930859
0.5735294 2.244593    78
## [73]  {root vegetables,
##        whipped/sour cream}       => {other vegetables} 0.008540925
0.5000000 2.584078    84
## [74]  {root vegetables,
##        whipped/sour cream}       => {whole milk}       0.009456024
0.5535714 2.166484    93
## [75]  {whipped/sour cream,
##        yogurt}                   => {whole milk}       0.010879512
0.5245098 2.052747   107
## [76]  {rolls/buns,
##        whipped/sour cream}       => {whole milk}       0.007829181
0.5347222 2.092715    77
## [77]  {other vegetables,
##        whipped/sour cream}       => {whole milk}       0.014641586
0.5070423 1.984385   144
## [78]  {pip fruit,
##        sausage}                  => {whole milk}       0.005592272
0.5188679 2.030667    55
## [79]  {pip fruit,
##        root vegetables}          => {other vegetables} 0.008134215
0.5228758 2.702304    80
## [80]  {pip fruit,
##        root vegetables}          => {whole milk}       0.008947636
0.5751634 2.250988    88
## [81]  {pip fruit,
##        yogurt}                   => {whole milk}       0.009557702
0.5310734 2.078435    94
## [82]  {other vegetables,
```

```
##        pip fruit}                => {whole milk}       0.013523132
0.5175097 2.025351   133
## [83]  {pastry,
##        tropical fruit}           => {whole milk}       0.006710727
0.5076923 1.986930    66
## [84]  {pastry,
##        root vegetables}          => {other vegetables} 0.005897306
0.5370370 2.775491    58
## [85]  {pastry,
##        root vegetables}          => {whole milk}       0.005693950
0.5185185 2.029299    56
## [86]  {pastry,
##        yogurt}                   => {whole milk}       0.009150991
0.5172414 2.024301    90
## [87]  {citrus fruit,
##        root vegetables}          => {other vegetables} 0.010371124
0.5862069 3.029608   102
## [88]  {citrus fruit,
##        root vegetables}          => {whole milk}       0.009150991
0.5172414 2.024301    90
## [89]  {root vegetables,
##        shopping bags}            => {other vegetables} 0.006609049
0.5158730 2.666112    65
## [90]  {sausage,
##        tropical fruit}           => {whole milk}       0.007219115
0.5182482 2.028241    71
## [91]  {root vegetables,
##        sausage}                  => {whole milk}       0.007727504
0.5170068 2.023383    76
## [92]  {root vegetables,
##        tropical fruit}           => {other vegetables} 0.012302999
0.5845411 3.020999   121
## [93]  {root vegetables,
##        tropical fruit}           => {whole milk}       0.011997966
0.5700483 2.230969   118
## [94]  {tropical fruit,
##        yogurt}                   => {whole milk}       0.015149975
0.5173611 2.024770   149
## [95]  {root vegetables,
##        yogurt}                   => {other vegetables} 0.012913066
0.5000000 2.584078   127
## [96]  {root vegetables,
##        yogurt}                   => {whole milk}       0.014539908
0.5629921 2.203354   143
## [97]  {rolls/buns,
##        root vegetables}          => {other vegetables} 0.012201322
0.5020921 2.594890   120
## [98]  {rolls/buns,
##        root vegetables}          => {whole milk}       0.012709710
0.5230126 2.046888   125
```

```
## [99]  {other vegetables,
##        yogurt}                     => {whole milk}       0.022267412
0.5128806 2.007235   219
## [100] {fruit/vegetable juice,
##        other vegetables,
##        yogurt}                     => {whole milk}       0.005083884
0.6172840 2.415833    50
## [101] {fruit/vegetable juice,
##        whole milk,
##        yogurt}                     => {other vegetables} 0.005083884
0.5376344 2.778578    50
## [102] {other vegetables,
##        root vegetables,
##        whipped/sour cream}         => {whole milk}       0.005185562
0.6071429 2.376144    51
## [103] {root vegetables,
##        whipped/sour cream,
##        whole milk}                 => {other vegetables} 0.005185562
0.5483871 2.834150    51
## [104] {other vegetables,
##        whipped/sour cream,
##        yogurt}                     => {whole milk}       0.005592272
0.5500000 2.152507    55
## [105] {whipped/sour cream,
##        whole milk,
##        yogurt}                     => {other vegetables} 0.005592272
0.5140187 2.656529    55
## [106] {other vegetables,
##        pip fruit,
##        root vegetables}            => {whole milk}       0.005490595
0.6750000 2.641713    54
## [107] {pip fruit,
##        root vegetables,
##        whole milk}                 => {other vegetables} 0.005490595
0.6136364 3.171368    54
## [108] {other vegetables,
##        pip fruit,
##        yogurt}                     => {whole milk}       0.005083884
0.6250000 2.446031    50
## [109] {pip fruit,
##        whole milk,
##        yogurt}                     => {other vegetables} 0.005083884
0.5319149 2.749019    50
## [110] {citrus fruit,
##        other vegetables,
##        root vegetables}            => {whole milk}       0.005795628
0.5588235 2.187039    57
## [111] {citrus fruit,
##        root vegetables,
##        whole milk}                 => {other vegetables} 0.005795628
```

```
0.6333333 3.273165    57
## [112] {root vegetables,
##        tropical fruit,
##        yogurt}                     => {whole milk}       0.005693950
0.7000000 2.739554    56
## [113] {other vegetables,
##        root vegetables,
##        tropical fruit}             => {whole milk}       0.007015760
0.5702479 2.231750    69
## [114] {root vegetables,
##        tropical fruit,
##        whole milk}                 => {other vegetables} 0.007015760
0.5847458 3.022057    69
## [115] {other vegetables,
##        tropical fruit,
##        yogurt}                     => {whole milk}       0.007625826
0.6198347 2.425816    75
## [116] {tropical fruit,
##        whole milk,
##        yogurt}                     => {other vegetables} 0.007625826
0.5033557 2.601421    75
## [117] {other vegetables,
##        root vegetables,
##        yogurt}                     => {whole milk}       0.007829181
0.6062992 2.372842    77
## [118] {root vegetables,
##        whole milk,
##        yogurt}                     => {other vegetables} 0.007829181
0.5384615 2.782853    77
## [119] {other vegetables,
##        rolls/buns,
##        root vegetables}            => {whole milk}       0.006202339
0.5083333 1.989438    61
## [120] {other vegetables,
##        rolls/buns,
##        yogurt}                     => {whole milk}       0.005998983
0.5221239 2.043410    59
```

Set threshold for lift and confidence

In order to find other interesting associations on products that are puchased less, we loosen our filtering criteria to support = 0.002 and confidence = 0.4 and we sorted the association rules found by lift.

As we expected, items purchased less frequently are shown after adjusting our support filter. In addition, association rules found among them tend to have higher lift for their lower support.

```
groc_rules_1 <- apriori(grocery, parameter=list(support=.002, confidence=.4,
maxlen=6))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5   0.002      1
##  maxlen target   ext
##       6  rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 19
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [147 item(s)] done [0.00s].
## creating transaction tree ... done [0.01s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [1914 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(sort(subset(groc_rules_1, count >= 10, rhs), by = "lift")[c(1:50)])
```

```
##      lhs                          rhs                    support
confidence     lift count
## [1]  {hard cheese,
##       whipped/sour cream}    => {butter}             0.002033554
0.4545455 8.202669    20
## [2]  {butter,
##       hard cheese}           => {whipped/sour cream} 0.002033554
0.5128205 7.154028    20
## [3]  {butter,
##       other vegetables,
##       tropical fruit}        => {whipped/sour cream} 0.002338587
0.4259259 5.941818    23
## [4]  {frozen vegetables,
##       other vegetables,
##       yogurt}                => {whipped/sour cream} 0.002236909
0.4230769 5.902073    22
## [5]  {beef,
##       citrus fruit,
##       other vegetables}      => {root vegetables}    0.002135231
0.6363636 5.838280    21
## [6]  {citrus fruit,
##       other vegetables,
##       tropical fruit,
##       whole milk}            => {root vegetables}    0.003152008
0.6326531 5.804238    31
## [7]  {citrus fruit,
```

```
##       frozen vegetables,
##       other vegetables}      => {root vegetables}    0.002033554
0.6250000 5.734025    20
## [8]  {beef,
##       other vegetables,
##       tropical fruit}        => {root vegetables}    0.002745297
0.6136364 5.629770    27
## [9]  {bottled water,
##       root vegetables,
##       yogurt}                => {tropical fruit}     0.002236909
0.5789474 5.517391    22
## [10] {herbs,
##       other vegetables,
##       whole milk}            => {root vegetables}    0.002440264
0.6000000 5.504664    24
## [11] {other vegetables,
##       root vegetables,
##       tropical fruit,
##       whole milk}            => {citrus fruit}       0.003152008
0.4492754 5.428284    31
## [12] {grapes,
##       pip fruit}             => {tropical fruit}     0.002135231
0.5675676 5.408941    21
## [13] {herbs,
##       yogurt}                => {root vegetables}    0.002033554
0.5714286 5.242537    20
## [14] {beef,
##       other vegetables,
##       soda}                  => {root vegetables}    0.002033554
0.5714286 5.242537    20
## [15] {liquor}                => {bottled beer}       0.004677173
0.4220183 5.240594    46
## [16] {citrus fruit,
##       other vegetables,
##       root vegetables,
##       whole milk}            => {tropical fruit}     0.003152008
0.5438596 5.183004    31
## [17] {other vegetables,
##       rice}                  => {root vegetables}    0.002236909
0.5641026 5.175325    22
## [18] {beef,
##       citrus fruit,
##       whole milk}            => {root vegetables}    0.002236909
0.5641026 5.175325    22
## [19] {butter,
##       other vegetables,
##       whole milk,
##       yogurt}                => {tropical fruit}     0.002338587
0.5348837 5.097463    23
## [20] {beef,
```

```
##       butter,
##       whole milk}            => {root vegetables}    0.002033554
0.5555556 5.096911    20
## [21] {beef,
##       tropical fruit,
##       whole milk}            => {root vegetables}    0.002541942
0.5555556 5.096911    25
## [22] {grapes,
##       other vegetables,
##       whole milk}            => {tropical fruit}     0.002033554
0.5263158 5.015810    20
## [23] {butter,
##       other vegetables,
##       tropical fruit,
##       whole milk}            => {yogurt}             0.002338587
0.6969697 4.996135    23
## [24] {herbs,
##       whole milk}            => {root vegetables}    0.004168785
0.5394737 4.949369    41
## [25] {other vegetables,
##       sliced cheese,
##       whole milk}            => {root vegetables}    0.002440264
0.5333333 4.893035    24
## [26] {bottled water,
##       other vegetables,
##       whole milk,
##       yogurt}                => {tropical fruit}     0.002033554
0.5128205 4.887199    20
## [27] {citrus fruit,
##       other vegetables,
##       whole milk,
##       yogurt}                => {tropical fruit}     0.002440264
0.5106383 4.866403    24
## [28] {beef,
##       sausage}               => {root vegetables}    0.002948653
0.5272727 4.837432    29
## [29] {rice,
##       whole milk}            => {root vegetables}    0.002440264
0.5217391 4.786665    24
## [30] {oil,
##       other vegetables,
##       whole milk}            => {root vegetables}    0.002643620
0.5200000 4.770709    26
## [31] {citrus fruit,
##       other vegetables,
##       soda}                  => {root vegetables}    0.002135231
0.5121951 4.699104    21
## [32] {other vegetables,
##       pip fruit,
##       tropical fruit,
```

```
##       whole milk}            => {root vegetables}    0.002440264
0.5106383 4.684821    24
## [33] {other vegetables,
##       pip fruit,
##       whipped/sour cream}    => {tropical fruit}     0.002745297
0.4909091 4.678383    27
## [34] {beef,
##       butter}                => {root vegetables}    0.002948653
0.5087719 4.667698    29
## [35] {citrus fruit,
##       fruit/vegetable juice,
##       other vegetables}      => {tropical fruit}     0.002338587
0.4893617 4.663636    23
## [36] {citrus fruit,
##       other vegetables,
##       pip fruit}             => {tropical fruit}     0.002846975
0.4827586 4.600708    28
## [37] {herbs,
##       other vegetables}      => {root vegetables}    0.003863752
0.5000000 4.587220    38
## [38] {butter,
##       onions}                => {root vegetables}    0.002033554
0.5000000 4.587220    20
## [39] {other vegetables,
##       rolls/buns,
##       tropical fruit,
##       whole milk}            => {root vegetables}    0.002033554
0.5000000 4.587220    20
## [40] {citrus fruit,
##       root vegetables,
##       tropical fruit,
##       whole milk}            => {other vegetables}   0.003152008
0.8857143 4.577509    31
## [41] {rolls/buns,
##       root vegetables,
##       whole milk,
##       yogurt}                => {tropical fruit}     0.002236909
0.4782609 4.557845    22
## [42] {butter,
##       other vegetables,
##       yogurt}                => {tropical fruit}     0.003050330
0.4761905 4.538114    30
## [43] {citrus fruit,
##       other vegetables,
##       tropical fruit}        => {root vegetables}    0.004473818
0.4943820 4.535678    44
## [44] {beef,
##       tropical fruit}        => {root vegetables}    0.003762074
0.4933333 4.526057    37
## [45] {onions,
```

```
##       other vegetables,
##       whole milk}            => {root vegetables}    0.003253686
0.4923077 4.516648    32
## [46] {beef,
##       other vegetables,
##       rolls/buns}            => {root vegetables}    0.002846975
0.4912281 4.506743    28
## [47] {citrus fruit,
##       fruit/vegetable juice,
##       other vegetables}      => {root vegetables}    0.002338587
0.4893617 4.489620    23
## [48] {citrus fruit,
##       other vegetables,
##       whole milk,
##       yogurt}                => {root vegetables}    0.002338587
0.4893617 4.489620    23
## [49] {hard cheese,
##       other vegetables,
##       whole milk}            => {root vegetables}    0.002135231
0.4883721 4.480541    21
## [50] {other vegetables,
##       rolls/buns,
##       tropical fruit,
##       whole milk}            => {yogurt}             0.002541942
0.6250000 4.480230    25
```

Intuitively, we could also tell that items under same category are frequently bought together, for example, hard cheese => whipped/sour cream, grapes,pip fruit => citrus fruits, and liquor => bottled beer. Furthermore, we also found interesting associations among different categories, for instance, people bought beef are more likely to buy root vegetables.

If we let support = 0.01 and confidence = 0.1 and observe the condition when lift > 3, we could see a clearer pattern of beef being bought along with root vegetables. This finding might help the marketing strategy of root vegetables since seller might not think of the fact that root vegetables are actually purchase a lot for side dishes when people want to have a steak or make beef stew.

```
groc_rules_2 <- apriori(grocery, parameter=list(support=.01, confidence=.1,
maxlen=6))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target   ext
##       6  rules FALSE
##
## Algorithmic control:
```

```
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE   FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.02s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [435 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(subset(groc_rules_2, subset = lift > 3))
```

```
##     lhs                    rhs                      support confidence
lift count
## [1] {beef}              => {root vegetables}    0.01738688  0.3313953
3.040367   171
## [2] {root vegetables}  => {beef}               0.01738688  0.1595149
3.040367   171
## [3] {whole milk,
##      yogurt}            => {curd}               0.01006609  0.1796733
3.372304    99
## [4] {other vegetables,
##      yogurt}            => {whipped/sour cream} 0.01016777  0.2341920
3.267062   100
## [5] {citrus fruit,
##      root vegetables}  => {other vegetables}   0.01037112  0.5862069
3.029608   102
## [6] {citrus fruit,
##      other vegetables} => {root vegetables}    0.01037112  0.3591549
3.295045   102
## [7] {root vegetables,
##      tropical fruit}    => {other vegetables}   0.01230300  0.5845411
3.020999   121
## [8] {other vegetables,
##      tropical fruit}    => {root vegetables}    0.01230300  0.3427762
3.144780   121
```