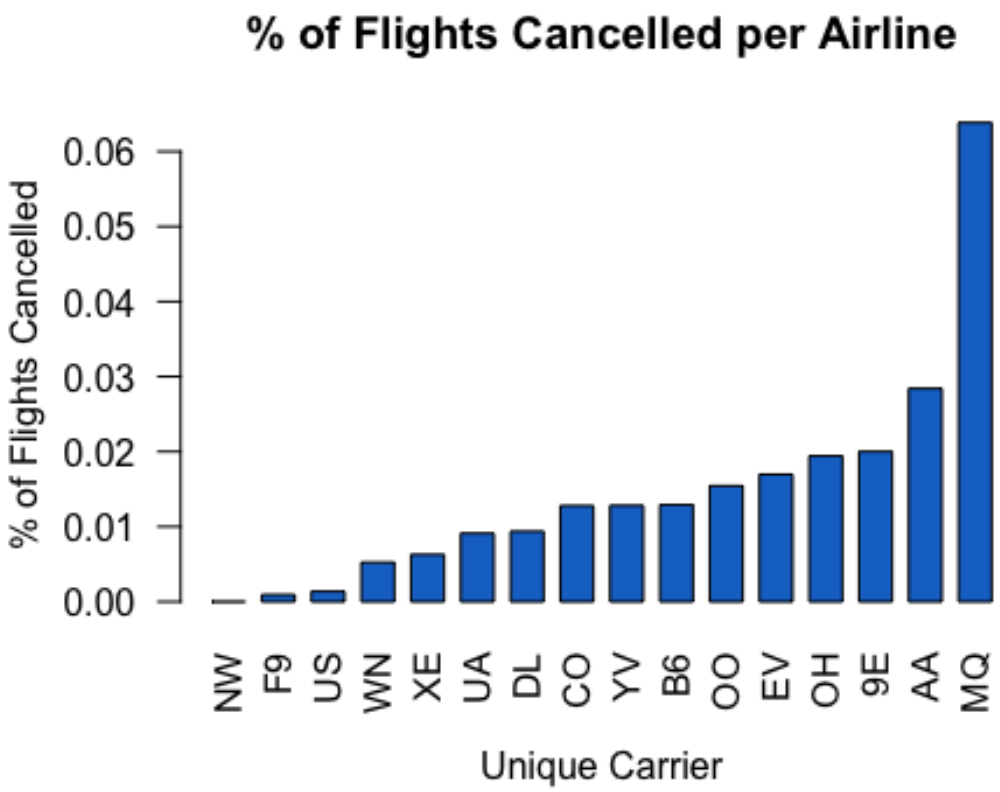
STA S380 HW2

Hope Knopf, Marie Gleichauf, Chelsea Matthews

8/17/2018

Flights at ABIA

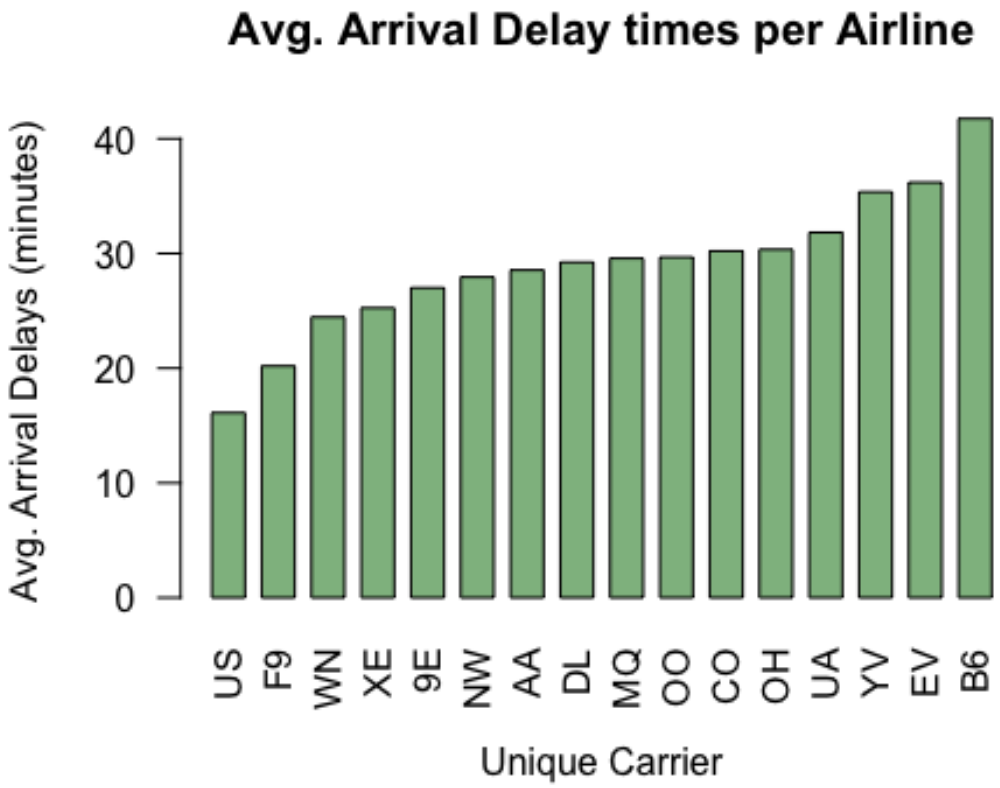
```
library(tm)
#setwd('/users/chelseamatthews/Documents')
airlinedata = read.csv('~/Documents/UT/Summer Classes/Intro to Predictive
Modeling/Part 2/HW 2/ABIA.csv', header=TRUE)

#percentage of flights cancelled for each airline
dfcancelled =
data.frame(aggregate(airlinedata$Cancelled~airlinedata$UniqueCarrier,airlined
ata,sum))

df=data.frame(aggregate(airlinedata$FlightNum ~
airlinedata$UniqueCarrier,airlinedata, length))
finaldf = merge(dfcancelled,df)
finaldf = within(finaldf, percent <-
airlinedata.Cancelled/airlinedata.FlightNum)
finaldf = finaldf[order(finaldf$percent),]

barplot(finaldf$percent, names = finaldf$arrivaldelays.UniqueCarrier,
        xlab = 'Unique Carrier', ylab = '% of Flights Cancelled',
        main = "% of Flights Cancelled per Airline",las=2, space = .5, col =
'dodgerblue3',
        names.arg = c("NW","F9","US","WN","XE","UA","DL","CO","YV","B6",
"OO","EV","OH","9E","AA","MQ"))
```



American Eagle (MQ) had the highest percentage of cancelled flights. American Airlines (AA) had the next highest percent of cancelled flights.

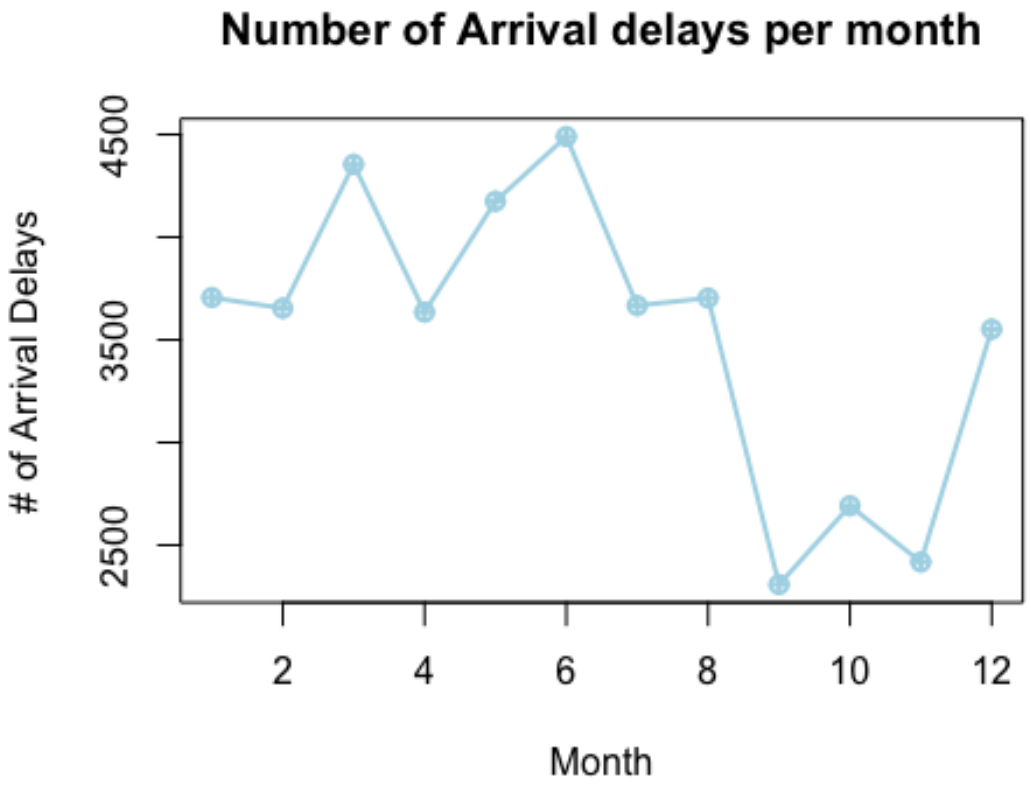
Arrivals

```
#average arrival delay

arrdelay = airlinedata[which(airlinedata[,15]>0),]
df_arrdelay = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$UniqueCarrier,arrdelay,mean))
df_arrdelay = df_arrdelay[order(df_arrdelay$arrdelay.ArrDelay),]

barplot(df_arrdelay$arrdelay.ArrDelay, names =
df_arrdelay$arrdelay.UniqueCarrier,
        xlab = "Unique Carrier", ylab = "Avg. Arrival Delays (minutes)",
        main = "Avg. Arrival Delay times per Airline", las = 2, space=.5, col
= 'darkseagreen')
```



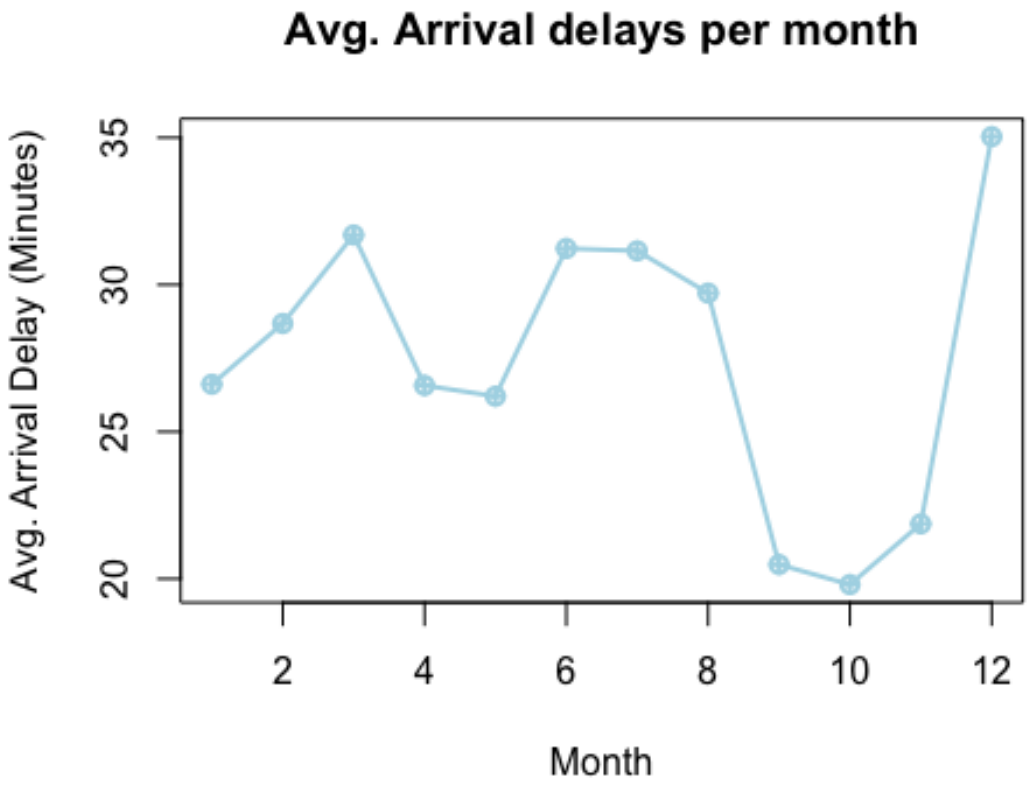
JetBlue(B6) has the longest average arrival delays, followed by ExpressJet (EV) and Mesa Airlines(YV).

```
#arrival delays per month
df_arrdelaymon = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$Month,arrdelay,length))
plot(df_arrdelaymon$arrdelay.Month, df_arrdelaymon$arrdelay.ArrDelay,
     xlab = "Month", ylab = "# of Arrival Delays",
     main = "Number of Arrival delays per month", type ='o',
col='lightblue',lwd=2, pch=10)
```



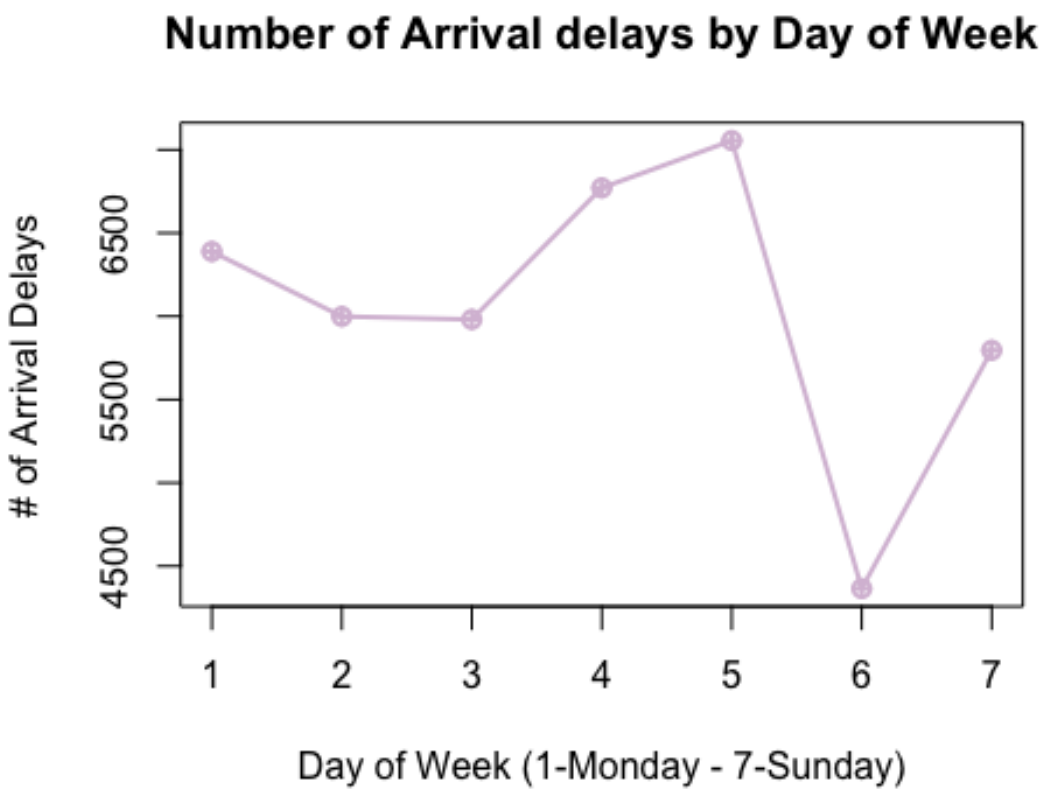
March and June had the highest number of arrival delays, while September and November had the smallest number of arrival delays.

```
#average arrival delays per month
df_arrdelaymon = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$Month,arrdelay,mean))
plot(df_arrdelaymon$arrdelay.Month, df_arrdelaymon$arrdelay.ArrDelay,
     xlab = "Month", ylab = "Avg. Arrival Delay (Minutes)",
     main = "Avg. Arrival delays per month", type ='o',
col='lightblue',lwd=2, pch=10)
```



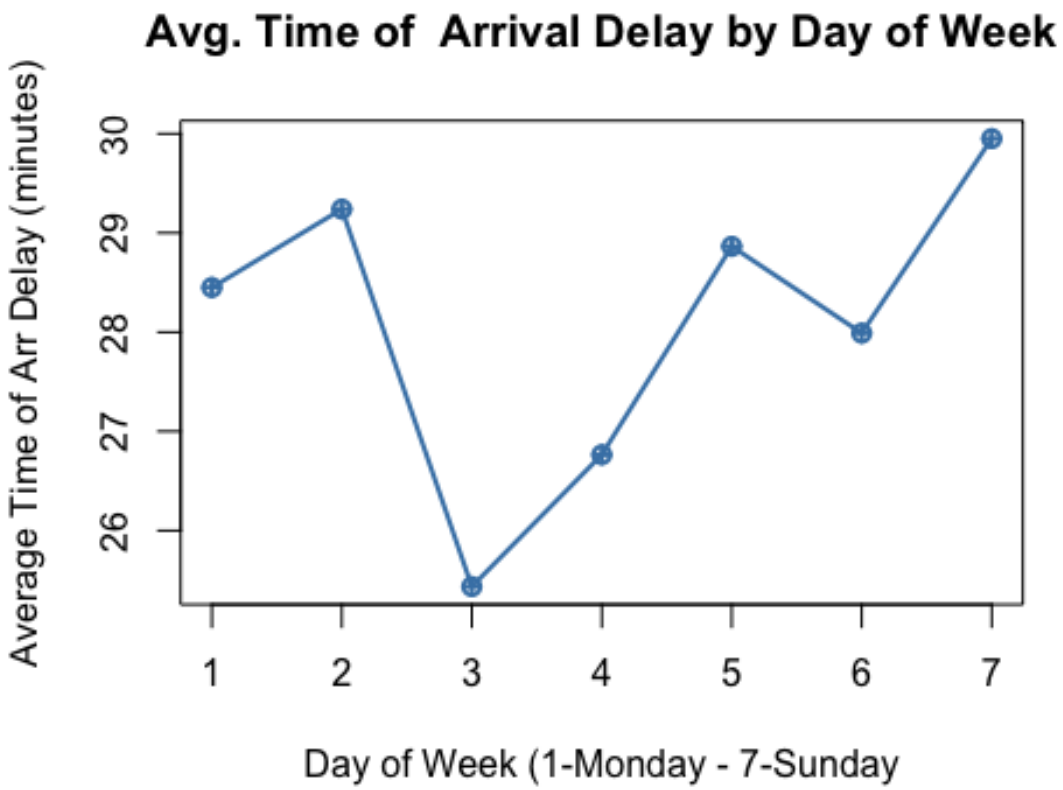
September, October and November had the shortest average arrival delays, while December had the longest.

```
#arrival delays by day of the week

df_arrdelayday = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$DayOfWeek,arrdelay,length))

plot(df_arrdelayday$arrdelay.DayOfWeek, df_arrdelayday$arrdelay.ArrDelay,
     xlab = "Day of Week (1-Monday - 7-Sunday)", ylab = "# of Arrival
Delays",
     main = "Number of Arrival delays by Day of Week", type ='o',
col='thistle',lwd=2, pch=10)
```



Saturdays had the lowest amount of arrival delays, while Thursdays and Fridays had the most.

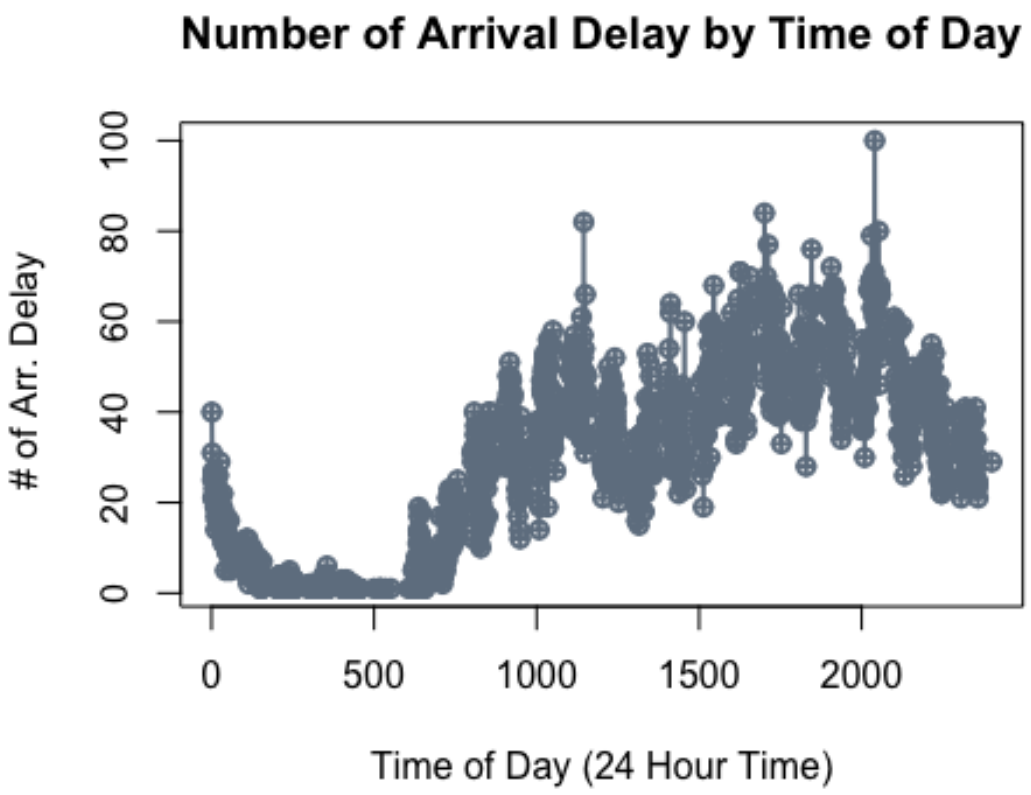
```
#average time of arrival delays by day of the week

df_arrdelayday = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$DayOfWeek,arrdelay,mean))

plot(df_arrdelayday$arrdelay.DayOfWeek, df_arrdelayday$arrdelay.ArrDelay,
     xlab = "Day of Week (1-Monday - 7-Sunday", ylab = "Average Time of Arr
Delay (minutes)",
     main = "Avg. Time of  Arrival Delay by Day of Week", type ='o',
col='steelblue',lwd=2, pch=10)
```



The Average time of arrival delays is the lowest on Wednesdays and highest on Sundays.

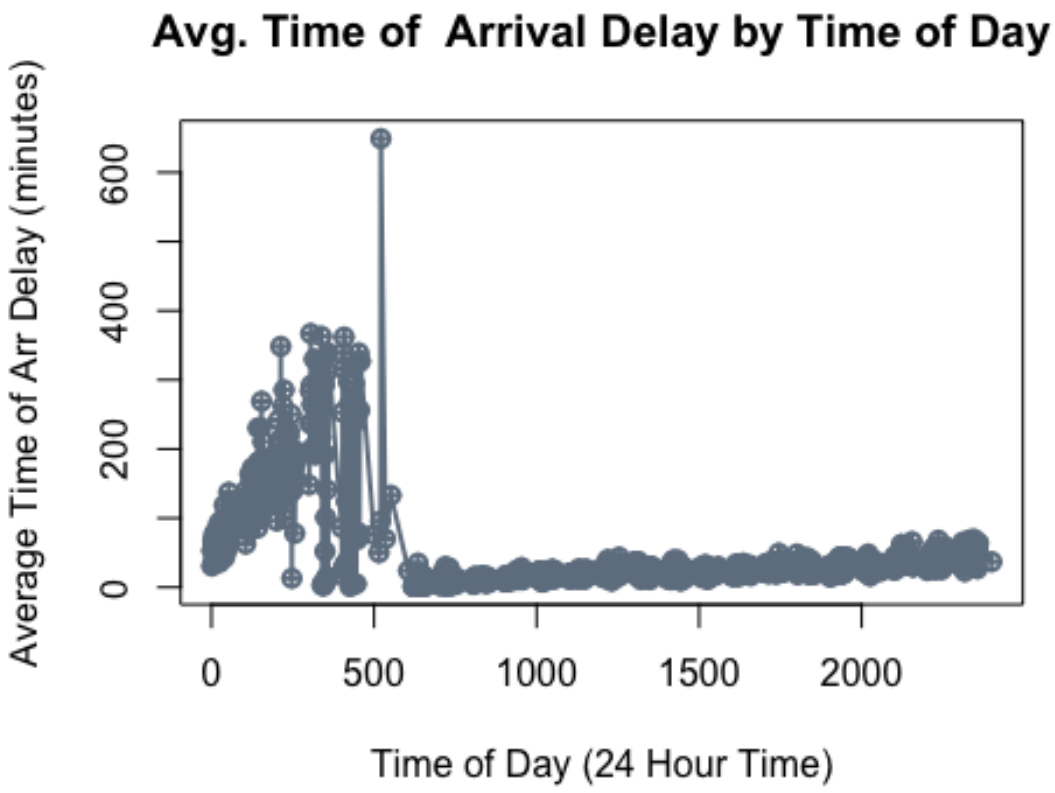
average time of arrival delays by time of day

```
#average time of arrival delays by time of day

df_arrdelaytime = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$ArrTime,arrdelay,length))

plot(df_arrdelaytime$arrdelay.ArrTime, df_arrdelaytime$arrdelay.ArrDelay,
     xlab = "Time of Day (24 Hour Time)", ylab = "# of Arr. Delay ",
     main = "Number of Arrival Delay by Time of Day", type ='o',
col='slategray',lwd=2, pch=10)
```



Between 1 am and 6 am there are the fewest amount of arrival delays. This is probably due to the lesser amount of flights during this time, since many airports don't have flights bewtween 2 am and 5 am . Around 11 am and 8 pm seem to ahve the the highest number of arrival delays. The best time for arrivals seems to be from around 6 am to 8 am. The number stays pretty consistent throughout the day after 10 am.

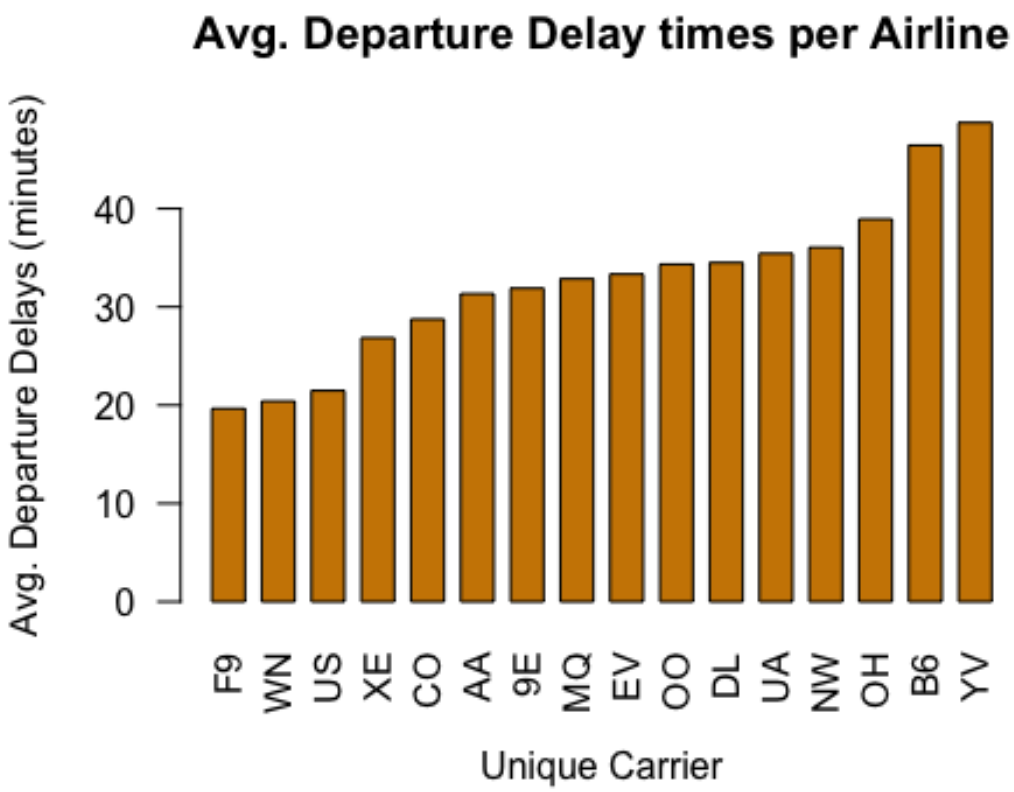
```
#average time of arrival delays by time of day

df_arrdelaytime = data.frame(aggregate(arrdelay$ArrDelay ~
arrdelay$ArrTime,arrdelay,mean))

plot(df_arrdelaytime$arrdelay.ArrTime, df_arrdelaytime$arrdelay.ArrDelay,
     xlab = "Time of Day (24 Hour Time)", ylab = "Average Time of Arr Delay
(minutes)",
     main = "Avg. Time of  Arrival Delay by Time of Day", type ='o',
col='slategray',lwd=2, pch=10)
```



Between Midnight and 5 am the average time of arrival delay is the highest (with a peak at around 5:30am). After about 6 am, average arrival delay by time of day is pretty low.

Departures

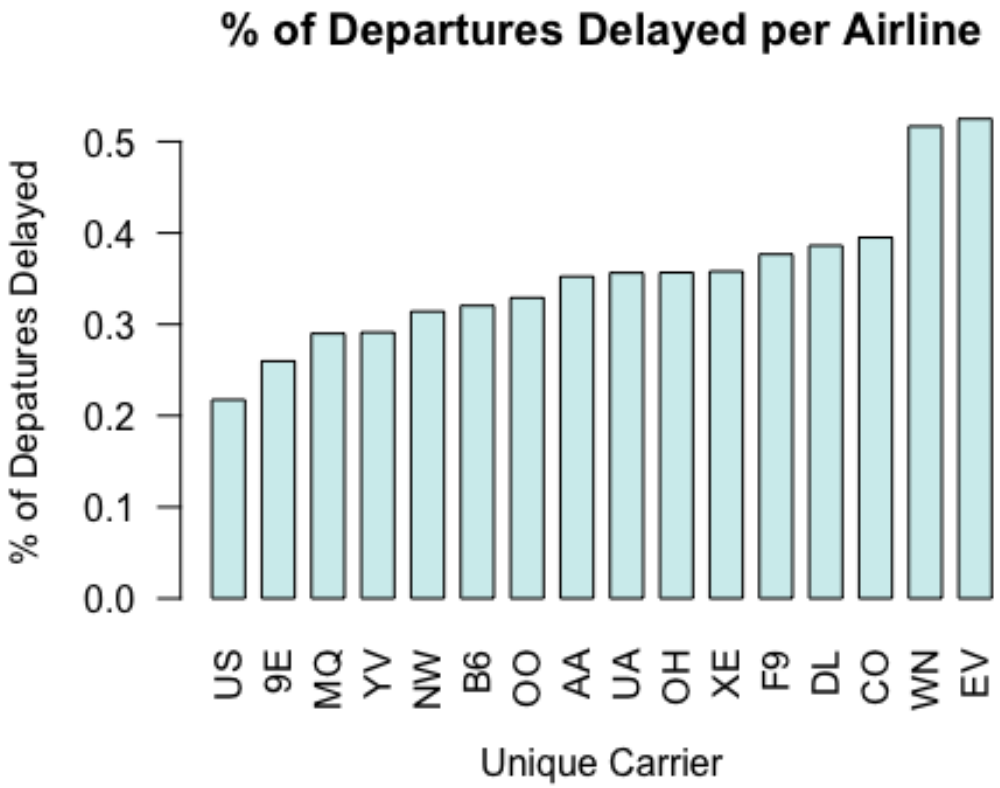
```
#average departure delay

depdelay = airlinedata[which(airlinedata[,16]>0),]
df_depdelay = data.frame(aggregate(depdelay$DepDelay ~
depdelay$UniqueCarrier,depdelay,mean))
df_depdelay = df_depdelay[order(df_depdelay$depdelay.DepDelay),]

barplot(df_depdelay$depdelay.DepDelay, names =
df_depdelay$depdelay.UniqueCarrier,
        xlab = "Unique Carrier", ylab = "Avg. Departure Delays (minutes)",
        main = "Avg. Departure Delay times per Airline", las = 2, space=.5,
col = 'orange3')
```



Departure delays are usually linked to arrival delays. Looking at departure delays is more important in understanding the likelihood of arriving at your destination on time/ or making conncections at another airport. The top three airlines with the longest departure delay times were Mesa Airlines (YV), PSA Airlines (OH), and Jet Blue(B6). The airlines with the longest delayed departure times is similar to the airlines with the longest avergae arrival delays.

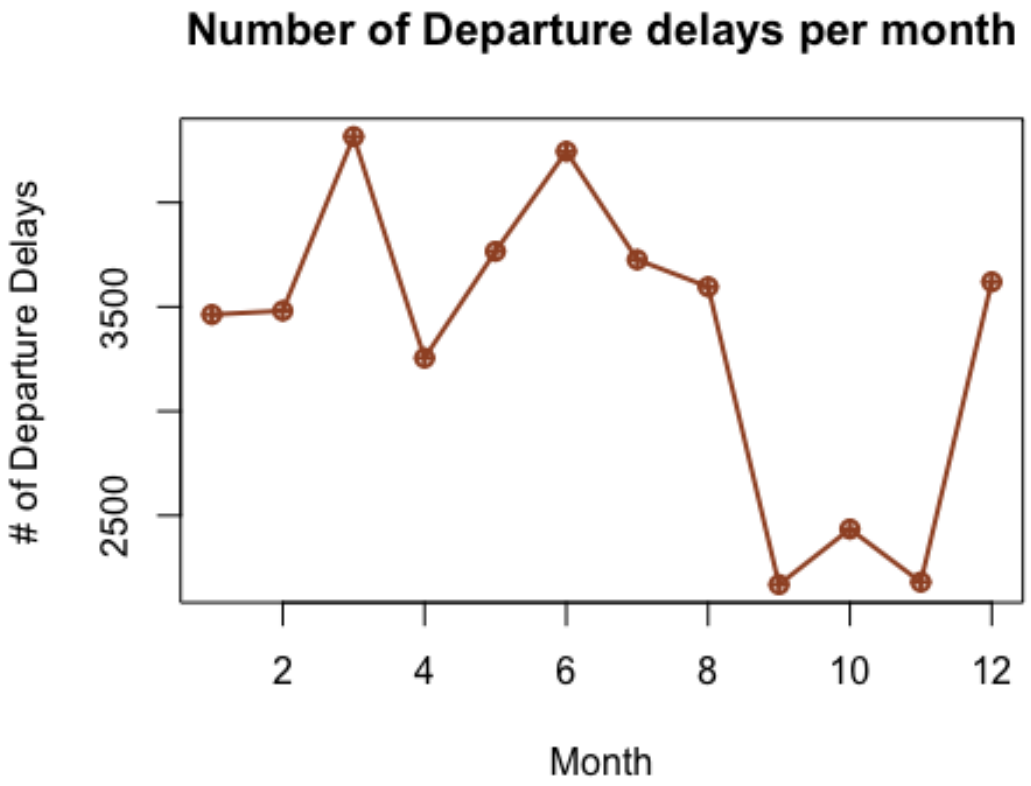
```
#% of of departures delayed per airline

df_delay_percent = data.frame(aggregate(depdelay$DepDelay ~
depdelay$UniqueCarrier,
                                        depdelay, length))
df_percent = data.frame(aggregate(airlinedata$FlightNum ~
airlinedata$UniqueCarrier,
                                  airlinedata, length))

percent_finaldf = merge(df_delay_percent, df_percent,
by.x="depdelay.UniqueCarrier", by.y="airlinedata.UniqueCarrier")
percent_finaldf = within(percent_finaldf, percent <-
depdelay.DepDelay/airlinedata.FlightNum)
percent_finaldf = percent_finaldf[order(percent_finaldf$percent),]
barplot(percent_finaldf$percent, names =
percent_finaldf$depdelay.UniqueCarrier,
```

```
        xlab = "Unique Carrier", ylab = "% of Depatures Delayed",
        main = "% of Departures Delayed per Airline", las=2, space=.5,
col='lightcyan2')
```



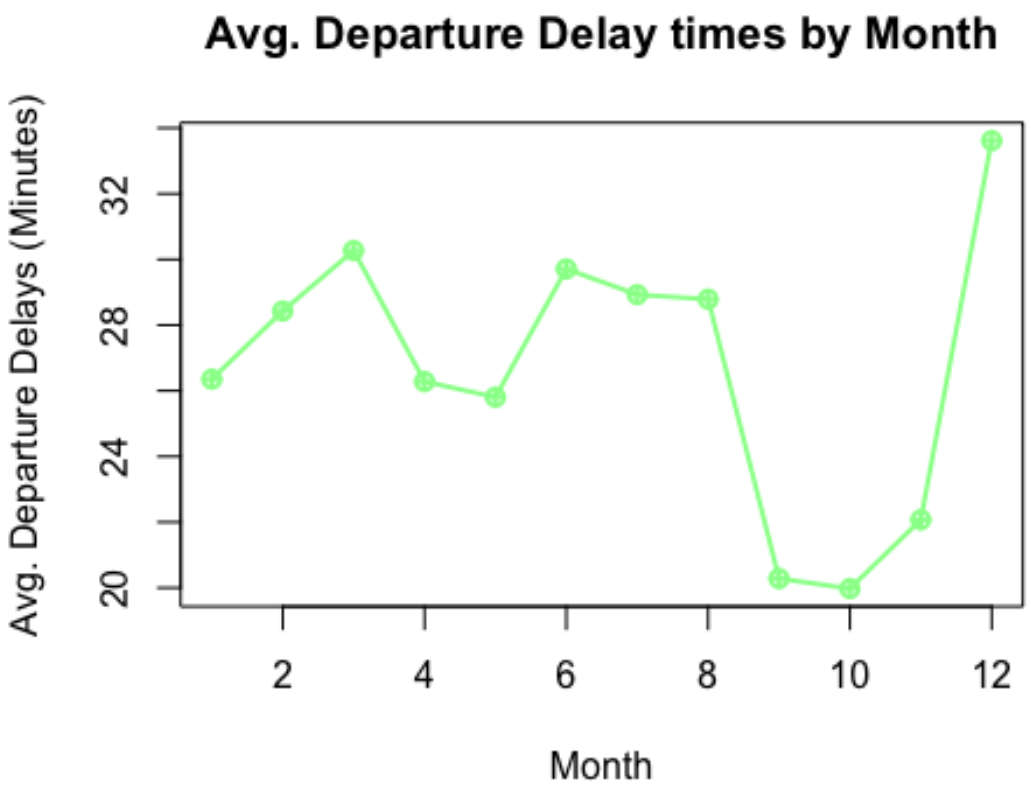
Looking at the % of Departures delayed per Airline, ExpressJet(EV) has 50% of flights delayed. Southwest (WN) also has close to 50% of flights delayed. The three airlines with the highest % of departures delayed (EpressJet, Southwest and Continential Airlines) are not the same airlines with the highest average departure delay times. This means that while one might have longer average departure delays, it does not mean they are delayed the most.

```
#departure delays per month
df_depdelaymon = data.frame(aggregate(depdelay$DepDelay ~
depdelay$Month,depdelay,length))
plot(df_depdelaymon$depdelay.Month, df_depdelaymon$depdelay.DepDelay,
     xlab = "Month", ylab = "# of Departure Delays",
     main = "Number of Departure delays per month", type ='o',
col='sienna',lwd=2, pch=10)
```



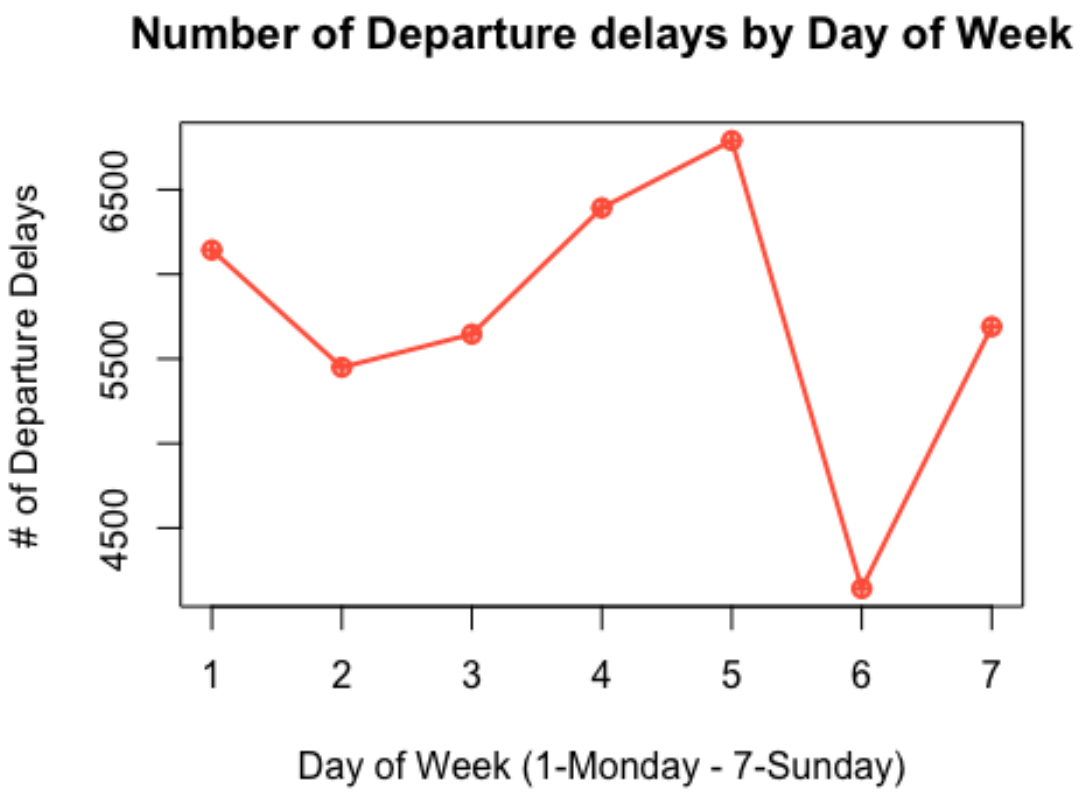
The highest number of departure delays occur in March and June. The months with the least amount of departure delays were September and November. Next, we will look at which months had the longest average delays.

```
#average departure delays per month
dfmm =data.frame(aggregate(depdelay$DepDelay~depdelay$Month, depdelay,mean))

plot(dfmm$depdelay.Month, dfmm$depdelay.DepDelay,
     xlab = "Month", ylab="Avg. Departure Delays (Minutes)",
     main = "Avg. Departure Delay times by Month", type='o',
     col = 'palegreen',lwd=2, pch =10)
```



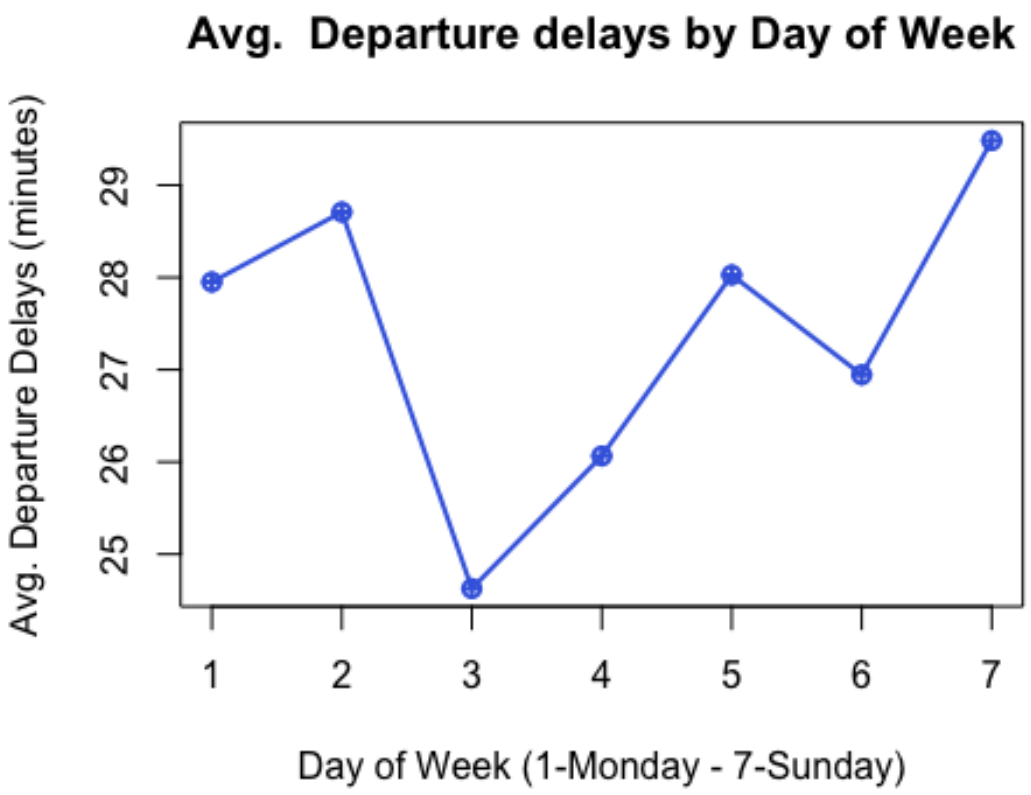
While March and June had highest number of delays. December had the longest average delays. September and October had the shortest average departure delays.

```
#deprture delays by day of the week

df_depdelayday = data.frame(aggregate(depdelay$DepDelay ~
depdelay$DayOfWeek,depdelay,length))

plot(df_depdelayday$depdelay.DayOfWeek, df_depdelayday$depdelay.DepDelay,
     xlab = "Day of Week (1-Monday - 7-Sunday)", ylab = "# of Departure
Delays",
     main = "Number of Departure delays by Day of Week", type ='o',
col='tomato',lwd=2, pch=10)
```



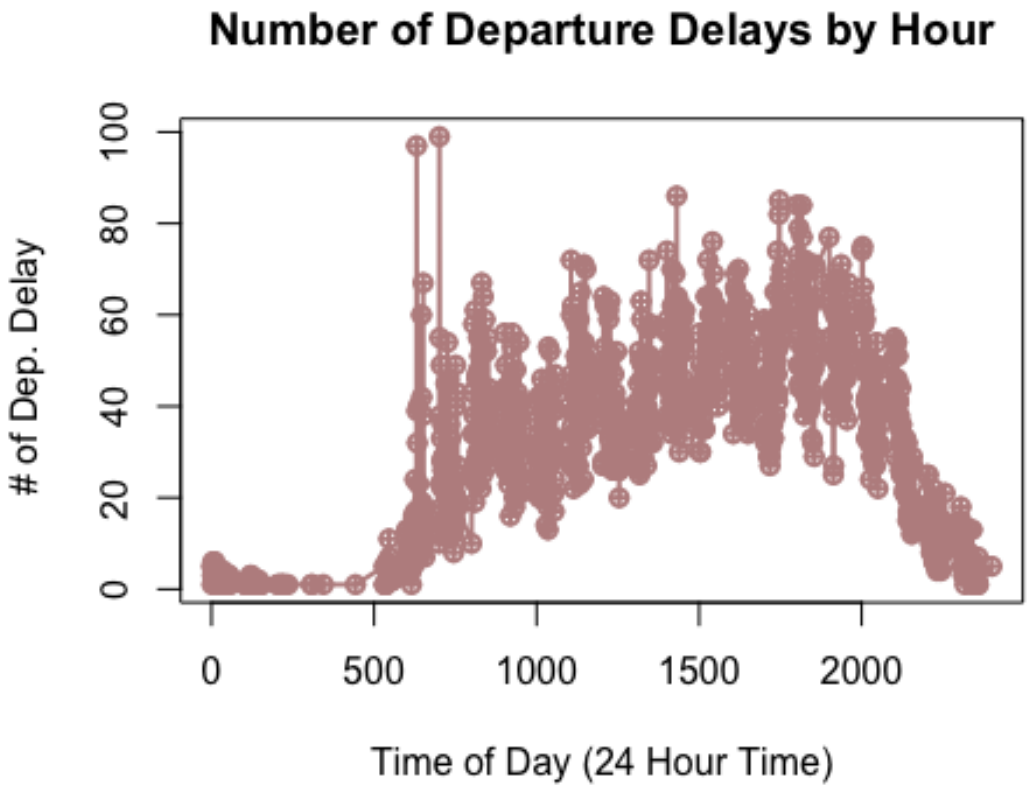
Fridays have the most number of Departure delays, followed by Thursdays. Saturdays had the lowest number of departure delays.

```
#average departure delays by day of the  week

df_depdelayday = data.frame(aggregate(depdelay$DepDelay ~
depdelay$DayOfWeek,depdelay,mean))

plot(df_depdelayday$depdelay.DayOfWeek, df_depdelayday$depdelay.DepDelay,
     xlab = "Day of Week (1-Monday - 7-Sunday)", ylab = "Avg. Departure
Delays (minutes)",
     main = "Avg.  Departure delays by Day of Week", type ='o',
col='royalblue',lwd=2, pch=10)
```



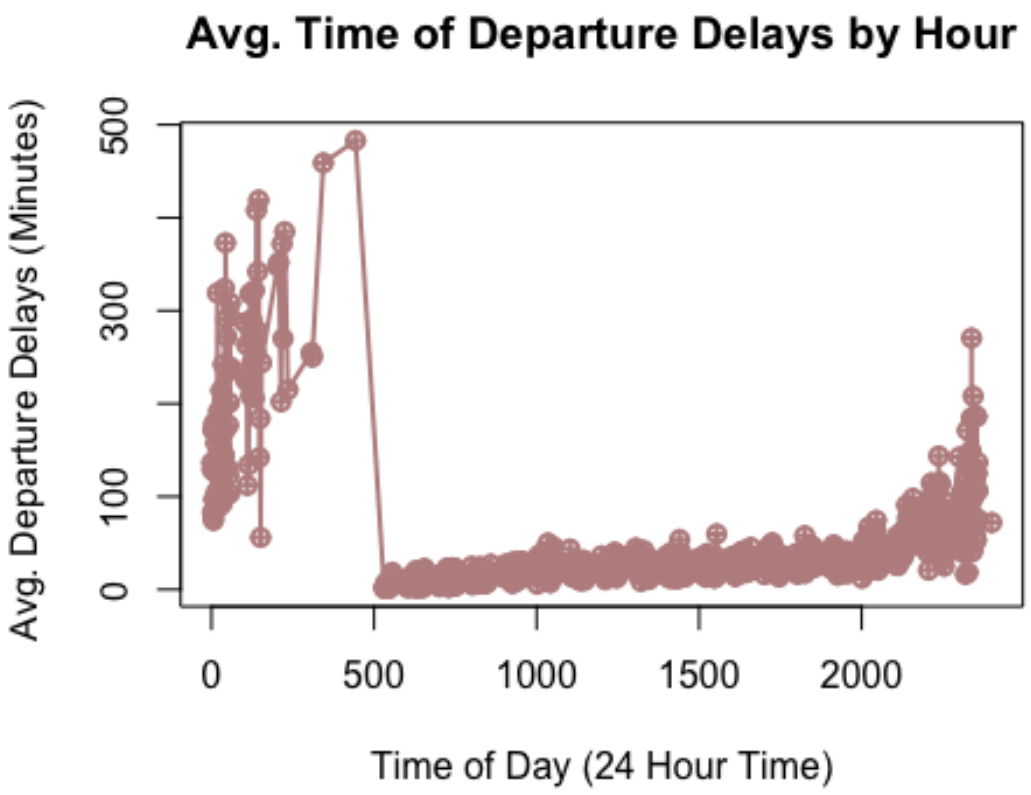
Wednesdays had the shortest average time of departure delays, while Sunday had the longest delays. Tuesdays had the second longest average departure delays.

```
#average time of departure delays by time of day

df_depdelaytime = data.frame(aggregate(depdelay$DepDelay ~
depdelay$DepTime,depdelay,length))

plot(df_depdelaytime$depdelay.DepTime, df_depdelaytime$depdelay.DepDelay,
     xlab = "Time of Day (24 Hour Time)", ylab = "# of Dep. Delay ",
     main = "Number of Departure Delays by Hour", type ='o',
col='rosybrown',lwd=2, pch=10)
```



It seems like the most delayed happen in the middle of the day between 6 am and 8 pm. There a re a coupleof peaks between 6 am and 7 pm. After 8pm, there are less departure delays and between midnight and 5 am there are almost zero. But, then again there are not many departing flights between midnight and 5 am.

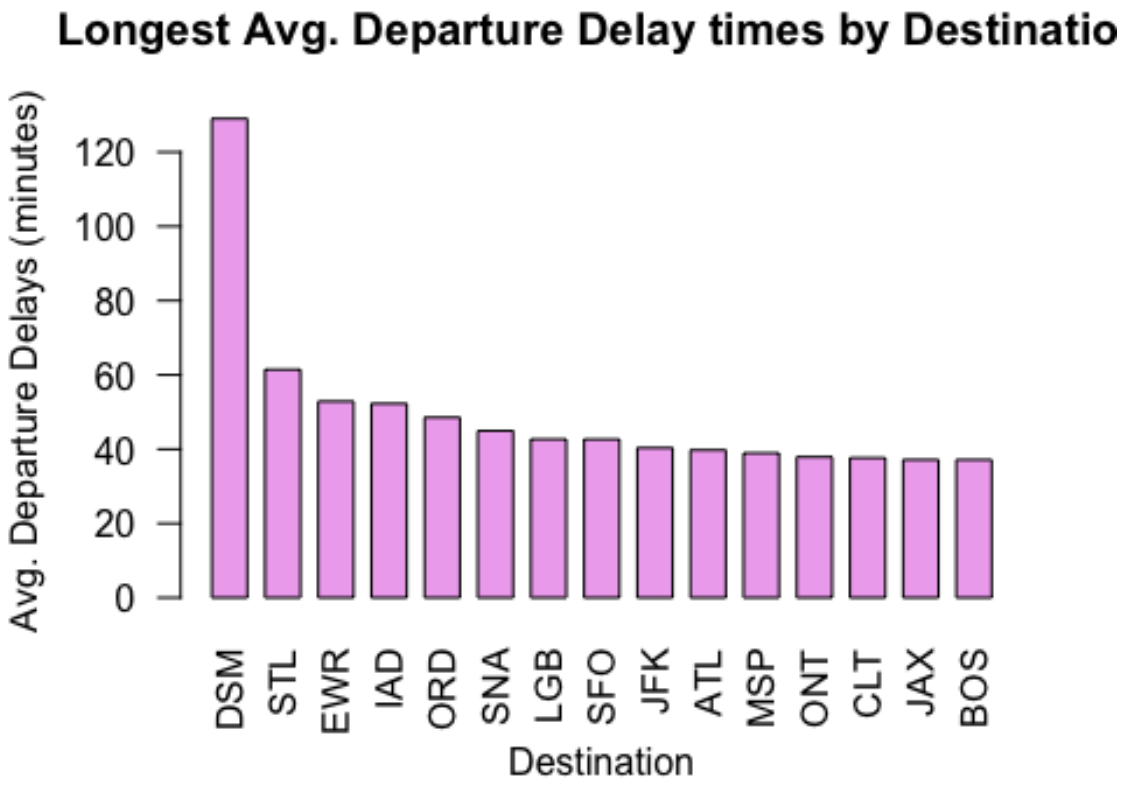
```
#average time of departure delays by time of day

df_depdelaytime = data.frame(aggregate(depdelay$DepDelay ~
depdelay$DepTime,depdelay,mean))


plot(df_depdelaytime$depdelay.DepTime, df_depdelaytime$depdelay.DepDelay,
     xlab = "Time of Day (24 Hour Time)", ylab = "Avg. Departure Delays
(Minutes) ",
     main = "Avg. Time of Departure Delays by Hour", type ='o',
col='rosybrown',lwd=2, pch=10)
```



While there are not many departure delays between midnight and 5 am, the ones that do occur are long delays. In contrast the delays during the peak delay times seem to be shorts. The night delays (after 8pm) are longer, but not as long as the early morning delays.

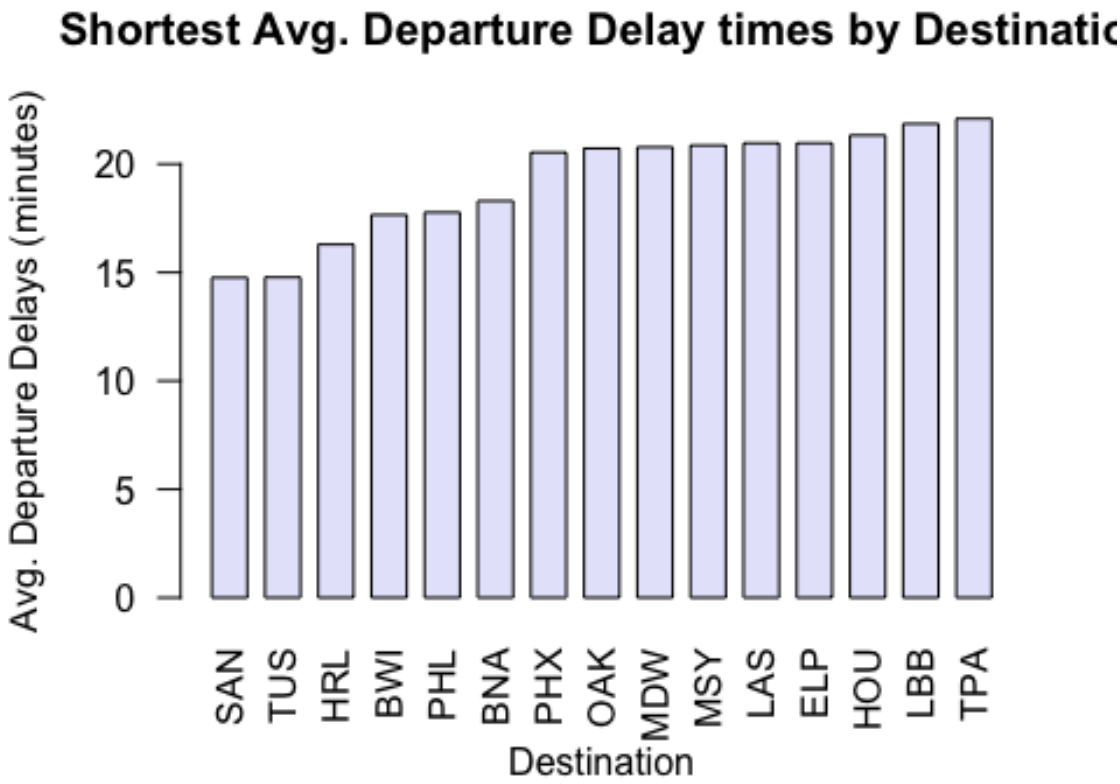
Destinations and Origins

```
departures = subset(airlinedata, Origin == 'AUS')

#average departure delay by destination
departdelay = departures[which(departures[,16]>0),]
df_depdelays = data.frame(aggregate(departdelay$DepDelay ~ departdelay$Dest,
departdelay, mean))
df_depdelays2 = df_depdelays[order(-
df_depdelays$departdelay.DepDelay),][1:15,]

barplot(df_depdelays2$departdelay.DepDelay, names =
df_depdelays2$departdelay.Dest,xlab = "Destination", ylab = 'Avg. Departure
Delays (minutes)', main = 'Longest Avg. Departure Delay times by
Destination', las =2, space=.5, col='plum2')
```



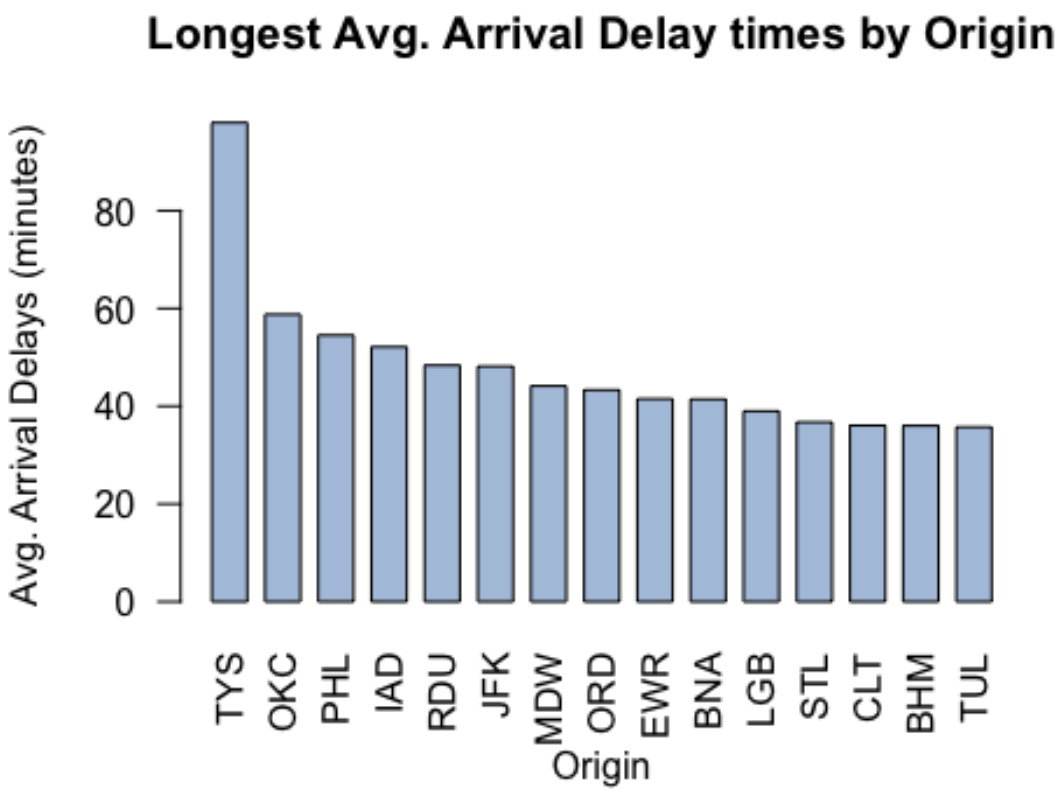
The above graph shows which airports are the destinations in which there are the longest average delays. The longest delays happen for flights headed to Des Moines, Iowa (DSM). THe second longest are for St. Louis (STL) and the third longest delays are for flights headed to Newark, NJ (EWR).

```
#destinations with the shortest average delays

departures = subset(airlinedata, Origin == 'AUS')

#average departure delay by destination
departdelay = departures[which(departures[,16]>0),]
df_depdelays = data.frame(aggregate(departdelay$DepDelay ~ departdelay$Dest,
departdelay, mean))
df_depdelays2 =
df_depdelays[order(df_depdelays$departdelay.DepDelay),][1:15,]

barplot(df_depdelays2$departdelay.DepDelay, names =
df_depdelays2$departdelay.Dest,xlab = "Destination", ylab = 'Avg. Departure
Delays (minutes)', main = ' Shortest Avg. Departure Delay times by
Destination', las =2, space=.5, col='lavender')
```



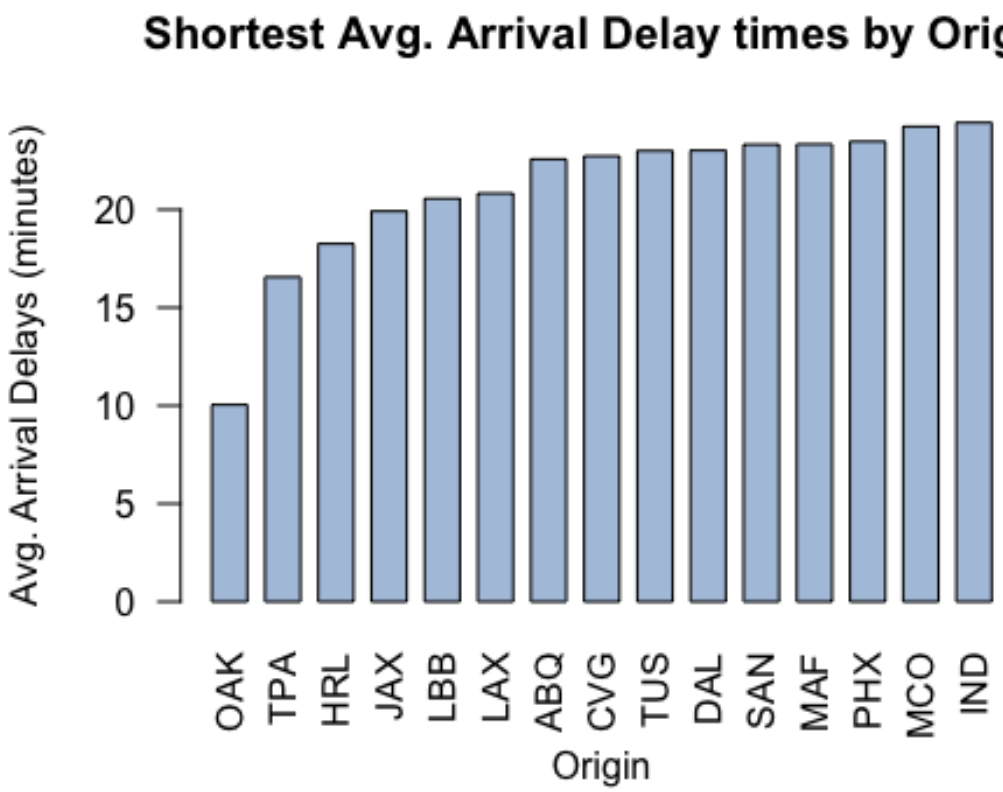
The three destinations with the shortest average departure delays are San Diego (SAN), Tucson (TUS) and Valley International Airport (HRL).

```
#average arrival delay by Origin
arrivals = subset(airlinedata, Origin != 'AUS')
arrivedelay = arrivals[which(arrivals[,15]>0),]
df_arrdelays = data.frame(aggregate(arrivedelay$ArrDelay ~
arrivedelay$Origin, arrivedelay, mean))
df_arrdelays2 = df_arrdelays[order(-df_arrdelays$arrivedelay.ArrDelay),
][1:15,]

barplot(df_arrdelays2$arrivedelay.ArrDelay, names =
df_arrdelays2$arrivedelay.Origin,xlab = "Origin", ylab = 'Avg. Arrival Delays
(minutes)', main = 'Longest Avg. Arrival Delay times by Origin', las =2,
space=.5, col='lightsteelblue')
```



The longest average arrival delays happens with planes coming from McGhee Tyson Airport in Knoxville, TN (TYS), Will Rogers World Airport in Oklahoma City, and Philadelphia (PHL).

```
#average shortest arrival delay by Origin
arrivals = subset(airlinedata, Origin != 'AUS')
arrivedelay = arrivals[which(arrivals[,15]>0),]
df_arrdelays = data.frame(aggregate(arrivedelay$ArrDelay ~
arrivedelay$Origin, arrivedelay, mean))
df_arrdelays2 = df_arrdelays[order(df_arrdelays$arrivedelay.ArrDelay),
][1:15,]

barplot(df_arrdelays2$arrivedelay.ArrDelay, names =
df_arrdelays2$arrivedelay.Origin,xlab = "Origin", ylab = 'Avg. Arrival Delays
(minutes)', main = 'Shortest Avg. Arrival Delay times by Origin', las =2,
space=.5, col='lightsteelblue')
```



The shorest average arrival delays happen with planes coming from Oakland, California (OAK), Tampa, Florida (TPA) and Valley International (HRL).

Author Attribution

Preprocessing

First we need to read in all of our data and then preprocess it to make everything lowercase, remove numbers, remove punctuation, remove excess white spaces, and remove stopwords. We then create the document term matrix for both the train and test together to address the issue of some terms appearing in the test but not train or vice versa.

```
library(tm)

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

authors_train <- Sys.glob('~/Documents/UT/Summer Classes/Intro to Predictive
Modeling/Part 2/HW 2/ReutersC50/C50train/*')
file_list_train = NULL
labels_train = NULL
```

```
authors_test = Sys.glob('~/Documents/UT/Summer Classes/Intro to Predictive
Modeling/Part 2/HW 2/ReutersC50/C50test/*')
file_list_test = NULL
labels_test = NULL

for(i in authors_train) {
  author_name_train = substring(i, first = 84)
  files_to_add_train = Sys.glob(paste0(i, '/*.txt'))
  file_list_train = append(file_list_train, files_to_add_train)
  labels_train = append(labels_train, rep(author_name_train,
length(files_to_add_train)))
}

author_names = NULL
for(i in authors_test) {
  author_name_test = substring(i, first = 83)
  author_names = append(author_names, author_name_test)
  files_to_add_test = Sys.glob(paste0(i, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add_test)
  labels_test = append(labels_test, rep(author_name_test,
length(files_to_add_test)))
}

file_list <- append(file_list_train,file_list_test) #combine train and test
for simplicity
labels <- unique(append(labels_train,labels_test))

all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
all_corpus = Corpus(VectorSource(all_docs))
#names(all_corpus) = labels

all_corpus = tm_map(all_corpus, content_transformer(tolower)) # make
everything lowercase

## Warning in tm_map.SimpleCorpus(all_corpus, content_transformer(tolower)):
## transformation drops documents

all_corpus = tm_map(all_corpus, content_transformer(removeNumbers)) # remove
numbers

## Warning in tm_map.SimpleCorpus(all_corpus,
## content_transformer(removeNumbers)): transformation drops documents

all_corpus = tm_map(all_corpus, content_transformer(removePunctuation)) #
remove punctuation

## Warning in tm_map.SimpleCorpus(all_corpus,
## content_transformer(removePunctuation)): transformation drops documents
```

```
all_corpus = tm_map(all_corpus, content_transformer(stripWhitespace)) ##
remove excess white-space

## Warning in tm_map.SimpleCorpus(all_corpus,
## content_transformer(stripWhitespace)): transformation drops documents

all_corpus = tm_map(all_corpus, content_transformer(removeWords),
stopwords("en"))  #remove stop words

## Warning in tm_map.SimpleCorpus(all_corpus,
## content_transformer(removeWords), : transformation drops documents

#creating a document term matrix with tf idf scores
dtm <- DocumentTermMatrix(all_corpus,control = list(weighting = function(x)
weightTfIdf(x, normalize = FALSE)))
dtm <- removeSparseTerms(dtm, 0.95)
```

Naive Bayes

We first try the Naive Bayes model on the training data set. We find the weight vector for each author that contains the weight of each word used by that author. We smooth the data so that when words are not in an author's bag of words, a small non-zero probability is given so that the posterior probabilities don't become zero.

```
#Naive Bayes

x <- as.matrix(dtm)
x_train <- x[1:2500,]
x_test <- x[2501:5000,]
smooth_count = 1/nrow(x_train)

author_sums <- rowsum(x_train +smooth_count, labels_train)
wt <- rowSums(author_sums)
author_wt <- log(author_sums/wt)    #  log of prob of word occuring with
particular author

predicted_probabilities <- x_test%*%t(author_wt)  #use x_test to multiply log
probabilities from weights of training set authors

predicted_authors = NULL
for( i in 1:2500) {
  predicted_authors =
c(predicted_authors,which.max(predicted_probabilities[i,]))
  }

predicted_authors <- as.data.frame(predicted_authors)
predicted_authors$actual <- rep(1:50,each = 50)

#assign author with max sum of probabilities for each of 2500 test points in
new list
```

```
results_bayes=table(predicted_authors$predicted_authors,
predicted_authors$actual)
results_bayes
```

```
##
##       1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
##   1  42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   2   0 30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0
##   3   0  1 28  0  1  0  0  0  3  0  0  0  0  0  0  0  5  7  0  7  0  4  0
##   4   0  0  0  7  0  0  0  0  0  0  0  0  2  0  7  0  0  0  0  0  1  0  0
##   5   0  0  0  0 27  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   6   1  0  0  0  0 36  1  7  0  2  1  0  0  0  0  0  0  0  0  0  0  0  9
##   7   1  0  0  0  0  0 21  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0
##   8   0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   9   0  0  0  0  0  1  0  0 10  0  0  0  0  0  0  0  0  0  1  3  0  1  0
##   10  0  0  0  0  0  0  0  0  0 26  0  0  0  0  0  0  0  0  0  0  0  0  1
##   11  0  0  0  0  0  0  0  0  0  0 48  0  0  0  0  0  0  0  0  0  0  0  0
##   12  0  0  0  2  0  0  1  0  0  0  0 37  0  0  0  0  0  0  0  0  0  0  0
##   13  0  0  0  0  0  0 21  0  0  0  0  0 16  0  0  0  0  0  0  0  0  0  0
##   14  0  0  0 15  0  0  1  1  3  0  0  3 13  0 19  1  0  0  1  0  6  1  0
##   15  0  7  0  0  0  0  1  0  0  0  0  0  1 22  0  0  0  0  7  0  0  0  0
##   16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 48  0  0  0  0  0  0  0
##   17  0  0 20  0  1  0  0  1  2  0  0  0  0  1  0  0  0 32  0  0  0  2  0
##   18  0 12  0  0  0  0  0  0  0  0  0  0  1 25  0  0  0  0 34  0  0  0  0
##   19  0  0  1  0  0  0  0  0  5  0  0  0  0  0  0  0  4  0  1 30  0  1  0
##   20  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0 30  1  0  1  0  0  0
##   21  0  0  0  0  0  0  1  1  3  0  0  0  1  0  0  0  0  0  0  0 37  0  0
##   22  0  0  0  0  2  0  0  0  1  3  0  0  0  0  0  0  1  0  0  1  0 30  0
##   23  0  0  0  0  0  4  0  5  0  4  0  0  1  0  0  0  0  0  0  0  0  0 26
##   24  0  0  0  0  6  0  2  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  1
##   25  0  0  0  0  0  0  0  0  7  1  0  0  0  1  0  0  2  2  0  2  0  5  0
##   26  0  0  0  0  0  0  0  0  0  0  0  0  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  0  0
##   28  0  0  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  0  0  0  0
##   30  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   31  0  0  0  0  5  0  0  0  0  0  0  0  2  1  0  0  0  1  0  0  0  0  0
##   32  1  0  0  0  0  0  0  2  0  3  0  0  0  0  0  0  0  1  0  0  0  0  1
##   33  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  3  0  0
##   34  0  0  0  0  3  0  0  1  1  0  0  0  0  0  0  0  0  0  0  0  0  0  1
##   35  0  0  0  1  0  0  0  0  0  0  0  0  0  0  6  0  0  0  3  0  2  0  0
##   36  0  0  0  0  0  0  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  3
##   37  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0  6
##   38  0  0  0  3  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##   39  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   40  0  0  0  0  0  2  0  2  0  1  0  0  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  0  0
##   42  5  0  0  0  0  1  0  0  0  4  0  0  0  0  0  0  0  0  0  0  0  0  1
##   43  0  0  0  0  4  0  0  0  0  0  1  6  1  0  2  0  0  0  0  0  0  0  0
##   44  0  0  0 10  0  0  0  0  0  0  0  3  2  0 10  0  0  0  2  0  1  0  0
```

```
##   45  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  5  0  1  0  2  0
##   46  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   47  0  0  1  0  0  2  0  2  0  3  0  0  0  0  0  0  0  0  0  0  0  0  1
##   48  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  8  1  0  5  0  4  0
##   49  0  0  0  0  0  4  1 18  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   50  0  0  0 11  0  0  0  0  0  0  0  0  2  0  5  1  0  0  0  0  0  0  0
##
##      24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46
##   1   0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  1  0  0  0  0
##   2   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0
##   3   0  3  0  0  0  0  0  4  0  0  0  0  0  0  0  1  0  1  0  0  0 18  0
##   4   0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   5   4  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   6   0  0  1  4  0  0  2  0  6  0  0  0  1  5  0  0  2  0  0  0  0  0  0
##   7   0  0  0  6  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   8   0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   9   0  1  0  0  0  0  2  0  0  0  0  0  0  1  0  1  0  0  0  0  0  0  0
##   10  0  0 11  1  0  0  0  0  4  0  0  0  2  0  0  0  0  0  4  0  0  0  0
##   11  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   12  0  0  0  0  1  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  1  0  0
##   13  0  0  0  2  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   14  0  0  0  0  4  2  0  0  0  2  0 12  0  0  1  0  0  1  0  1 23  0  0
##   15  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   16  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0
##   17  2  0  0  0  0  0  1  0  0  0  4  0  0  0  0  4  1  0  0  0  0  7  0
##   18  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   19  0  1  0  0  0  0  2  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0
##   20  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   21  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0
##   22  1  6  0  0  0  0  3  0  1  0  0  0  0  0  0  4  0  1  0  0  0  0  0
##   23  0  1  1  0  0  0  0  0  1  0  1  0  0 10  0  0  5  0  1  0  0  0  0
##   24 30  0  0  0  0  0  0 11  0  0  1  0  0  0  0  0  0  0  0  1  0  0  1
##   25  5 34  0  0  0  0  3  0  0  0  0  0  1  0  0  0  2  1  0  0  0  0  0
##   26  0  0 28  0  0  0  0  0  1  0  0  0  0  3  0  0  0  0  0  0  0  0  0
##   27  0  0  0 31  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##   28  0  0  0  0 39  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##   29  0  0  0  0  0 47  2  0  0  2  0  1  0  0  0  0  0  0  0  0  1  0  0
##   30  0  0  0  0  0  0 24  0  0  0  0  0  0  0  0  6  1  0  0  0  0  0  0
##   31  2  0  0  1  0  0  0 23  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0
##   32  0  0  0  0  0  0  1  0 17  0  0  0  1  0  0  0  1  0  1  0  0  0  0
##   33  0  0  0  0  0  0  1  0  0 44  0  0  0  0  0  0  0  0  0  0  0  0  0
##   34  1  0  0  1  0  0  0  1  0  0 36  0  0  1  0  0  1  2  0  0  0  0  0
##   35  0  0  0  0  2  0  0  2  0  0  0 14  0  0  0  0  0  0  0  0  4  0  1
##   36  1  0  1  0  0  0  0  0  2  1  0  0 36  3  0  1  0  4  0  0  0  0  0
##   37  0  0  0  0  0  0  0  0  3  0  3  0  2 24  0  0  0  2  0  0  0  0  0
##   38  0  0  0  0  0  0  0  0  0  0  0  3  0  0 27  0  0  0  0  9  4  0 12
##   39  0  0  0  0  0  1  7  0  0  0  1  0  0  0  0 31  0  0  0  0  0  0  0
##   40  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 36  0  0  0  0  0  0
##   41  0  0  0  0  0  0  0  0  2  0  0  0  1  0  0  0  0 30  0  0  0  0  0
##   42  0  1  3  3  0  0  0  0  8  0  1  0  4  1  0  0  0  2 34  0  0  0  0
```

```
##   43  0  1  0  0  0  0  0  3  0  0  0  0  0  0  2  0  0  0  0 28  0  0 12
##   44  0  0  0  0  1  0  0  2  0  1  0  9  0  0  2  0  0  2  0  2 11  0  1
##   45  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0 24  0
##   46  0  0  0  0  1  0  0  0  0  0  0  2  0  0 17  0  0  0  0  8  0  0 22
##   47  0  0  5  0  0  0  0  0  2  0  1  0  2  0  0  0  0  0  7  0  0  0  0
##   48  2  1  0  0  0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0
##   49  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  1  0  2  0  0  0  0
##   50  2  0  0  0  2  0  0  0  0  0  0  6  0  0  0  0  0  1  0  0  6  0  1
##
##      47 48 49 50
##   1   3  0  0  0
##   2   0  0  0  0
##   3   0  2  0  0
##   4   0  0  0  4
##   5   0  1  0  0
##   6   1  0  1  0
##   7   0  0  0  0
##   8   0  0 13  0
##   9   0  0  0  0
##   10  9  0  0  0
##   11  0  0  0  0
##   12  0  0  0  2
##   13  0  0  0  0
##   14  0  0  3  7
##   15  0  0  0  0
##   16  0  0  0  0
##   17  0  0  0  0
##   18  0  0  0  0
##   19  0  5  0  0
##   20  0  0  0  0
##   21  0  0  0  0
##   22  0  3  0  0
##   23  3  0  0  0
##   24  0  1  0  0
##   25  0  0  1  0
##   26  0  0  0  0
##   27  0  0  0  0
##   28  0  0  0  1
##   29  0  0  0  0
##   30  0  0  0  0
##   31  0  2  0  0
##   32  0  0  1  0
##   33  0  0  0  0
##   34  1  0  1  0
##   35  0  0  0  4
##   36  1  0  2  0
##   37  0  0  5  0
##   38  0  0  0  1
##   39  0  0  0  0
##   40  0  0  1  0
```

```
##   41  2  0  0  0
##   42  9  0  1  0
##   43  0  0  1  4
##   44  0  0  0  8
##   45  0  4  0  0
##   46  0  0  0  3
##   47 21  0  0  0
##   48  0 32  0  0
##   49  0  0 20  0
##   50  0  0  0 16
```

```
correct = NULL
for (i in 1:nrow(results_bayes)) {
  correct = append(correct, results_bayes[i,i])
}

correct_by_author = data.frame(correct, row.names = author_names)
correct_by_author
```

```
##                                           correct
## HW 2/ReutersC50/C50test/AaronPressman          42
## HW 2/ReutersC50/C50test/AlanCrosby             30
## HW 2/ReutersC50/C50test/AlexanderSmith         28
## HW 2/ReutersC50/C50test/BenjaminKangLim         7
## HW 2/ReutersC50/C50test/BernardHickey          27
## HW 2/ReutersC50/C50test/BradDorfman            36
## HW 2/ReutersC50/C50test/DarrenSchuettler       21
## HW 2/ReutersC50/C50test/DavidLawder             4
## HW 2/ReutersC50/C50test/EdnaFernandes          10
## HW 2/ReutersC50/C50test/EricAuchard            26
## HW 2/ReutersC50/C50test/FumikoFujisaki         48
## HW 2/ReutersC50/C50test/GrahamEarnshaw         37
## HW 2/ReutersC50/C50test/HeatherScoffield       16
## HW 2/ReutersC50/C50test/JanLopatka              0
## HW 2/ReutersC50/C50test/JaneMacartney           0
## HW 2/ReutersC50/C50test/JimGilchrist           48
## HW 2/ReutersC50/C50test/JoWinterbottom          0
## HW 2/ReutersC50/C50test/JoeOrtiz                0
## HW 2/ReutersC50/C50test/JohnMastrini            1
## HW 2/ReutersC50/C50test/JonathanBirt            1
## HW 2/ReutersC50/C50test/KarlPenhaul            37
## HW 2/ReutersC50/C50test/KeithWeir              30
## HW 2/ReutersC50/C50test/KevinDrawbaugh         26
## HW 2/ReutersC50/C50test/KevinMorrison          30
## HW 2/ReutersC50/C50test/KirstinRidley          34
## HW 2/ReutersC50/C50test/KouroshKarimkhany      28
## HW 2/ReutersC50/C50test/LydiaZajc              31
## HW 2/ReutersC50/C50test/LynneO'Donnell         39
## HW 2/ReutersC50/C50test/LynnleyBrowning        47
## HW 2/ReutersC50/C50test/MarcelMichelson        24
```

```
## HW 2/ReutersC50/C50test/MarkBendeich           23
## HW 2/ReutersC50/C50test/MartinWolk             17
## HW 2/ReutersC50/C50test/MatthewBunce           44
## HW 2/ReutersC50/C50test/MichaelConnor          36
## HW 2/ReutersC50/C50test/MureDickie             14
## HW 2/ReutersC50/C50test/NickLouth              36
## HW 2/ReutersC50/C50test/PatriciaCommins        24
## HW 2/ReutersC50/C50test/PeterHumphrey          27
## HW 2/ReutersC50/C50test/PierreTran             31
## HW 2/ReutersC50/C50test/RobinSidel             36
## HW 2/ReutersC50/C50test/RogerFillion           30
## HW 2/ReutersC50/C50test/SamuelPerry            34
## HW 2/ReutersC50/C50test/SarahDavison           28
## HW 2/ReutersC50/C50test/ScottHillis            11
## HW 2/ReutersC50/C50test/SimonCowell            24
## HW 2/ReutersC50/C50test/TanEeLyn               22
## HW 2/ReutersC50/C50test/TheresePoletti         21
## HW 2/ReutersC50/C50test/TimFarrand             32
## HW 2/ReutersC50/C50test/ToddNissen             20
## HW 2/ReutersC50/C50test/WilliamKazer           16

sum(correct_by_author)/2500 #accuracy

## [1] 0.4936
```

Naive Bayes only acheives an accuracy of 49% which is not very good. Jim Gilchrist and Fumiko Fujisaki are predicted best (48 correct, 2 wrongly attributed) , while authors like John Mastrini are predicted poorly. Some of the most common misclassifications were Scott Hillis for Jane Macartney and Tan Eelyn for Peter Humphrey.

Random Forest

The next model we will try is Random Forest. Since our dataset is large, we specify 200 as our number of trees and limit mtry to 6.

```
#Random Forest

library(randomForest)

## randomForest 4.6-14

## Type rfNews() to see new features/changes/bug fixes.

set.seed(11)

rf.fit = randomForest(x = x_train, y = as.factor(labels_train), mtry=6,
ntree=200)
rf.pred = predict(rf.fit, data=x_test)
rf_results = table(labels_test, rf.pred)
rf_correct = NULL
for (i in 1:nrow(rf_results)) {
```

```
  rf_correct = append(rf_correct, rf_results[i, i])
}

rf_correct_by_author = data.frame(rf_correct, row.names = author_names)
rf_correct_by_author
```

```
##                                           rf_correct
## HW 2/ReutersC50/C50test/AaronPressman             48
## HW 2/ReutersC50/C50test/AlanCrosby                41
## HW 2/ReutersC50/C50test/AlexanderSmith            30
## HW 2/ReutersC50/C50test/BenjaminKangLim           32
## HW 2/ReutersC50/C50test/BernardHickey             33
## HW 2/ReutersC50/C50test/BradDorfman               29
## HW 2/ReutersC50/C50test/DarrenSchuettler          42
## HW 2/ReutersC50/C50test/DavidLawder               42
## HW 2/ReutersC50/C50test/EdnaFernandes             30
## HW 2/ReutersC50/C50test/EricAuchard               32
## HW 2/ReutersC50/C50test/FumikoFujisaki            45
## HW 2/ReutersC50/C50test/GrahamEarnshaw            38
## HW 2/ReutersC50/C50test/HeatherScoffield          41
## HW 2/ReutersC50/C50test/JanLopatka                23
## HW 2/ReutersC50/C50test/JaneMacartney             43
## HW 2/ReutersC50/C50test/JimGilchrist              50
## HW 2/ReutersC50/C50test/JoWinterbottom            37
## HW 2/ReutersC50/C50test/JoeOrtiz                  33
## HW 2/ReutersC50/C50test/JohnMastrini              34
## HW 2/ReutersC50/C50test/JonathanBirt              45
## HW 2/ReutersC50/C50test/KarlPenhaul               45
## HW 2/ReutersC50/C50test/KeithWeir                 37
## HW 2/ReutersC50/C50test/KevinDrawbaugh            27
## HW 2/ReutersC50/C50test/KevinMorrison             30
## HW 2/ReutersC50/C50test/KirstinRidley             28
## HW 2/ReutersC50/C50test/KouroshKarimkhany         43
## HW 2/ReutersC50/C50test/LydiaZajc                 45
## HW 2/ReutersC50/C50test/LynneO'Donnell            49
## HW 2/ReutersC50/C50test/LynnleyBrowning           37
## HW 2/ReutersC50/C50test/MarcelMichelson           45
## HW 2/ReutersC50/C50test/MarkBendeich              41
## HW 2/ReutersC50/C50test/MartinWolk                32
## HW 2/ReutersC50/C50test/MatthewBunce              44
## HW 2/ReutersC50/C50test/MichaelConnor             36
## HW 2/ReutersC50/C50test/MureDickie                24
## HW 2/ReutersC50/C50test/NickLouth                 37
## HW 2/ReutersC50/C50test/PatriciaCommins           35
## HW 2/ReutersC50/C50test/PeterHumphrey             36
## HW 2/ReutersC50/C50test/PierreTran                37
## HW 2/ReutersC50/C50test/RobinSidel                42
## HW 2/ReutersC50/C50test/RogerFillion              47
## HW 2/ReutersC50/C50test/SamuelPerry               29
## HW 2/ReutersC50/C50test/SarahDavison              28
```

```
## HW 2/ReutersC50/C50test/ScottHillis                19
## HW 2/ReutersC50/C50test/SimonCowell                33
## HW 2/ReutersC50/C50test/TanEeLyn                   25
## HW 2/ReutersC50/C50test/TheresePoletti             26
## HW 2/ReutersC50/C50test/TimFarrand                 36
## HW 2/ReutersC50/C50test/ToddNissen                 38
## HW 2/ReutersC50/C50test/WilliamKazer               19
```

```
sum(rf_correct_by_author)/2500
```

```
## [1] 0.7192
```

Our random forest produces a higher accuracy of around 72% and is therefore my preferred model. Authors like Aaron Pressman and Jim Gilchrist are predicted well, meaning that perhaps they have strong characteristic writing styles specific to them. Other authors like Jan Lopatka and William Kazer are not well predicted, and thus potentially have less of a distinguishable vocabularly in their writing.

Practice with Association Rule Mining

```
library(arules)
```

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

```
##
## Attaching package: 'arules'
```

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

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

```
library(arulesViz)
```

```
## Loading required package: grid
```

```
groceries = read.transactions("~/Documents/UT/Summer Classes/Intro to
Predictive Modeling/Part 2/HW 2/groceries.txt", format = 'basket', sep = ',')
summary(groceries)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
```

```
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55
##   16   17   18   19   20   21   22   23   24   26   27   28   29   32
##   46   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
groceries_rules = apriori(groceries, parameter=list(support=.005,
confidence=.5, maxlen=8))
```

```
## 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
##       8  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 [0.00s].
## writing ... [120 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
inspect(groceries_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
```

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

```
##     lhs                     rhs                    support confidence
lift count
## [1] {onions,
##      root vegetables}    => {other vegetables} 0.005693950  0.6021505
3.112008    56
## [2] {curd,
##      tropical fruit}     => {yogurt}           0.005287239  0.5148515
3.690645    52
## [3] {pip fruit,
##      whipped/sour cream} => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [4] {citrus fruit,
##      root vegetables}    => {other vegetables} 0.010371124  0.5862069
3.029608   102
## [5] {root vegetables,
##      tropical fruit}     => {other vegetables} 0.012302999  0.5845411
3.020999   121
## [6] {pip fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005490595  0.6136364
3.171368    54
## [7] {citrus fruit,
##      root vegetables,
```

```
##      whole milk}         => {other vegetables} 0.005795628  0.6333333
3.273165    57
## [8] {root vegetables,
##      tropical fruit,
##      whole milk}         => {other vegetables} 0.007015760  0.5847458
3.022057    69
```

```
inspect(subset(groceries_rules, subset=confidence > 0.6))
```

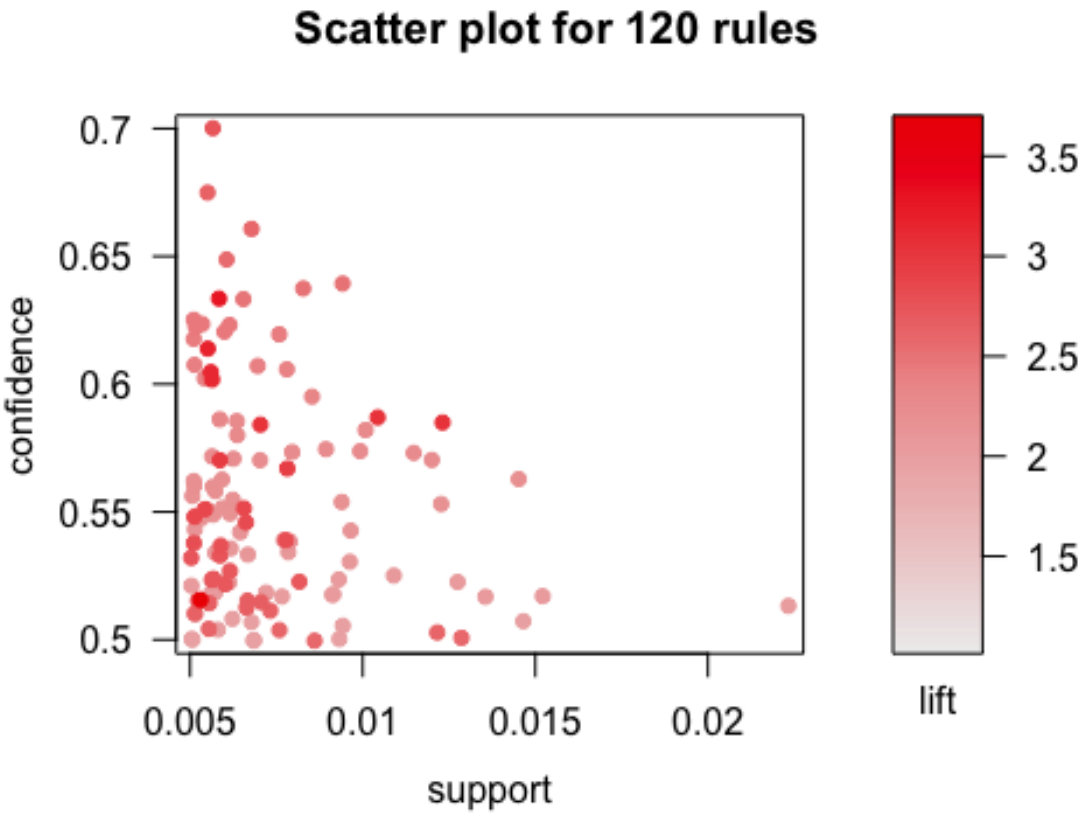
```
##      lhs                         rhs                    support confidence
lift count
## [1]  {onions,
##       root vegetables}       => {other vegetables} 0.005693950  0.6021505
3.112008    56
## [2]  {curd,
##       tropical fruit}        => {whole milk}       0.006507372  0.6336634
2.479936    64
## [3]  {domestic eggs,
##       margarine}             => {whole milk}       0.005185562  0.6219512
2.434099    51
## [4]  {butter,
##       domestic eggs}         => {whole milk}       0.005998983  0.6210526
2.430582    59
## [5]  {butter,
##       whipped/sour cream}    => {whole milk}       0.006710727  0.6600000
2.583008    66
## [6]  {bottled water,
##       butter}                => {whole milk}       0.005388917  0.6022727
2.357084    53
## [7]  {butter,
##       tropical fruit}        => {whole milk}       0.006202339  0.6224490
2.436047    61
## [8]  {butter,
##       root vegetables}       => {whole milk}       0.008235892  0.6377953
2.496107    81
## [9]  {butter,
##       yogurt}                => {whole milk}       0.009354347  0.6388889
2.500387    92
## [10] {domestic eggs,
##       pip fruit}             => {whole milk}       0.005388917  0.6235294
2.440275    53
## [11] {domestic eggs,
##       tropical fruit}        => {whole milk}       0.006914082  0.6071429
2.376144    68
## [12] {pip fruit,
##       whipped/sour cream}    => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [13] {pip fruit,
##       whipped/sour cream}    => {whole milk}       0.005998983  0.6483516
2.537421    59
```

```
## [14] {fruit/vegetable juice,
##       other vegetables,
##       yogurt}                => {whole milk}       0.005083884  0.6172840
2.415833    50
## [15] {other vegetables,
##       root vegetables,
##       whipped/sour cream}    => {whole milk}       0.005185562  0.6071429
2.376144    51
## [16] {other vegetables,
##       pip fruit,
##       root vegetables}       => {whole milk}       0.005490595  0.6750000
2.641713    54
## [17] {pip fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005490595  0.6136364
3.171368    54
## [18] {other vegetables,
##       pip fruit,
##       yogurt}                => {whole milk}       0.005083884  0.6250000
2.446031    50
## [19] {citrus fruit,
##       root vegetables,
##       whole milk}            => {other vegetables} 0.005795628  0.6333333
3.273165    57
## [20] {root vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.005693950  0.7000000
2.739554    56
## [21] {other vegetables,
##       tropical fruit,
##       yogurt}                => {whole milk}       0.007625826  0.6198347
2.425816    75
## [22] {other vegetables,
##       root vegetables,
##       yogurt}                => {whole milk}       0.007829181  0.6062992
2.372842    77
```

```
inspect(subset(groceries_rules, subset=lift > 3 & confidence > 0.6))
```

```
##     lhs                     rhs                    support confidence
lift count
## [1] {onions,
##      root vegetables}    => {other vegetables} 0.005693950  0.6021505
3.112008    56
## [2] {pip fruit,
##      whipped/sour cream} => {other vegetables} 0.005592272  0.6043956
3.123610    55
## [3] {pip fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005490595  0.6136364
```

```
3.171368    54
## [4] {citrus fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005795628  0.6333333
3.273165    57

plot(groceries_rules)

## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```



A lot of the item sets are very similar products grouped together, like citrus fruit and tropical fruit or root vegetables and other vegetables. The highest lift values were primarily sets of items that inform the purchase of 'other vegetables.' We chose a threshhold for lift of 3, because most of the lift values ranged from 1-3, so the values with lift >3 showed us the highly informative baskets. Most of the confidence values ranged from 0.5-0.7 so we chose a threshhold of confidence > 0.6. The rules we found made sense, and primarily tell us about what groups of items tell us about the liklihood of buying whole milk and vegetables.