Exercises 2

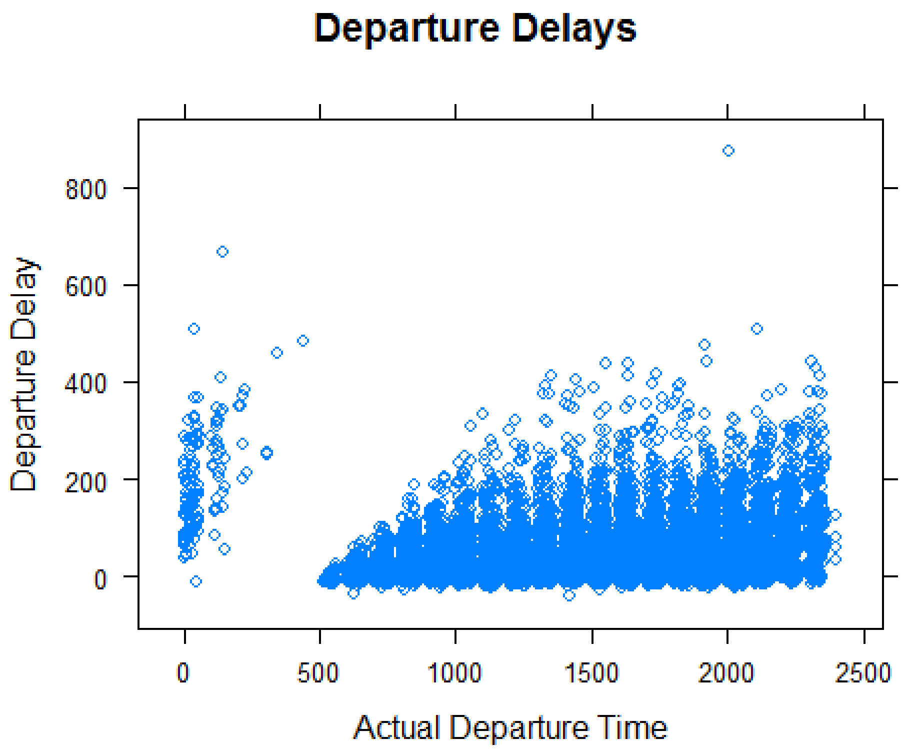
Erik Honore

August 11, 2016

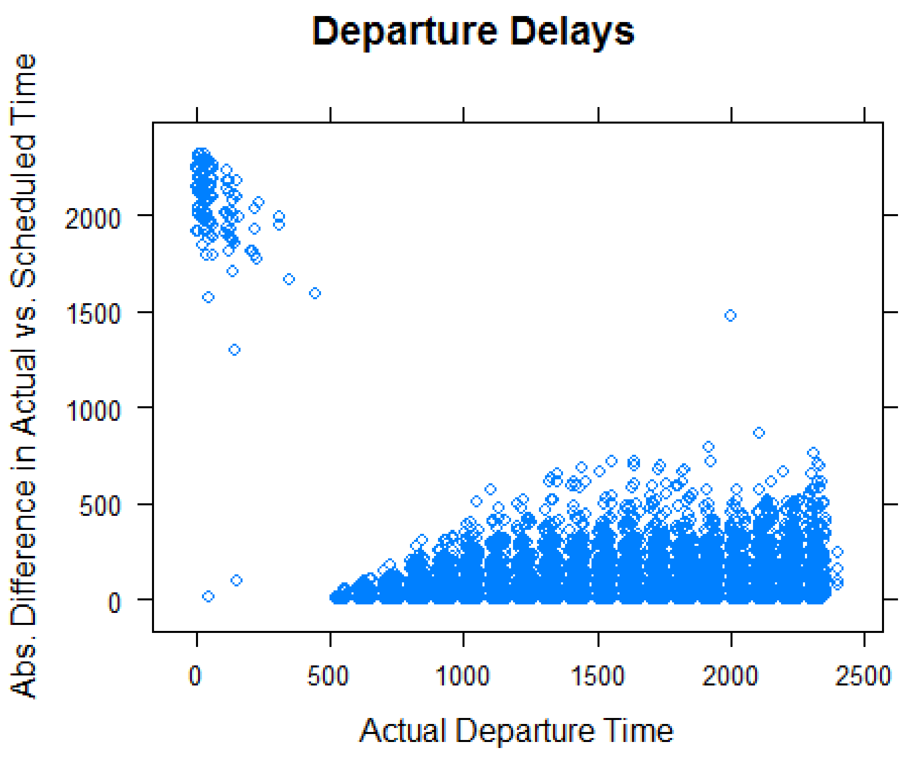
Flights at ABIA

Plotting Analysis

```
#What is the best time of day to fly to minimize delays?

xyplot(abia$DepDelay~abia$DepTime, xlab='Actual Departure Time', ylab=
'Departure Delay', main="Departure Delays")
```



```
abs_diff_schedule_and_actual = abs(abia$DepTime-abia$CRSDepTime)

xyplot(abs_diff_schedule_and_actual~abia$DepTime, xlab= 'Actual Departure
Time', ylab= 'Abs. Difference in Actual vs. Scheduled Time', main="Departure
Delays")
```

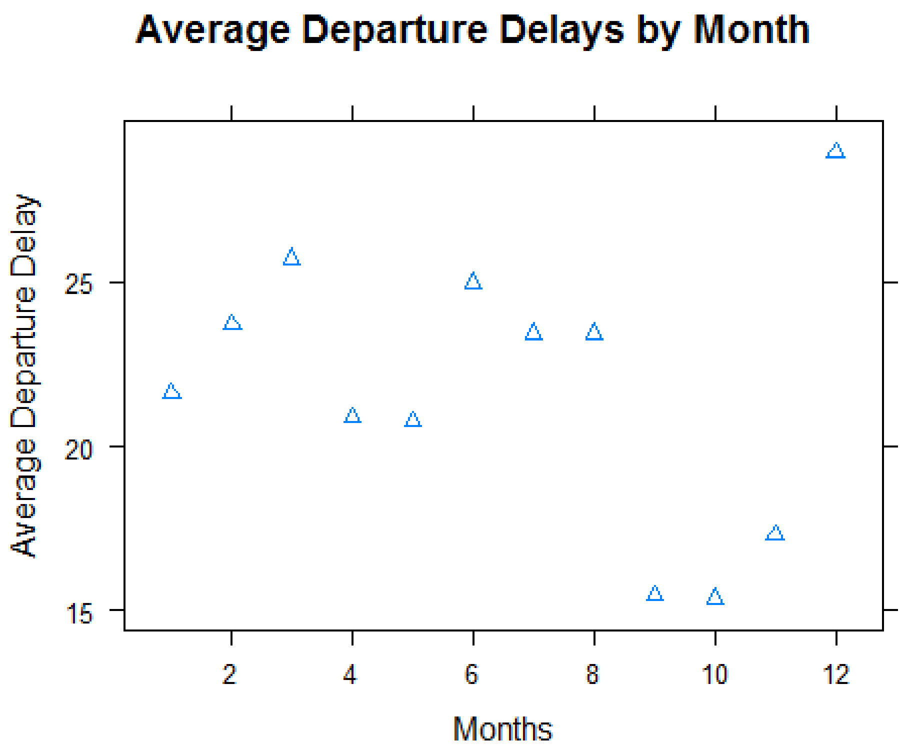


In terms of departing delays, which are typically what most people think about when dealing with air travel, we can see from two of the three graphs here that they best time to try to fly in order to avoid delays is between 5am and 10am. This is likely due to the fact that there are simply less flights during these hours, so delays are not as big of an issue as they are as the day goes on.

The second plot from above shows basically a similar pattern. The y is the absolute difference between scheduled departing time and the actual. This is useful in showing that as the day goes on, the range of this difference, which could be thought of as a simple measure of variability, increases. We can see that the spread of points for the mid to later hours in the day is far larger than in the early morning hours.

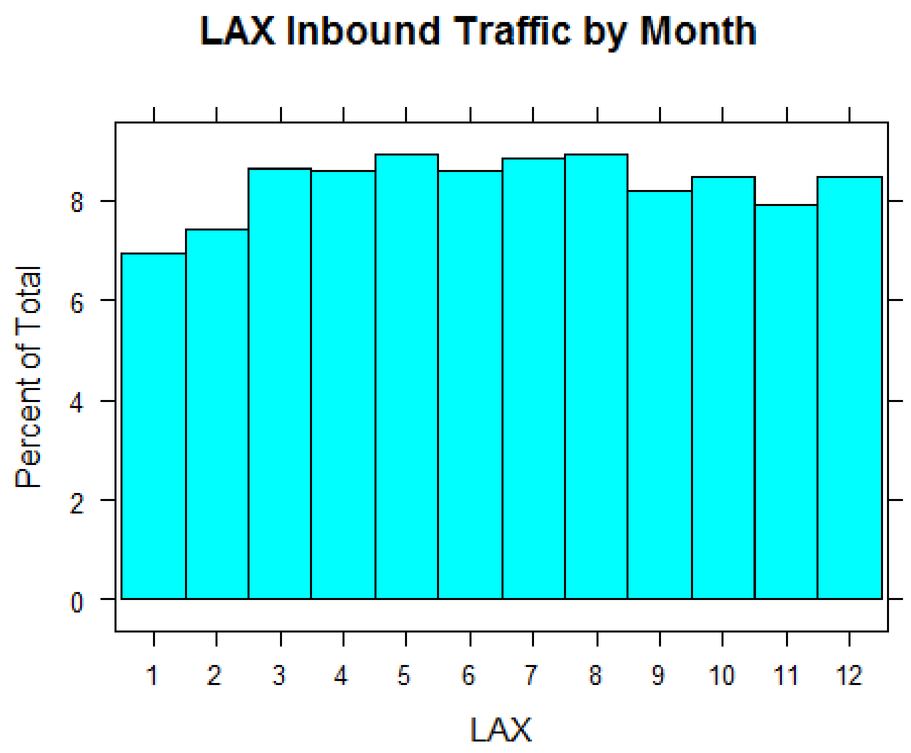
```
#What is the best time of year to fly to minimize delays?
month_vect = c(1:12)
abia_copy = abia
avg_vect = c()

for (i in month_vect){

  abia1 = abia_copy[abia_copy[,2] == toString(i),16] #get each month
individually
  abia1 = na.omit(abia1) #drop na's
  abia1 = abia1[abia1 >= 0] #drop the ahead-of-schedule negative values
  k = mean(abia1)
  avg_vect = c(avg_vect,k) #create ordered vector with all of the means
stored up
```

```r
}

xyplot(avg_vect~month_vect, xlab= 'Months',ylab='Average Departure
Delay',main='Average Departure Delays by Month', pch=2)
```

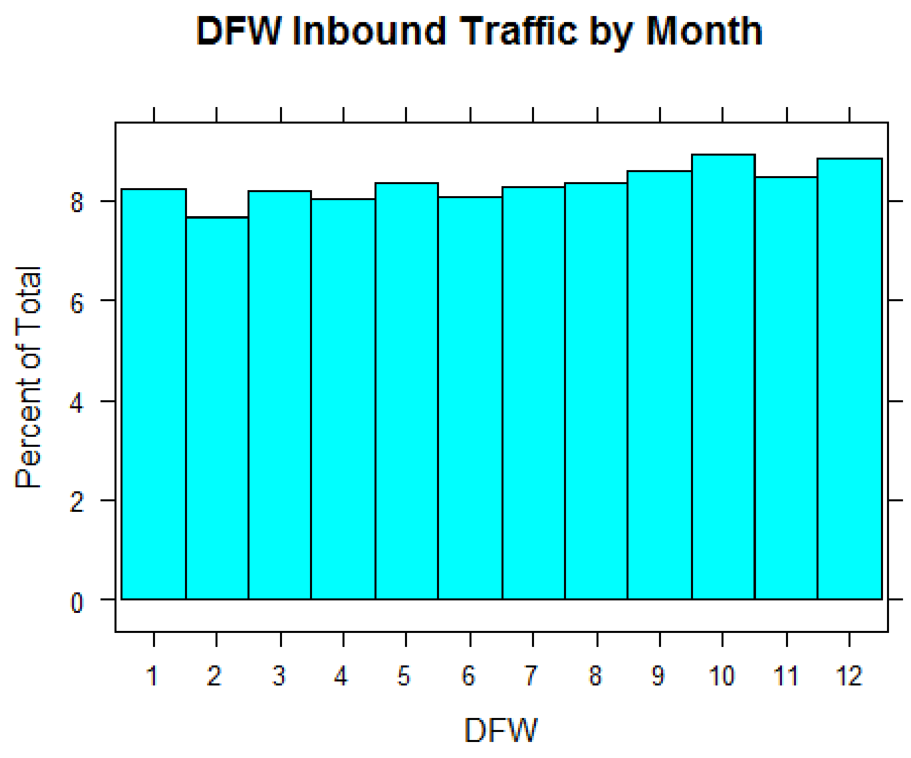


From this plot, we can see that the best time to fly in order to have the smallest delays would probably be the months of September through November. During those months, there isn't too much family vacationing happening, so this basic plot of average departure delays on month makes sense.

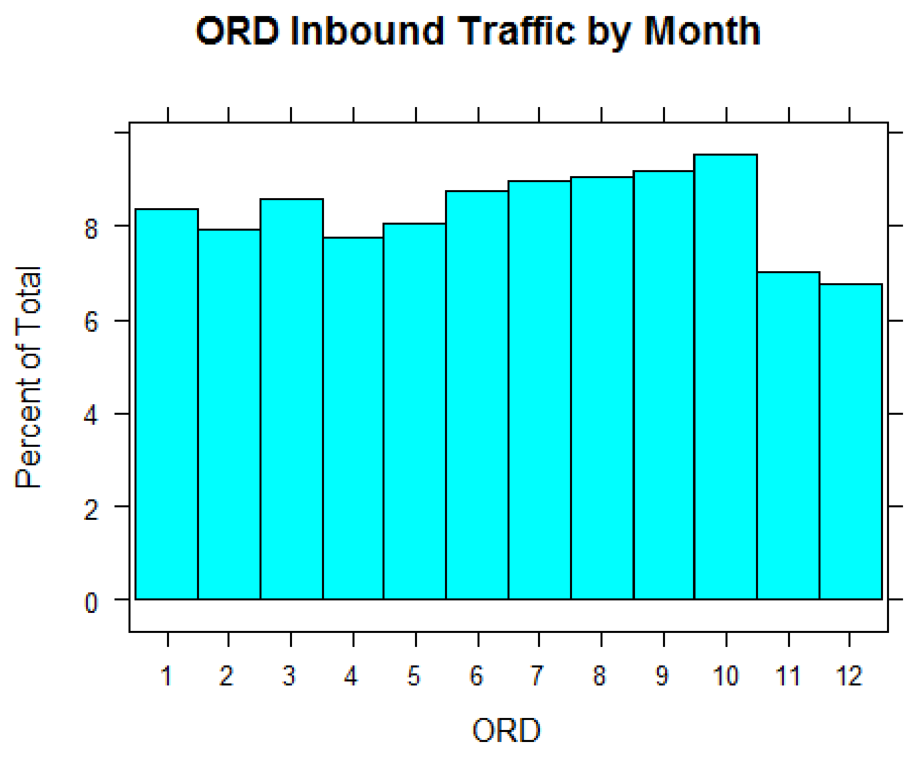
```r
#How do patterns of flights to different destinations or parts of the country
change over the course of the year?

#LAX
lax_copy = abia_copy
lax_copy = lax_copy[lax_copy[,18] == 'LAX',]
lax_copy = lax_copy[,c(2,18)]
histogram(~lax_copy$Month, xlab = "LAX", main='LAX Inbound Traffic by Month')
```



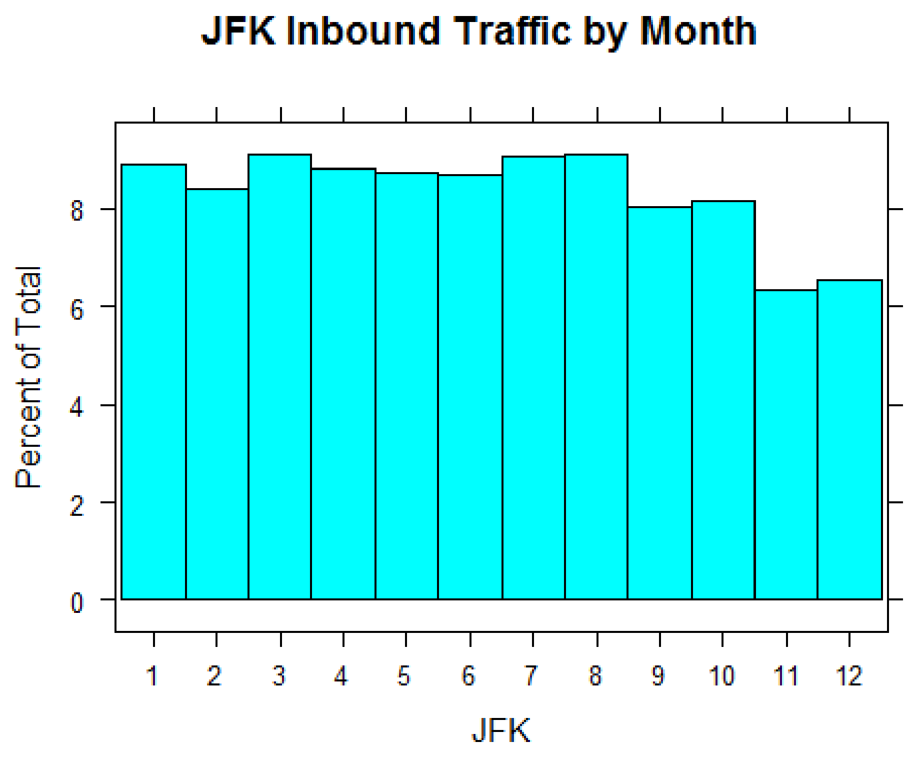
```
# DFW
dfw_copy = abia_copy
dfw_copy = dfw_copy[dfw_copy[,18] == 'DFW',]
dfw_copy = dfw_copy[,c(2,18)]
histogram(~dfw_copy$Month, xlab = 'DFW',main='DFW Inbound Traffic by Month')
```



```
#ORD
ord_copy = abia_copy
ord_copy = ord_copy[ord_copy[,18] == 'ORD',]
ord_copy = ord_copy[,c(2,18)]
histogram(~ord_copy$Month, xlab = 'ORD',main='ORD Inbound Traffic by Month')
```



```
#jfk
jfk_copy = abia_copy
jfk_copy = jfk_copy[jfk_copy[,18] == 'JFK',]
jfk_copy = jfk_copy[,c(2,18)]
histogram(~jfk_copy$Month, xlab = 'JFK',main='JFK Inbound Traffic by Month')
```

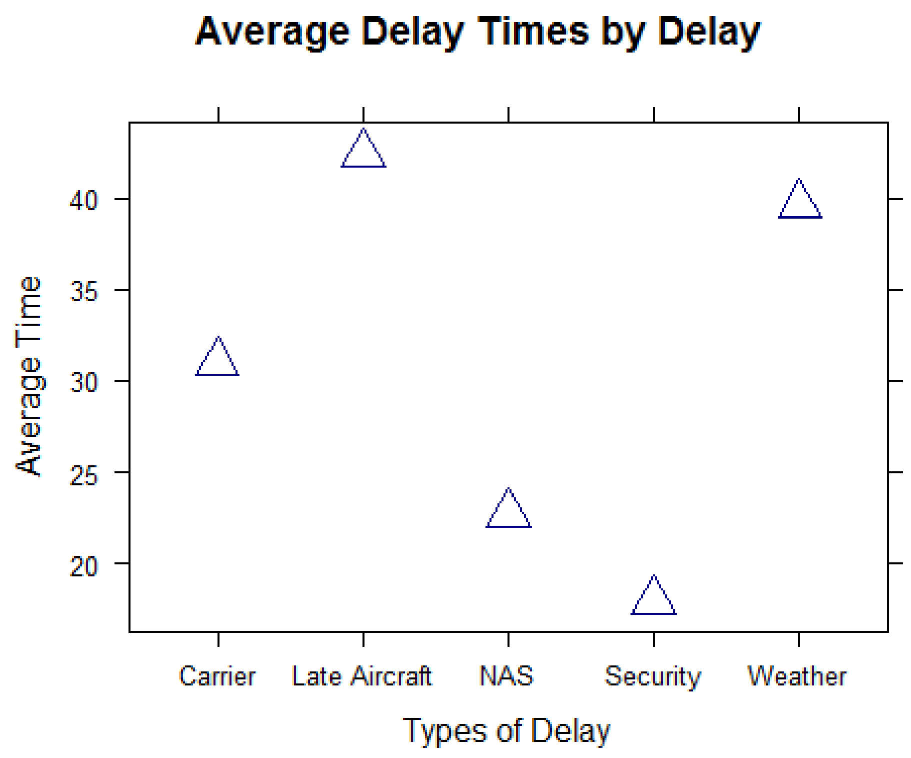


To look at the question of flights to different destinations, I reduced my scope to look at only a few busy airports in various regions in the US. I also only looked at the destination column for those airports as well. With LAX, we can see that it has the highest traffic in terms of flights in during the middle of the year. It has its lowest traffic during the first months of the year. DFW is somewhat opposite in that it has higher traffic in the beginning and end of the year, with slight reduction of traffic in the middle of the year. ORD has a gradual increase in traffic until October where traffic suddenly drops. This is likely due to weather and climate that make Chicago less friendly as a vacation destination. A similar trend is scene with NY's JFK. Although its decrease in flight traffic starts earlier towards September.

For these next two plots, I pulled the delay type data out of the data frame, removed all the NAs and 0 values thus only leaving instances where the type of delay was actually realized.

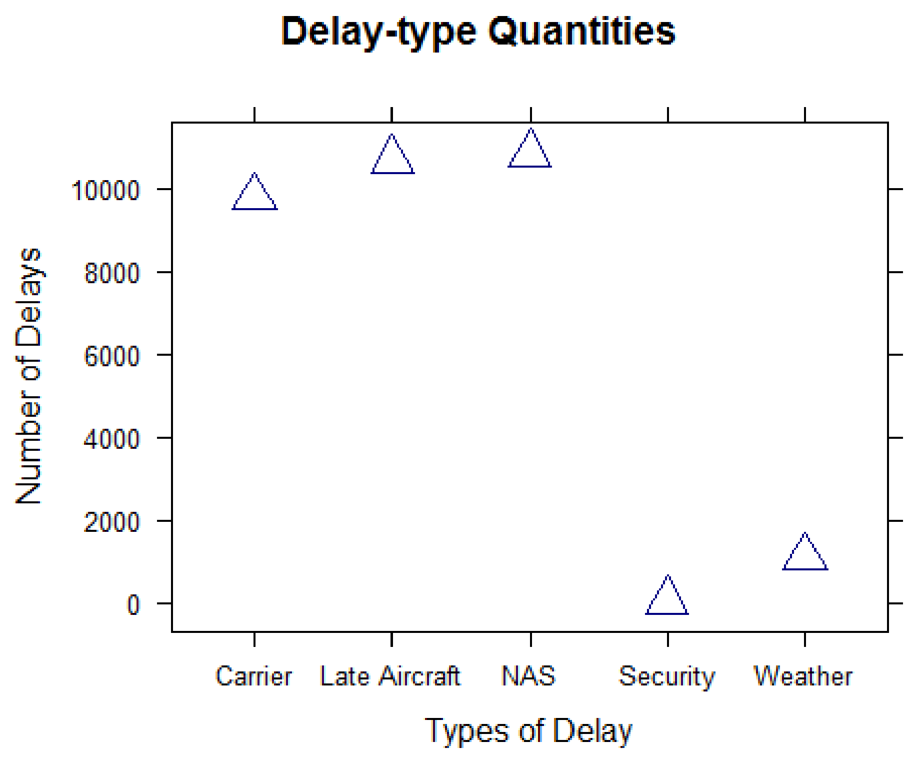
```
#What is the histogram for each type of delay. See how 1 delay may be more
lopsided than another etc.

#mask out for only the delay types, and then remove na's, then do two
analyses: one with just the counts of each delay, and 1 with the avg lengths
of each delay

delay_copy = abia_copy

delay_copy = delay_copy[,25:29]
delay_copy = na.omit(delay_copy)
#fix(delay_copy)
```

```r
#plot(delay_copy)

#find avg delay length for each type

#carrier delay
carrier_mat = delay_copy[,1]
carrier_mat = carrier_mat[carrier_mat != 0]
carrier_mean = mean(carrier_mat)
num_carrdelay = length(carrier_mat)

#weather delay
weather_mat = delay_copy[,2]
weather_mat = weather_mat[weather_mat != 0]
weather_mean = mean(weather_mat)
num_weathdelay = length(weather_mat)

#nas delay
nas_mat = delay_copy[,3]
nas_mat = nas_mat[nas_mat != 0]
nas_mean = mean(nas_mat)
num_nasdelay = length(nas_mat)

#security delay
sec_mat = delay_copy[,4]
sec_mat = sec_mat[sec_mat != 0]
sec_mean = mean(sec_mat)
num_secdelay = length(sec_mat)

#late aircraft delay
la_mat = delay_copy[,5]
la_mat = la_mat[la_mat != 0]
la_mean = mean(la_mat)
num_ladelay = length(la_mat)


delaytypes = c("Carrier", "Weather","NAS", "Security", "Late Aircraft")
delay_avgs = c(carrier_mean,weather_mean, nas_mean, sec_mean,la_mean)
delay_nums =
c(num_carrdelay,num_weathdelay,num_nasdelay,num_secdelay,num_ladelay)
delaymeans_frame = data.frame(delaytypes, delay_avgs, delay_nums)

xyplot(delaymeans_frame$delay_avgs~delaymeans_frame$delaytypes, xlab= 'Types
of Delay', ylab= 'Average Time', main= 'Average Delay Times by Delay',
col='navy', pch=2, cex=2)
```



As we can see from the first graph here, late aircraft force the highest average delay times. This make sense if we consider the fact that late deparures from elsewhere will most likely cause a plane to be late for its next scheduled departure. Fortunately, security issues account for the smallest average delay type. In summary, weather and late craft are naturally the biggest causes of delay.

```
xyplot(delaymeans_frame$delay_nums~delaymeans_frame$delaytypes, xlab= 'Types of Delay', ylab= 'Number of Delays', main= 'Delay-type Quantities',col='navy', pch=2, cex=2)
```



This graph shows that security and weather happen to be the least numerous types of delays for 2008. This is interesting because although weather delays occurred in small number, they caused the 2nd highest average delays times for the year. Typically, this means that severe weather happens less frequently, but is much more serious in terms of flight. The NAS category represents the sort of miscellaneous delays of airport logistical issues, congestion, etc.

Author Attribution

Set up the Data

```
library(tm)

## Loading required package: NLP

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

for(author in author_dirs) { #for each list of directories make a list of
files, and give a label to each file in directory. the label is the author
name.
    author_name = substring(author, first=29)
    files_to_add = Sys.glob(paste0(author, '/*.txt'))
    file_list = append(file_list, files_to_add)
    labels = append(labels, rep(author_name, length(files_to_add)))
}

#file_list contains all the files for all the authors now

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

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

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


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

#probably need to do stemming here
library(SnowballC)
my_corpus <- tm_map(my_corpus, stemDocument)

DTM = DocumentTermMatrix(my_corpus)
#DTM

#inspect(DTM[1:10,1:20])
DTM = removeSparseTerms(DTM, 0.975) #a word had to occur in atleast 2 in 100
docs to make cutoff.Can't generalize on words that occurred only 1 time.
#DTM

#docs 1-2500 are train, 2501-5000 are test
#get it to df shape
X = as.data.frame(data.matrix(DTM))
#did this instead of x = as.matrix(DTM)
#now add the classes (the authors) as a final column somehow
names = row.names(X)
```

```r
cat_col = c()

for (i in names){ #loop that extract author names from the document titles

  str = i

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

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

}

X[,"Category of Author"] <- cat_col #put the author names as a category
column in the end of df
```

Model - KNN

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

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

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

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

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

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

#knn.pred

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

#conf.mat

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

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

Accuracy: 77%

The KNN model does performs well in this setting. There are several authors that the model is falsely classifying as a different author. Those include: Alan Crosby, who was predicted to be John Mastrini 13 times. Alexander Smith was predicted to be Joe Ortiz 11 times. Darren Schuettler was predicted to be Heather Scoffield 20 times. There are quite a few instances where the predicted article author is off by quite a lot, and where many of those errors were with respect to exactly 1 author. My guess is that when these sorts of large errors occurred, it was between two authors that were perhaps writing about similar things with similar word usage.

Model - Naive Bayes

```
library(e1071)

model_nb = naiveBayes(as.factor(`Category of Author`)~.,data= train)
#X,subset=train_sample)

nb_pred = predict(model_nb, newdata = test[,-1277])

conf.mat1 <- table("Predictions" = nb_pred, Actual = cat_test) #confusion
matrix

accuracy1 <- sum(diag(conf.mat1))/length(test) * 100
accuracy1

## [1] 45.96711
```

Accuracy: 45%

Overall the NB model performed pretty well. It was obviously not as effective as the KNN most likely due to the independence assumption. The words used in these documents in the context of classification are definitely dependent or related. NB performing slightly worse in this context is really no surprise.

Problem 3 Market Basket

```
#detach(package:maxent,unload=TRUE)
detach(package:tm,unload=TRUE)

library(arules)

## Loading required package: Matrix

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

```
library(arulesViz)
```

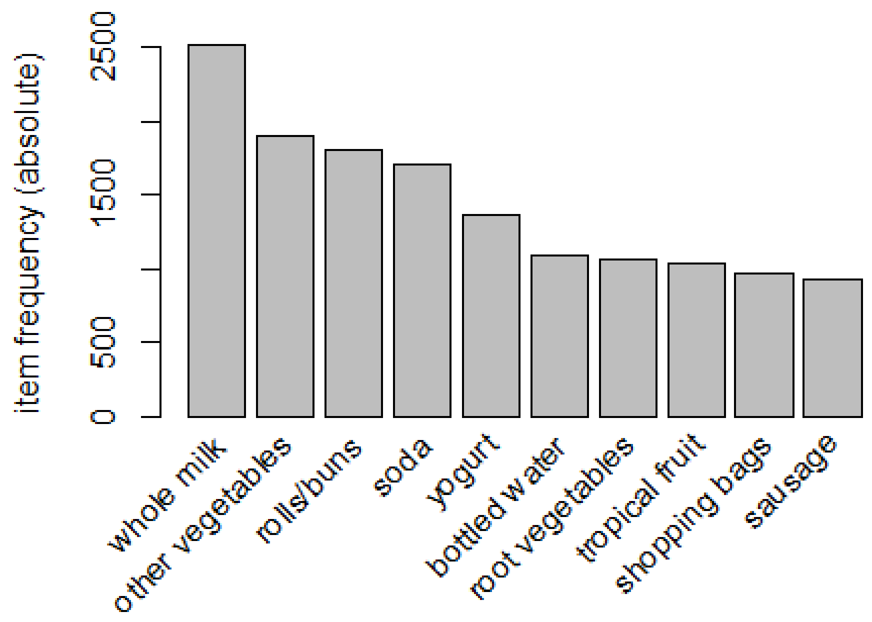
```
## Loading required package: grid
```

```
groceries_raw = read.transactions('groceries.txt', sep=",")
```

```
frequentItems <- eclat (groceries_raw, parameter = list(supp = 0.07, maxlen =
15))
```

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target   ext
##     FALSE    0.07      1     15 frequent itemsets FALSE
##
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 688
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating sparse bit matrix ... [18 row(s), 9835 column(s)] done [0.00s].
## writing  ... [19 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
itemFrequencyPlot (groceries_raw,topN=10,type="absolute")
```



```
foodrules <- apriori(groceries_raw, parameter=list(support=.002,
confidence=.7, maxlen=3))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.7    0.1    1 none FALSE            TRUE   0.002      1      3
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 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 [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [22 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

#foodrules
```

```
inspect(foodrules) #increased the support slightly to narrow down the rules,
everything in

##    lhs                       rhs                    support confidence
lift
## 1  {specialty cheese,
##     yogurt}               => {whole milk}       0.002033554  0.7142857
2.795464
## 2  {rice,
##     root vegetables}      => {other vegetables} 0.002236909  0.7096774
3.667723
## 3  {rice,
##     root vegetables}      => {whole milk}       0.002440264  0.7741935
3.029922
## 4  {citrus fruit,
##     herbs}                => {other vegetables} 0.002135231  0.7241379
3.742457
## 5  {herbs,
##     tropical fruit}       => {whole milk}       0.002338587  0.8214286
3.214783
## 6  {herbs,
##     rolls/buns}           => {whole milk}       0.002440264  0.8000000
3.130919
## 7  {baking powder,
##     root vegetables}      => {other vegetables} 0.002541942  0.7142857
3.691540
## 8  {baking powder,
##     yogurt}               => {whole milk}       0.003253686  0.7111111
2.783039
## 9  {butter,
##     soft cheese}          => {whole milk}       0.002033554  0.7407407
2.898999
## 10 {soft cheese,
##     whipped/sour cream} => {other vegetables} 0.002236909  0.7333333
3.789981
## 11 {root vegetables,
##     soft cheese}          => {other vegetables} 0.002440264  0.7272727
3.758659
## 12 {butter milk,
##     dessert}              => {whole milk}       0.002033554  0.7142857
2.795464
## 13 {butter milk,
##     whipped/sour cream} => {whole milk}       0.002948653  0.7631579
2.986732
## 14 {sliced cheese,
##     whipped/sour cream} => {whole milk}       0.002745297  0.7105263
2.780751
## 15 {butter,
##     onions}               => {whole milk}       0.003050330  0.7500000
2.935237
```

```
## 16 {curd,
##     hamburger meat}      => {whole milk}       0.002541942  0.8064516
3.156169
## 17 {cream cheese,
##     sugar}               => {whole milk}       0.002033554  0.7407407
2.898999
## 18 {domestic eggs,
##     sugar}               => {whole milk}       0.003558719  0.7142857
2.795464
## 19 {butter,
##     coffee}              => {whole milk}       0.003355363  0.7021277
2.747881
## 20 {butter,
##     curd}                => {whole milk}       0.004880529  0.7164179
2.803808
## 21 {curd,
##     domestic eggs}       => {whole milk}       0.004778851  0.7343750
2.874086
## 22 {butter,
##     pork}                => {whole milk}       0.003863752  0.7037037
2.754049
```

```
inspect(subset(foodrules, subset=lift > 2))
```

```
##    lhs                     rhs                     support confidence
lift
## 1  {specialty cheese,
##     yogurt}              => {whole milk}       0.002033554  0.7142857
2.795464
## 2  {rice,
##     root vegetables}     => {other vegetables} 0.002236909  0.7096774
3.667723
## 3  {rice,
##     root vegetables}     => {whole milk}       0.002440264  0.7741935
3.029922
## 4  {citrus fruit,
##     herbs}               => {other vegetables} 0.002135231  0.7241379
3.742457
## 5  {herbs,
##     tropical fruit}      => {whole milk}       0.002338587  0.8214286
3.214783
## 6  {herbs,
##     rolls/buns}          => {whole milk}       0.002440264  0.8000000
3.130919
## 7  {baking powder,
##     root vegetables}     => {other vegetables} 0.002541942  0.7142857
3.691540
## 8  {baking powder,
##     yogurt}              => {whole milk}       0.003253686  0.7111111
2.783039
```

```
## 9  {butter,
##     soft cheese}         => {whole milk}       0.002033554  0.7407407
2.898999
## 10 {soft cheese,
##     whipped/sour cream} => {other vegetables} 0.002236909  0.7333333
3.789981
## 11 {root vegetables,
##     soft cheese}         => {other vegetables} 0.002440264  0.7272727
3.758659
## 12 {butter milk,
##     dessert}             => {whole milk}       0.002033554  0.7142857
2.795464
## 13 {butter milk,
##     whipped/sour cream} => {whole milk}       0.002948653  0.7631579
2.986732
## 14 {sliced cheese,
##     whipped/sour cream} => {whole milk}       0.002745297  0.7105263
2.780751
## 15 {butter,
##     onions}              => {whole milk}       0.003050330  0.7500000
2.935237
## 16 {curd,
##     hamburger meat}      => {whole milk}       0.002541942  0.8064516
3.156169
## 17 {cream cheese,
##     sugar}               => {whole milk}       0.002033554  0.7407407
2.898999
## 18 {domestic eggs,
##     sugar}               => {whole milk}       0.003558719  0.7142857
2.795464
## 19 {butter,
##     coffee}              => {whole milk}       0.003355363  0.7021277
2.747881
## 20 {butter,
##     curd}                => {whole milk}       0.004880529  0.7164179
2.803808
## 21 {curd,
##     domestic eggs}       => {whole milk}       0.004778851  0.7343750
2.874086
## 22 {butter,
##     pork}                => {whole milk}       0.003863752  0.7037037
2.754049
```

```r
inspect(subset(foodrules, subset=lift > 3))
```

```
##   lhs                        rhs                     support confidence
lift
## 1 {rice,
##    root vegetables}    => {other vegetables} 0.002236909  0.7096774
3.667723
```

```
## 2 {rice,
##    root vegetables}    => {whole milk}       0.002440264  0.7741935
3.029922
## 3 {citrus fruit,
##    herbs}              => {other vegetables} 0.002135231  0.7241379
3.742457
## 4 {herbs,
##    tropical fruit}     => {whole milk}       0.002338587  0.8214286
3.214783
## 5 {herbs,
##    rolls/buns}         => {whole milk}       0.002440264  0.8000000
3.130919
## 6 {baking powder,
##    root vegetables}    => {other vegetables} 0.002541942  0.7142857
3.691540
## 7 {soft cheese,
##    whipped/sour cream} => {other vegetables} 0.002236909  0.7333333
3.789981
## 8 {root vegetables,
##    soft cheese}        => {other vegetables} 0.002440264  0.7272727
3.758659
## 9 {curd,
##    hamburger meat}     => {whole milk}       0.002541942  0.8064516
3.156169

#let's look closer at whole milk

foodrules <- apriori(groceries_raw, parameter=list(support=.002,
confidence=.7, maxlen=4), appearance = list (default="lhs",rhs="whole milk"),
control = list (verbose=F))

inspect(foodrules)

##    lhs                             rhs              support confidence
lift
## 1  {specialty cheese,
##     yogurt}                     => {whole milk} 0.002033554  0.7142857
2.795464
## 2  {rice,
##     root vegetables}            => {whole milk} 0.002440264  0.7741935
3.029922
## 3  {herbs,
##     tropical fruit}             => {whole milk} 0.002338587  0.8214286
3.214783
## 4  {herbs,
##     rolls/buns}                 => {whole milk} 0.002440264  0.8000000
3.130919
## 5  {baking powder,
##     yogurt}                     => {whole milk} 0.003253686  0.7111111
2.783039
```

```
## 6  {butter,
##     soft cheese}               => {whole milk} 0.002033554  0.7407407
2.898999
## 7  {butter milk,
##     dessert}                   => {whole milk} 0.002033554  0.7142857
2.795464
## 8  {butter milk,
##     whipped/sour cream}        => {whole milk} 0.002948653  0.7631579
2.986732
## 9  {sliced cheese,
##     whipped/sour cream}        => {whole milk} 0.002745297  0.7105263
2.780751
## 10 {butter,
##     onions}                    => {whole milk} 0.003050330  0.7500000
2.935237
## 11 {curd,
##     hamburger meat}            => {whole milk} 0.002541942  0.8064516
3.156169
## 12 {cream cheese,
##     sugar}                     => {whole milk} 0.002033554  0.7407407
2.898999
## 13 {domestic eggs,
##     sugar}                     => {whole milk} 0.003558719  0.7142857
2.795464
## 14 {butter,
##     coffee}                    => {whole milk} 0.003355363  0.7021277
2.747881
## 15 {butter,
##     curd}                      => {whole milk} 0.004880529  0.7164179
2.803808
## 16 {curd,
##     domestic eggs}             => {whole milk} 0.004778851  0.7343750
2.874086
## 17 {butter,
##     pork}                      => {whole milk} 0.003863752  0.7037037
2.754049
## 18 {hard cheese,
##     other vegetables,
##     yogurt}                    => {whole milk} 0.002033554  0.7142857
2.795464
## 19 {hygiene articles,
##     other vegetables,
##     yogurt}                    => {whole milk} 0.002135231  0.7500000
2.935237
## 20 {long life bakery product,
##     other vegetables,
##     yogurt}                    => {whole milk} 0.002643620  0.7222222
2.826524
## 21 {chicken,
##     domestic eggs,
##     other vegetables}          => {whole milk} 0.002440264  0.7272727
2.846290
## 22 {tropical fruit,
##     white bread,
##     yogurt}                    => {whole milk} 0.002033554  0.7142857
2.795464
## 23 {coffee,
##     other vegetables,
##     whipped/sour cream}        => {whole milk} 0.002033554  0.7142857
2.795464
## 24 {frozen vegetables,
##     root vegetables,
##     tropical fruit}            => {whole milk} 0.002338587  0.7666667
3.000464
## 25 {beef,
##     domestic eggs,
##     other vegetables}          => {whole milk} 0.002541942  0.7575758
2.964886
## 26 {beef,
##     rolls/buns,
##     tropical fruit}            => {whole milk} 0.002135231  0.7777778
3.043949
## 27 {butter,
##     curd,
##     yogurt}                    => {whole milk} 0.002338587  0.7931034
3.103929
## 28 {butter,
##     curd,
##     other vegetables}          => {whole milk} 0.002236909  0.7333333
2.870009
## 29 {curd,
##     domestic eggs,
##     yogurt}                    => {whole milk} 0.002033554  0.7407407
2.898999
## 30 {curd,
##     domestic eggs,
##     other vegetables}          => {whole milk} 0.002846975  0.8235294
3.223005
## 31 {curd,
##     tropical fruit,
##     whipped/sour cream}        => {whole milk} 0.002033554  0.7142857
2.795464
## 32 {curd,
##     pastry,
##     yogurt}                    => {whole milk} 0.002338587  0.7187500
2.812935
## 33 {citrus fruit,
##     curd,
##     yogurt}                    => {whole milk} 0.002135231  0.7000000
2.739554
```

```
## 34 {curd,
##     tropical fruit,
##     yogurt}                    => {whole milk} 0.003965430  0.7500000
2.935237
## 35 {curd,
##     rolls/buns,
##     yogurt}                    => {whole milk} 0.002541942  0.7352941
2.877683
## 36 {butter,
##     other vegetables,
##     pork}                      => {whole milk} 0.002236909  0.8461538
3.311549
## 37 {pork,
##     rolls/buns,
##     root vegetables}           => {whole milk} 0.002033554  0.7142857
2.795464
## 38 {frankfurter,
##     root vegetables,
##     tropical fruit}            => {whole milk} 0.002033554  0.7407407
2.898999
## 39 {frankfurter,
##     tropical fruit,
##     yogurt}                    => {whole milk} 0.002135231  0.7241379
2.834022
## 40 {frankfurter,
##     root vegetables,
##     yogurt}                    => {whole milk} 0.002338587  0.7187500
2.812935
## 41 {bottled beer,
##     domestic eggs,
##     other vegetables}          => {whole milk} 0.002033554  0.7692308
3.010499
## 42 {bottled beer,
##     other vegetables,
##     yogurt}                    => {whole milk} 0.002643620  0.7222222
2.826524
## 43 {brown bread,
##     other vegetables,
##     root vegetables}           => {whole milk} 0.003152008  0.7750000
3.033078
## 44 {margarine,
##     rolls/buns,
##     root vegetables}           => {whole milk} 0.002033554  0.7142857
2.795464
## 45 {butter,
##     domestic eggs,
##     root vegetables}           => {whole milk} 0.002440264  0.7500000
2.935237
## 46 {butter,
##     domestic eggs,
##     yogurt}                    => {whole milk} 0.002236909  0.7586207
2.968975
## 47 {butter,
##     tropical fruit,
##     whipped/sour cream}        => {whole milk} 0.002135231  0.7000000
2.739554
## 48 {butter,
##     root vegetables,
##     whipped/sour cream}        => {whole milk} 0.002643620  0.7647059
2.992790
## 49 {bottled water,
##     butter,
##     root vegetables}           => {whole milk} 0.002440264  0.7272727
2.846290
## 50 {bottled water,
##     butter,
##     other vegetables}          => {whole milk} 0.002643620  0.7222222
2.826524
## 51 {butter,
##     tropical fruit,
##     yogurt}                    => {whole milk} 0.003355363  0.7333333
2.870009
## 52 {butter,
##     root vegetables,
##     yogurt}                    => {whole milk} 0.003050330  0.7894737
3.089723
## 53 {domestic eggs,
##     other vegetables,
##     whipped/sour cream}        => {whole milk} 0.003558719  0.7000000
2.739554
## 54 {domestic eggs,
##     root vegetables,
##     tropical fruit}            => {whole milk} 0.002745297  0.7714286
3.019101
## 55 {fruit/vegetable juice,
##     root vegetables,
##     yogurt}                    => {whole milk} 0.002541942  0.7352941
2.877683
## 56 {pip fruit,
##     root vegetables,
##     whipped/sour cream}        => {whole milk} 0.002338587  0.7666667
3.000464
## 57 {citrus fruit,
##     tropical fruit,
##     whipped/sour cream}        => {whole milk} 0.002338587  0.7419355
2.903675
## 58 {citrus fruit,
##     rolls/buns,
##     whipped/sour cream}        => {whole milk} 0.002033554  0.7142857
2.795464
```

```
## 59 {tropical fruit,
##     whipped/sour cream,
##     yogurt}                    => {whole milk} 0.004372140  0.7049180
2.758802
## 60 {root vegetables,
##     sausage,
##     tropical fruit}            => {whole milk} 0.002745297  0.7714286
3.019101
## 61 {root vegetables,
##     tropical fruit,
##     yogurt}                    => {whole milk} 0.005693950  0.7000000
2.739554
```

```r
rules <- sort (foodrules, decreasing=TRUE,by="confidence")

redundant <- which (colSums(is.subset(rules, rules)) > 1) # get redundant
rules in vector

rules <- rules[-redundant]

#inspect(rules)
```

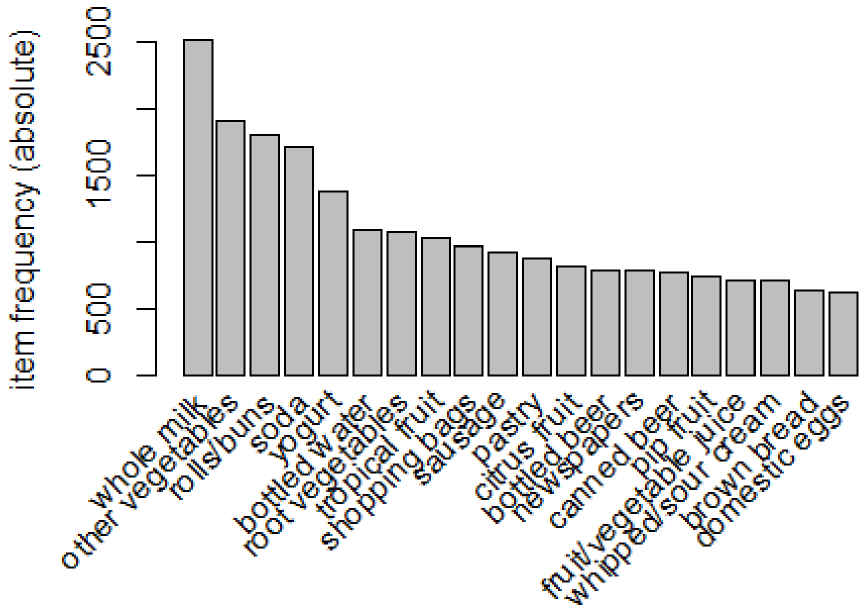
Whole milk is a staple numerous product that must've been present in nearly every transaction. This is evident from the item frequency plot. This is also shown in the inspections I did to see the rules and their parameters.

I chose a very small support but high confidence in order to see the products that are bought less often in total, but often bought together. This would hopefully give me strong rules from the data. I also looked at initially higher values of lift to see the co-occurrence of these types of products.

After testing a variety of support, confidence, and lift values in the inspections, I again found evidence that whole milk is basically present in most purchasing decisions from the given data. So I decided to see just how widespread this theory was my applying an appearance parameter in the rules. When I kept my parameters constant, the amount of rules that led to purchasing whole milk was 61. This is a diverse set of items that usually led to purchasing whole milk.

To switch gears, I'll now look at similar results for a product that wasn't nearly as numerous as whole milk. I decided to look at pastries, which made me change my parameters to get any rules at all.

```r
#Now go to an item that was bought far less than whole milk

itemFrequencyPlot (groceries_raw,topN=20,type="absolute")
```



```
foodrules <- apriori(groceries_raw, parameter=list(support=.001,
confidence=.3, maxlen=43), appearance = list (default="lhs",rhs="pastry"),
control = list (verbose=F))
#foodrules

#inspect(subset(foodrules, subset=lift > 4))

rules <- sort (foodrules, decreasing=TRUE,by="confidence")
inspect(subset(rules, subset = lift >= 4.5))

##   lhs                          rhs           support confidence     lift
## 1 {citrus fruit,
##    rolls/buns,
##    whipped/sour cream,
##    whole milk}             => {pastry} 0.001016777  0.5000000 5.620000
## 2 {soda,
##    waffles,
##    whole milk}             => {pastry} 0.001220132  0.4615385 5.187692
## 3 {grapes,
##    shopping bags}          => {pastry} 0.001016777  0.4166667 4.683333
## 4 {butter,
##    dessert}                => {pastry} 0.001118454  0.4074074 4.579259
```

From the sorted and narrowed rules for pastries, we can see at least one telling discovery. Butter and Dessert were often purchased along with pastries, which makes quite a bit of sense if you consider baking and baked goods. The other high lift baskets were perhaps more logically random when it came to pastries.