Assignment 2

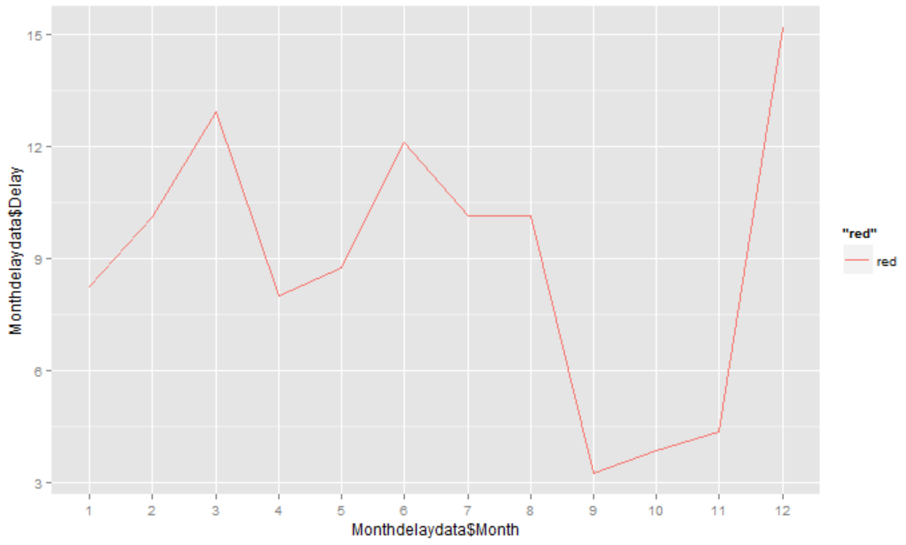
Shrihari Hudli

August 18, 2015

ANSWER 1. FLIGHTS AT ABIA

We consider two interesting questions that concern most air travelers. What is the best time of the year, i.e. the month, to avoid delays? What is the best time of the day to avoid delays? There will be two plots that emerge. First we find out what is the best month of the year to fly to avoid delays. Also, we consider only Departure Delays for this exercise.

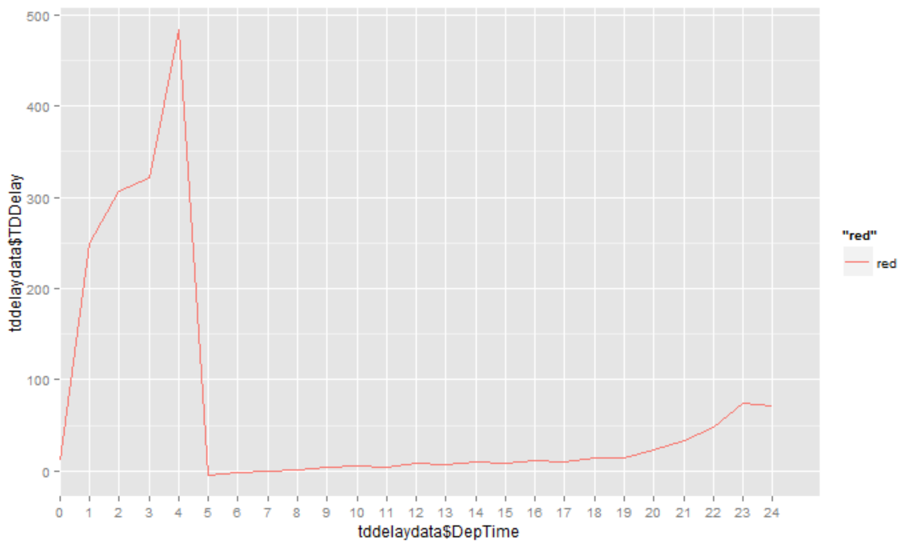
```r
library(ggplot2)
library(sqldf)
library(reshape2)

mydata <-read.csv("ABIA.csv")
mydata[is.na(mydata)]<-0
sqlstring <- "select Month, avg(DepDelay) as Delay from mydata group by Month
"
Monthdelaydata <- sqldf(sqlstring)
qplot(Monthdelaydata$Month,Monthdelaydata$Delay,geom="line",color="red")+scal
e_x_discrete()
```



The plot shows that minimum departure delay occurs during month 9, ie. September, which is the best month to fly to avoid delays. One way to explain this is the winter months usually have their share of severe weather, causing significant delays. The summer months see a surge in air travel and also severe weather.

Next, we consider the delay based on the time of day. To do this we consider the DepTime field in the table but we will truncate it to only the hour of the day, 0-24. For example, 123 will be truncated to 1, meaning 1 hour past midnight.

```
mydata$DepTime=as.integer(mydata$DepTime/100)
sqltddelay <- "select DepTime, avg(DepDelay) as TDDelay from mydata group by
DepTime"
tddelaydata<-sqldf(sqltddelay)
qplot(tddelaydata$DepTime,tddelaydata$TDDelay,geom="line",color="red")+scale_
x_discrete()
```



The plot shows the morning hours after 5 AM and through the afternoon hours are the best to fly to avoid delays.

Note: Although we have used the sqldf package here, the same results could have been achieved by other grouping functions as well. For example, we could have used the ddply function to group the data.

```
library(plyr)
dt<-ddply(mydata,~Month,summarise,Delay=mean(DepDelay))
```

ANSWER 2 - AUTHOR IDENTIFICATION

The first model we will use here is NaiveBayes from the e1071 package. The procedure followed is to construct two document term matrices (DTMs) out of the text files. One DTM will be for the training data, and the other for the test data. The second model we will be using is the Support Vector Machines model which is also considered to be one of the better performing models.

```
library(tm)
library(e1071)
# Uses the readerPlain function
readerPlain = function(fname){
```

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

author_dirs = Sys.glob('./STA380-master/data/ReutersC50/C50train/*')
file_list = NULL
labels = NULL
for(author in author_dirs) {
author_name = substring(author, first=29)
files_to_add = Sys.glob(paste0(author, '/*.txt'))
file_list = append(file_list, files_to_add)
labels = append(labels, rep(author_name, length(files_to_add)))
}
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list
# Preprocessing
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything
lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove nu
mbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # remov
e punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove
excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SM
ART"))
DTM = DocumentTermMatrix(my_corpus)
DTM = removeSparseTerms(DTM, 0.975)
X = as.matrix(DTM)
# Construct the list of authors for training set
auth_list = NULL
for (author in author_dirs) {
author_name=substring(author,first=71)
for (i in 1:50) auth_list= append(auth_list,author_name)
}
auth_vector=as.vector(auth_list)

#Run the Naive Bayes Classifier on the training set
model1<-naiveBayes(X,as.factor(auth_vector))

#Construct the DTM for the test set

author_dirs = Sys.glob('./STA380-master/data/ReutersC50/C50test/*')
file_list = NULL
labels = NULL
for(author in author_dirs) {
author_name = substring(author, first=29)
```

```
files_to_add = Sys.glob(paste0(author, '/*.txt'))
file_list = append(file_list, files_to_add)
labels = append(labels, rep(author_name, length(files_to_add)))
}
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
my_corpus = Corpus(VectorSource(all_docs))
names(my_corpus) = file_list
# Preprocessing
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything
lowercase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove nu
mbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # remov
e punctuation
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove
excess white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SM
ART"))
DTMtest = DocumentTermMatrix(my_corpus)
DTMtest = removeSparseTerms(DTMtest, 0.975)
Y=as.matrix(DTMtest)

# Predict with the Naive Bayes Model
pred1<-predict(model1,Y)

#Run the Support Vector Machines model
model2<-svm(X,as.factor(auth_vector),type="C-classification")

#when you try run predict, it complains about the Y matrix having different d
imensions
dim(X)
dim(Y)

#therefore make X have same dimensions as Y, addding columns without altering
word counts
X<-cbind(X,X)
X<-X[,-1415:-2778]
dim(X)
#retrain the model
model2<-svm(X,as.factor(auth_vector),type="C-classification")
#predict with new model
pred2<-predict(model2,Y)

#check the accuracy of the models.
library(caret)
confusionMatrix(table(pred1,auth_vector))$overall[1]
confusionMatrix(table(pred2,auth_vector))$overall[1]
```

It is found that the accuracy of Naive Bayes (0.6) is better than the SVM method (0.4). In both models, several authors are over-classified, ie. many documents that should not belong to them are classified as being so.

What to do if the test data contains words that are not in the training set? This is a common problem, because it is not possible to foresee every word that can appear in a document in the future. The best approach seems to be the following. We train the classifier initially with a set of words that appear in the training document set. Whenever new documents arrive for testing with new words in them, we can re-train the classifer with such new words and documents. This process of retraining the classifier is aimed at making it better with time. So the classifier is expected to improve with retraining.

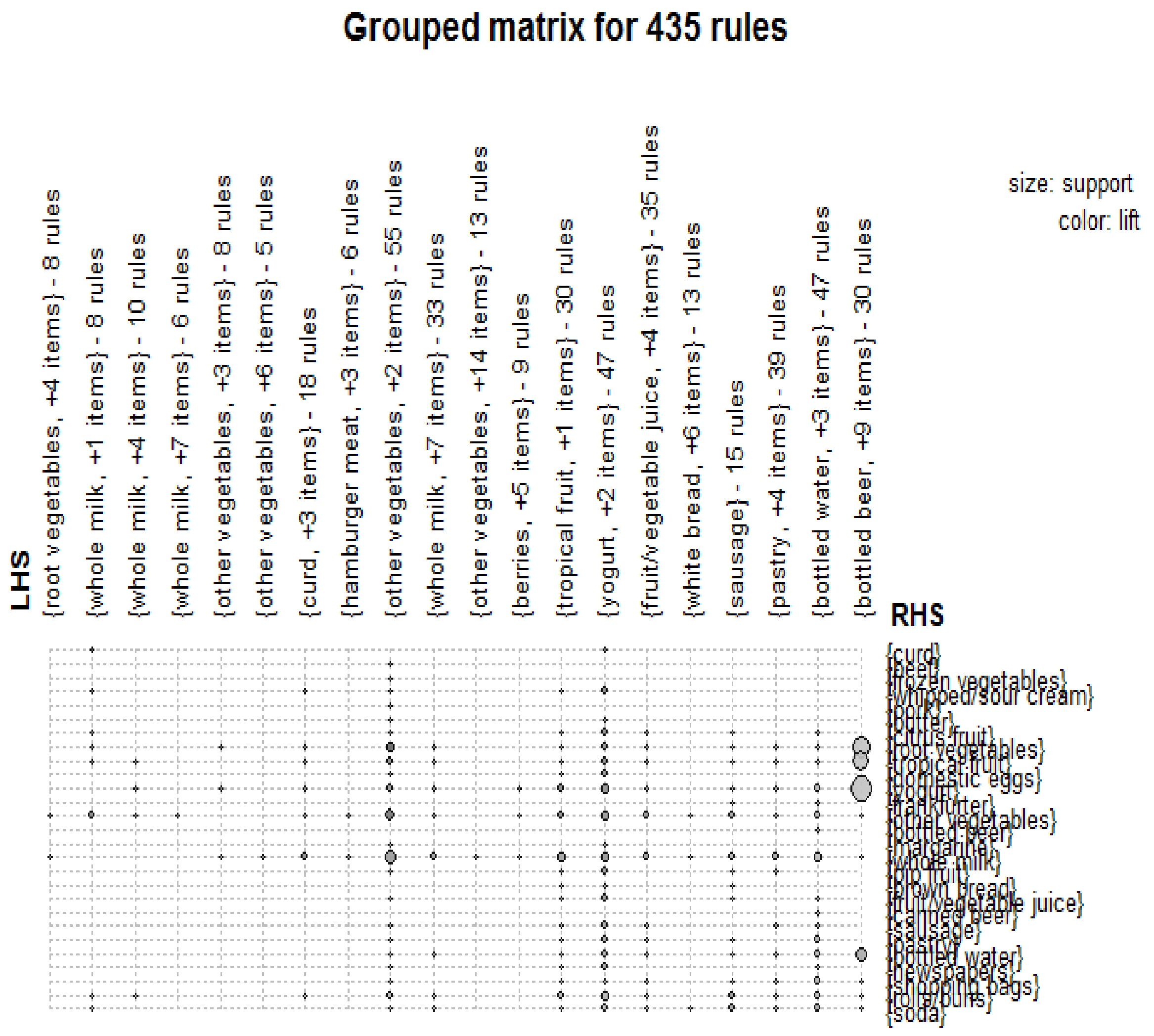
ANSWER 3 We will first walk through the playlists example.

```
library(arules)
playlists <- read.csv("./STA380-master/data/playlists.csv")
playlists$user <- factor(playlists$user)
playlists <- split(x=playlists$artist, f=playlists$user)
playlists <- lapply(playlists, unique)
playtrans <- as(playlists, "transactions")
musicrules <- apriori(playtrans,parameter=list(support=.01, confidence=.5, ma
xlen=4))
inspect(musicrules)
inspect(subset(musicrules, subset=lift > 5))
inspect(subset(musicrules, subset=confidence > 0.6))
inspect(subset(musicrules, subset=support > .02 & confidence > 0.6))
```

Next, we will do the groceries exercise.

```
grocerylists<-read.transactions("./STA380-master/data/groceries.txt",format="
basket",sep=",")
dim(grocerylists)
# 9835 169
# this seems to be right

grocerytrans <- as(grocerylists,"transactions")
rules <- apriori(grocerytrans,parameter = list(support=0.07,confidence=0.5, m
axlen=4))
#there are no rules found for these settings. Trying lower numbers
rules <- apriori(grocerytrans,parameter = list(support=0.01,confidence=0.5, m
axlen=4))
#only 15 rules found. Try even lower numbers
rules <- apriori(grocerytrans,parameter = list(support=0.01,confidence=0.1, m
axlen=4))
#435 rules found
inspect(rules)
inspect(subset(rules,subset=lift>5))
#found nothing. Try lower lift
```

```
inspect(subset(rules,subset=lift>2))
#96 rules found
inspect(subset(rules,subset=confidence>0.3 & support>0.03))
#62 rules found
inspect(subset(rules,subset=confidence>0.3 & support>0.03))
#14 rules found
#Top 3 rules with highest lift
inspect(head(sort(rules,by="lift"),3))
#Use the arulesViz library to plot rules
library(arulesViz)
plot(rules,method="grouped")
```



The rules with the strongest support are the ones with LHS as "bottled beer + 10 items" and RHS root vegetables or tropical fruit or yogurt. Also, rules with whole milk and other items, and yogurt with one other item as LHS seem to have strong support too. This also explains why milk, yogurt, and other such items are placed in the back of the store, allowing the store owners to showcase their other products close to the check out lanes.

Of course, it is possible to get more rules by relaxing the support and confidence numbers.Running the apriori algorithm with support as 0.006 and confidence at 0.1 results in 1121 rules.