Exercises_2

Nimish Amlathe, Stuti Madaan, Hitesh Prabhu

August 15, 2016

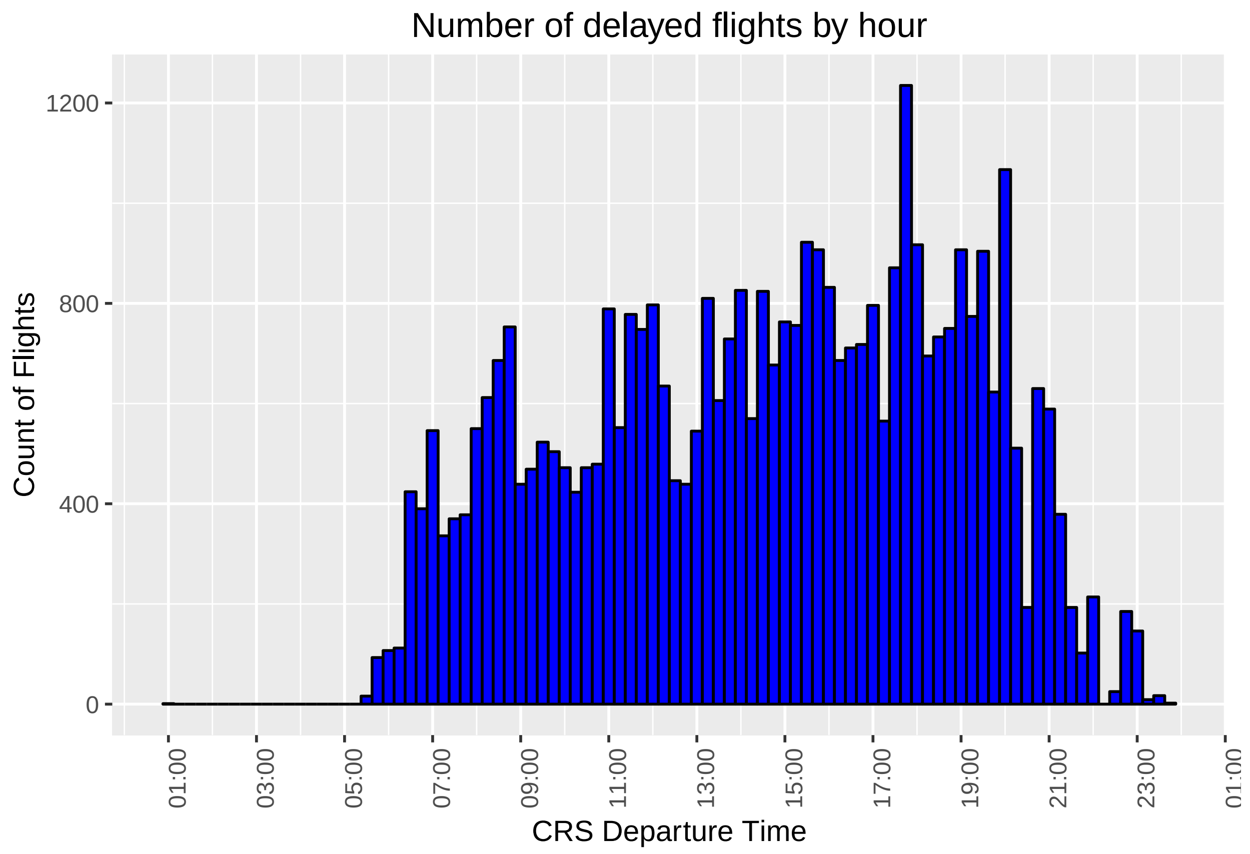
STA 380, Part 2: Exercises 2

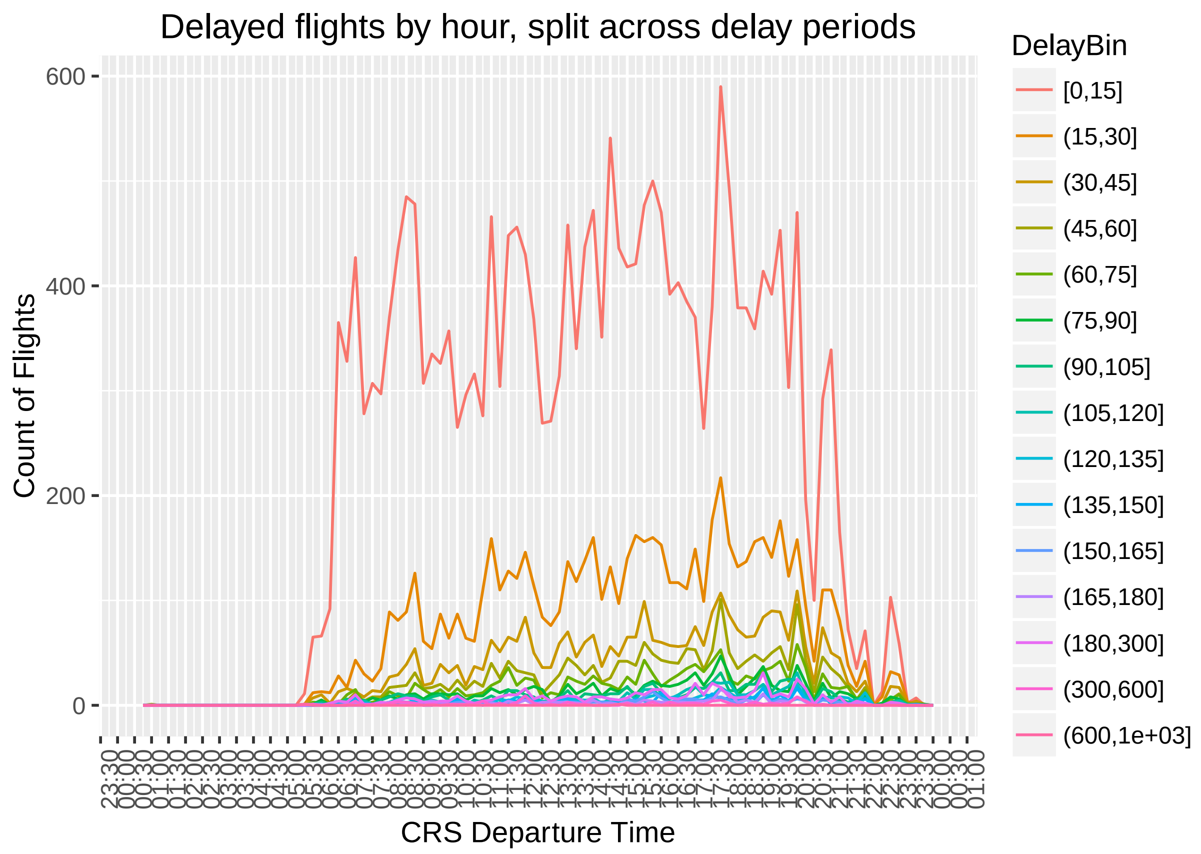
Flights at ABIA

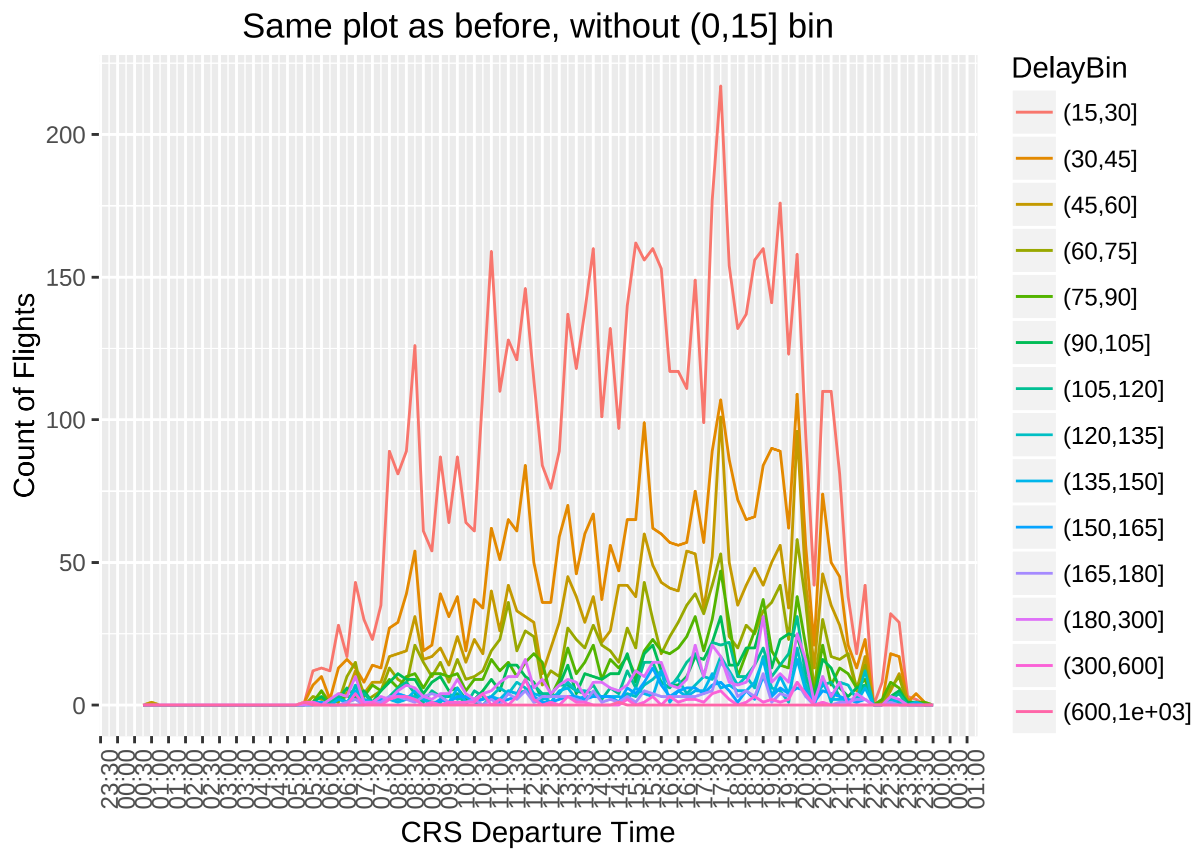
Reading in data and adding additional variables

```r
library(ggplot2)
library(scales) # to access breaks/formatting functions
library(stringr) # To use the str_pad function
library(dplyr)

options(scipen = 999)

ABIA <- read.csv("files/ABIA.csv")
glimpse(ABIA)
ABIA[, 'Departure_flight'] = ifelse(ABIA$Origin == 'AUS',1,0)
```

Quesion: What is the best time of day to fly to minimize delays?

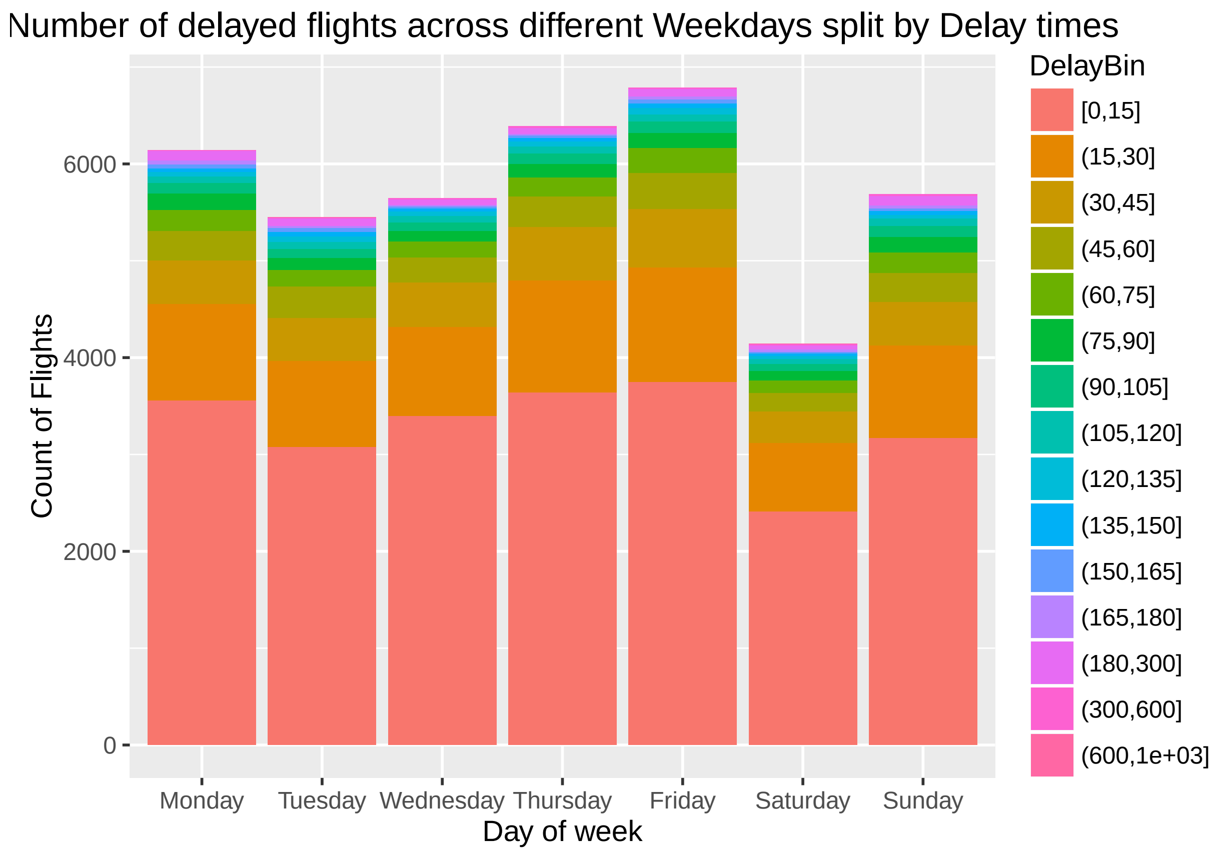


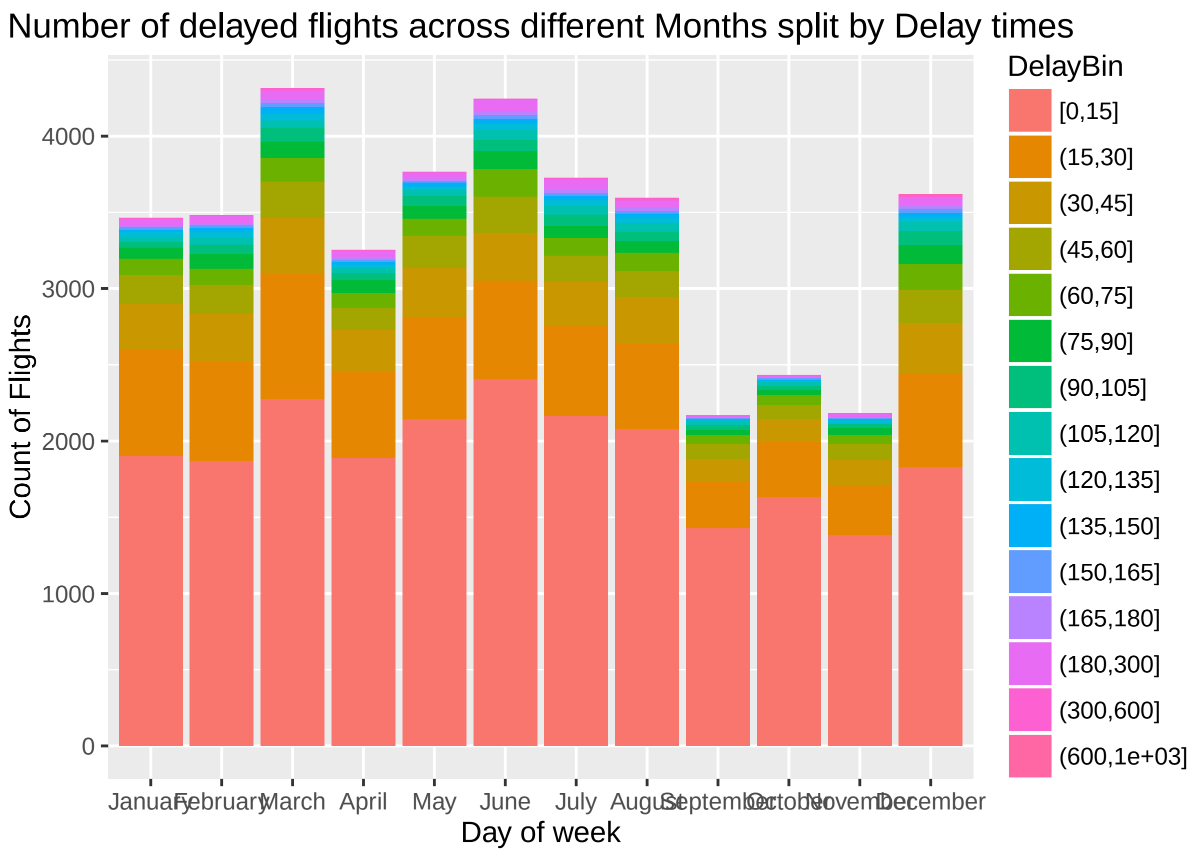




The best time to fly during the day is either in the early morning (between 8.30 to 9.30) or late in the night (after 8.30 pm). One should definitely avoid the time between 5 pm to 8 pm as this period has the most number of delays.

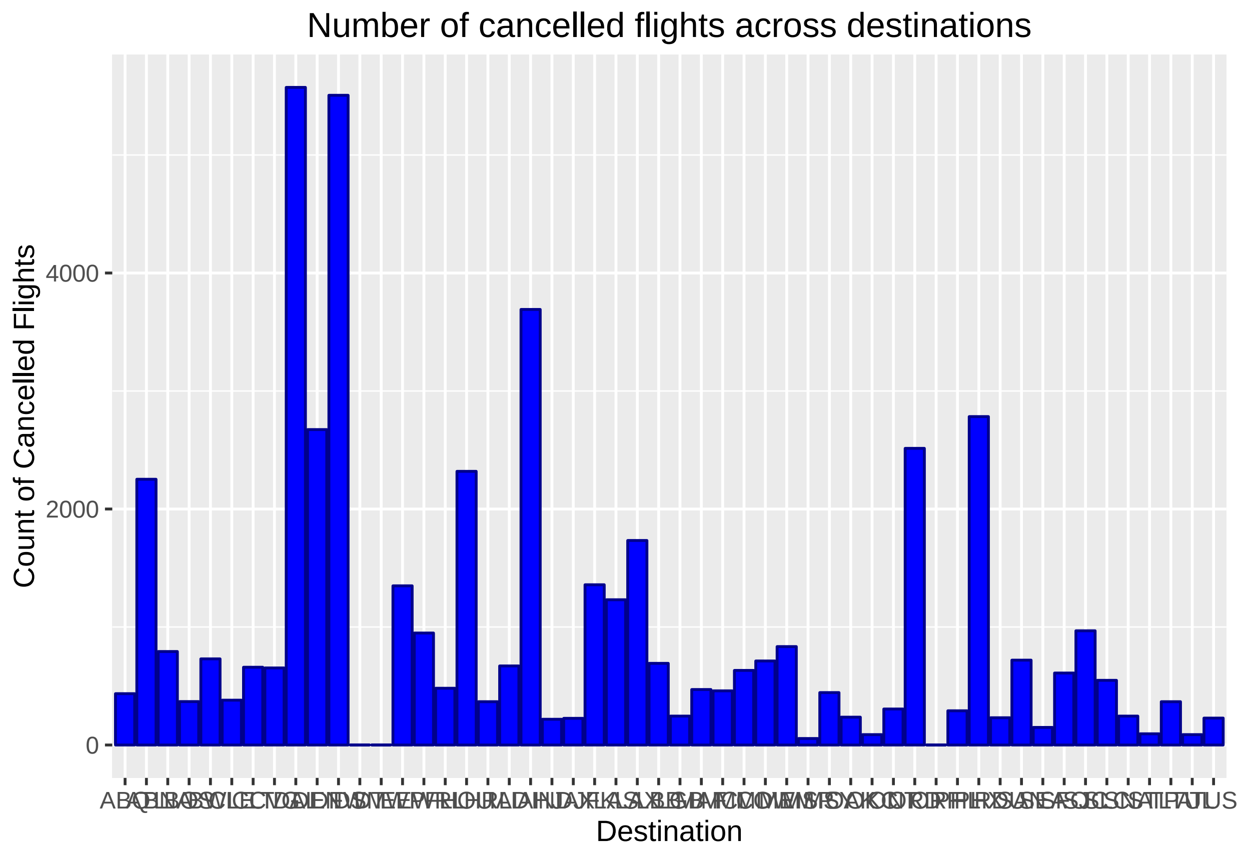
Question 2: What is the best time of year and time of the week to fly to minimize delays?



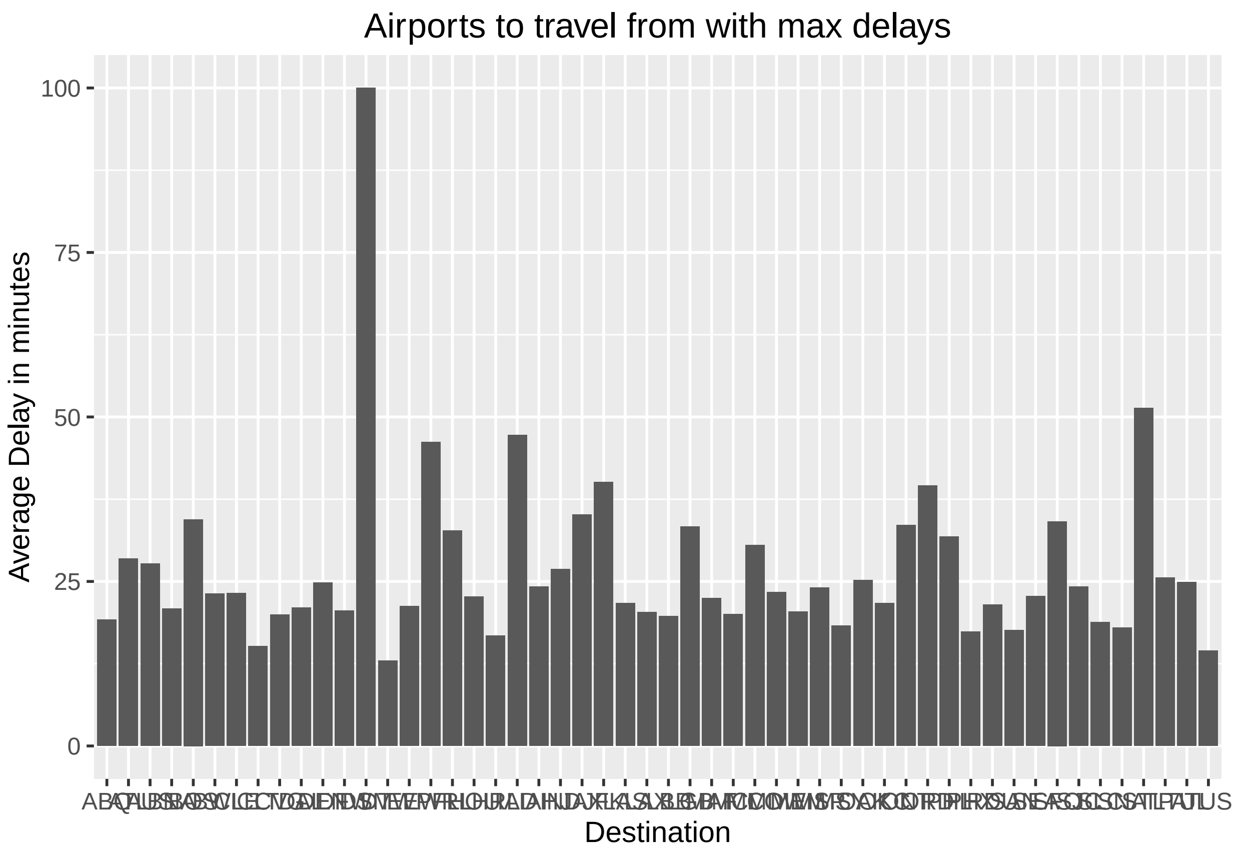


March and June have the most number of delays. After that, May, July, August and December have the next most number of delays. So, January, February and September through November are the best months to fly in.

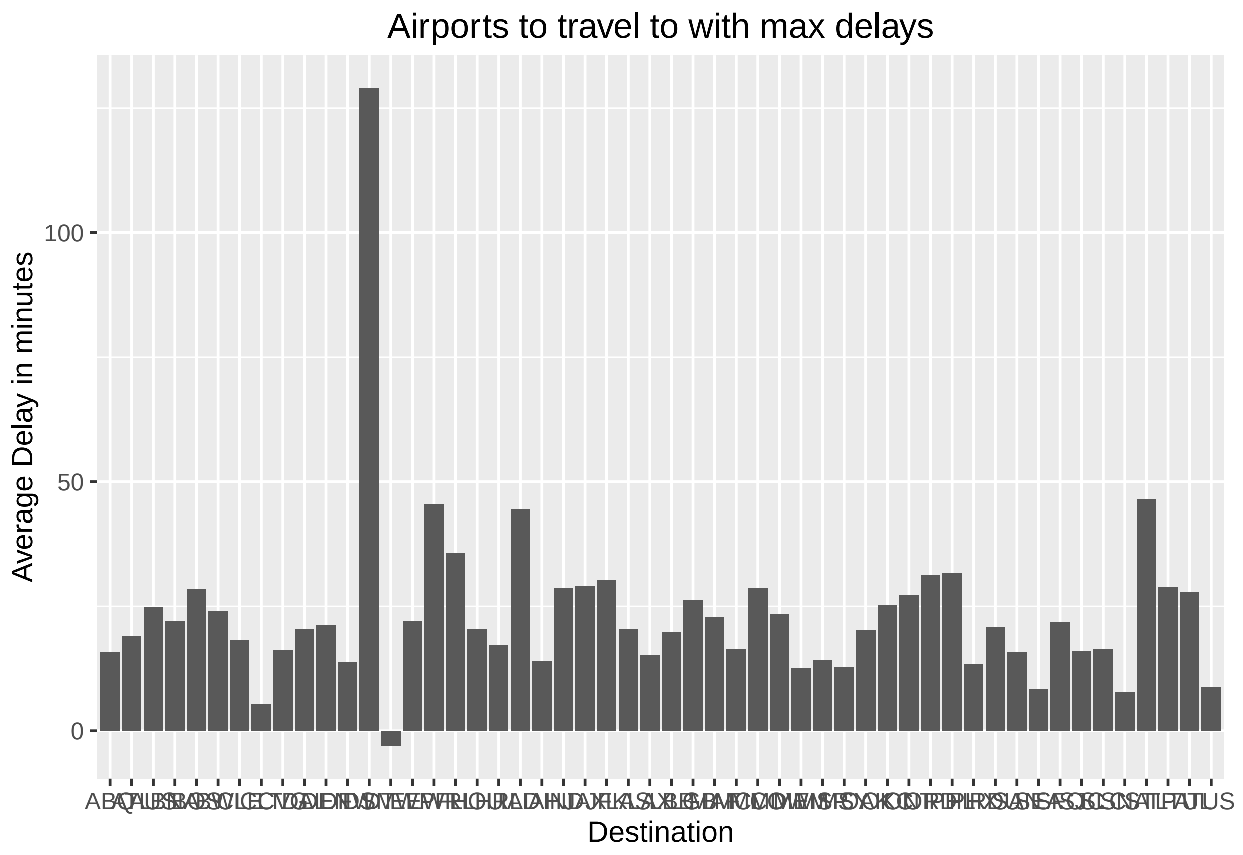
Question: What are the bad airports to fly to?



The histogram shows us that the Dallas Love Field along with Dallas Fort Worth are the worst airports when it comes to cancellation of flights. The George Bush Intercontinental Airport at Houston has the third most number of cancelled flights. In addition, busiest airports in the States like Chicago O'Hare and Atlanta are also among the list of most cancelled flights.



```
## Warning: Stacking not well defined when ymin != 0
```



If we go by the Arrival Delay histogram, the Des Moines International Airport is the worst airport to travel from. However this cannot be validated since our dataset has only one entry for DSM. Thus, Salt Lake City and Washington Dulles International Airport have an average delay time close to 75 mins.

According to the Departure Delay histogram, the Des Moines International Airport is the worst airport to travel to too. Again, this cannot be validated since our dataset has only one entry for DSM. The worst airports to travel too include the likes of Salt Lake City, Washington Duller Airport and Newark Liberty International Airport.

Author attribution

1. Naive Bayes Approach

Reading the files in the directory paths and creating list of files and authors to train on.

```
### Read Data ###
library(tm)
library(foreach)

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

## Reading in all authors' filepaths and author's names
author_dirs = Sys.glob('files/ReutersC50/C50train/*')
```

```
## Combining files and extracting Author names ##
file_list = NULL
labels = NULL
for(author in author_dirs)
{
  author_name = substring(author, first = 27)
  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)))
}

## Removing ',txt'from file names
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))
```

Now we create our training Document-term Matrix

```
# Creating training corpus and processing
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 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"))

# Creating DTM
DTM = DocumentTermMatrix(my_corpus)

# Removing sparse items
DTM = removeSparseTerms(DTM, 0.975)
```

Using the DTM, we calculate probabilities for each word in the corpus, and then we calculate the conditional probabilities for each document. This list w will contain 2500 rows (1 for each document) and 1389 columns (one for each word) alongwith probability for each word.

```
# Now a dense matrix, smoothing factor and find probability vectors
X = as.matrix(DTM)
smooth_count = 1/nrow(X)

# Naive Bayes: the training sets for all the authors
w = list()
smooth_count = 1/nrow(X)
j = 1
for (i in seq(1,length(file_list),50) )
{
  w[[j]] = colSums(X[i:(i + 49),] + smooth_count)/sum(colSums(X[i:(i + 49),]
                                                         + smooth_count))
  j = j + 1
}
```

Reading the files in the directory paths and creataing list of files and authors to test on.

```r
# Test Data Preparation similar to Training Data . Final DTM matrix to be used for modelling

readerPlain = function(fname){
  readPlain(elem = list(content = readLines(fname)),
            id = fname, language = 'en') }
## Rolling two directories together into a single corpus
author_dirs_test = Sys.glob('files/ReutersC50/C50test/*')
#author_dirs = author_dirs[1:2]
file_list_test = NULL
labels_test = NULL
for(author in author_dirs_test) {
  author_name = substring(author, first = 26)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list_test = append(file_list_test, files_to_add)
  labels_test = append(labels_test, rep(author_name, length(files_to_add)))
}

# Need a more clever regex to get better names here
all_docs_test = lapply(file_list_test, readerPlain)
names(all_docs_test) = file_list_test
names(all_docs_test) = sub('.txt', '', names(all_docs_test))

my_corpus_test = Corpus(VectorSource(all_docs_test))
names(my_corpus_test) = labels_test

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

DTM_test = DocumentTermMatrix(my_corpus_test)
```

Next, we remove any new words in the test dataset that the training dataset hasn't *seen* yet.

```r
# Keeping only those words which we used in the training set.
common_words = colnames(DTM_test)[colnames(DTM_test) %in% colnames(DTM)]
DTM_test <- DTM_test[, common_words]
```

Now we calculate the log-probabilities for each combination of document and author. THe author with the highest log-probability will be assigned to that document.

```r
# Taking test documents and Comparing log probabilities
X_test = as.matrix(DTM_test)
# Creating empty matrix to calculate log-probabilities
Y_test = matrix(0, nrow = 2500, ncol = 50)
K = list()
j = 1
for (i in 1:2500)
{
  for (j in 1:50)
  {
    Y_test[i,j] = sum(X_test[i,]*log(w[[j]]))
  }
}

# Finding the document which corresponds to maximum log-probability (Hence the author)
# This can be done in a more readable way using for loop
library(dplyr)
author_predictions <- as.vector(t(as.data.frame(t(Y_test)) %>%
                        summarise_each(funs(which.max(.) ) ) ) )
# Since authors are arranged so well with one author for every fifty files
author_actual <- as.vector(rep(1:50,each=50))
```

Finally, we calculate the confusion matrix to see how many times we predicted the correct author.

```r
#confusion matrix
library(caret)
library(e1071) # Weird. ConfusionMatrix asked for this library
confMatrix <- confusionMatrix(author_predictions, author_actual)
confMatrix$overall["Accuracy"]
```

```
## Accuracy
##   0.6036
```

The accuracy of the Naive-Bayes model was found to be 60.4%.

2. Random Forests

The 2nd model we chose to utilize was Random forests.

```r
library(randomForest)
library(caret)
library(e1071) # Weird. ConfusionMatrix asked for this library
set.seed(2)

rffit = randomForest(y = as.factor(rep(1:50,each=50)), x = as.matrix(DTM),
                     mtry = 50, ntree = 500)

predicted_price_rftree <- predict(rffit, newdata = as.matrix(DTM_test))
confMatrix_rf <- confusionMatrix(as.vector(predicted_price_rftree), author_actual)
```

```
## Warning in confusionMatrix.default(as.vector(predicted_price_rftree),
## author_actual): Levels are not in the same order for reference and data.
## Refactoring data to match.
```

```r
confMatrix_rf$overall["Accuracy"]
```

```
## Accuracy
##   0.6144
```

The accuracy of the random forest model was found to be 61.7%.

Association rule mining

Read in the grocery data.

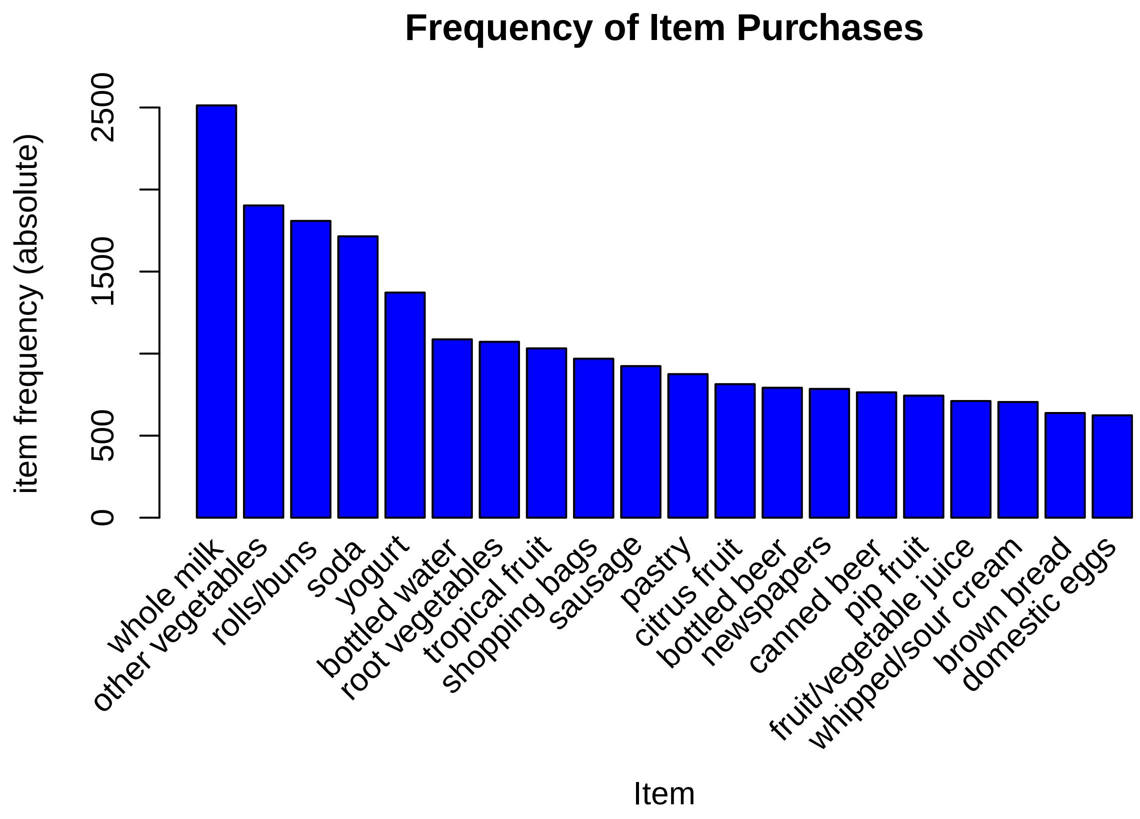
- First creating a list of baskets: vectors of items by consumer.
- Analagous to bags of words.
- Apriori algorithm expects a list of baskets in a special format.

- Removing duplicates and then Casting this variable as a arules "transactions" class.

```
library(arules)
library(arulesViz)

groceries = read.transactions(file = "files/groceries.txt", rm.duplicates = TRUE, format = "basket", sep
```

Let's plot an Item-frequency chart to guage how much more often certain items are present in the dataset.

```
# Plotting top 20 items by frequency
itemFrequencyPlot(groceries,topN=20,type = "absolute", col = 'blue', xlab = 'Item', main = 'Frequency of
```



Running the 'apriori' algorithm.

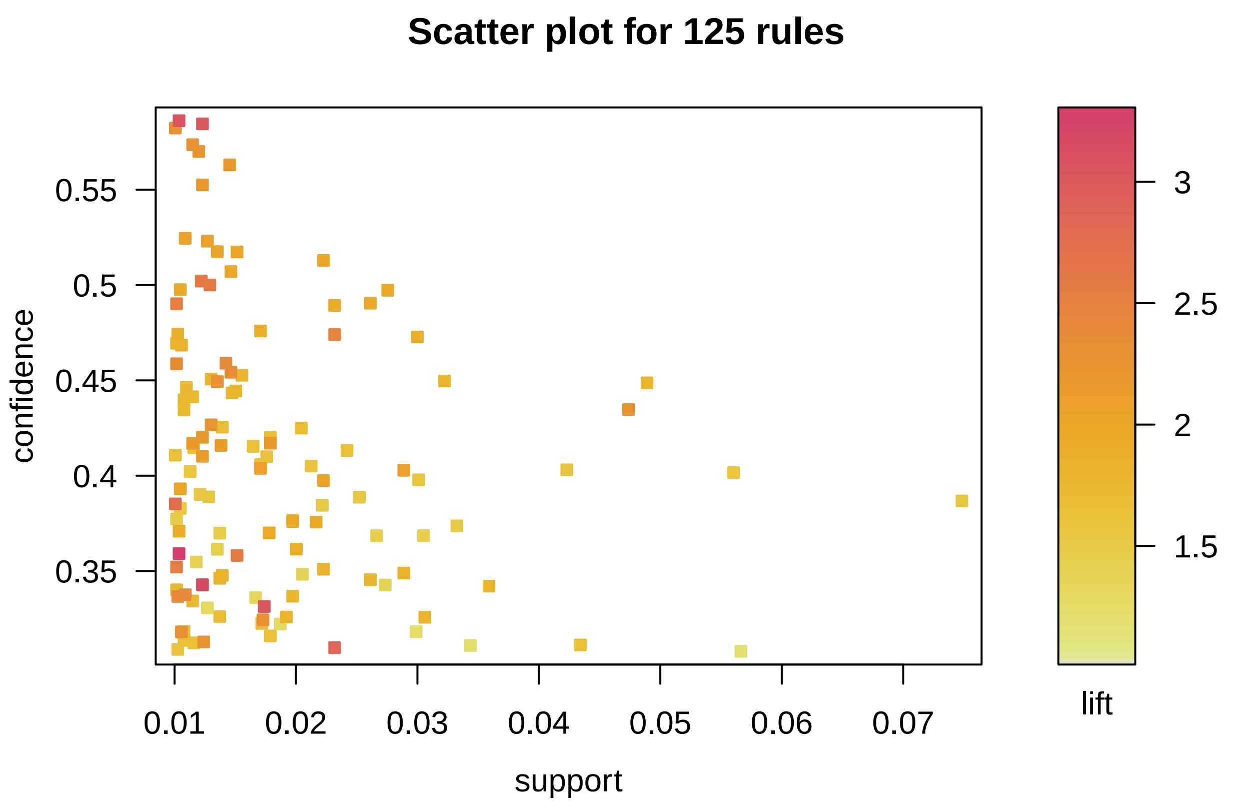
Looking at rules with support > .01 & confidence >.5 & length #items <= 6

```
grocrules <- apriori(groceries, parameter=list(support=.01, confidence=.5, maxlen=6))
inspect(grocrules)
```

Trying various subsets

```
## Trying various subsets
inspect(subset(grocrules, subset = lift > 3))
inspect(subset(grocrules, subset = confidence > 0.5))
inspect(subset(grocrules, subset = support > .01 & confidence > 0.3))
```

```
rules = apriori(groceries, parameter = list(support=.01, confidence=.3, target='rules'))
plot(rules)
```



Conclusion

Whole milk, other vegetables and yogurt are some of the most likely to be purchased items based on various itemsets. These are also amongst the items with the highest support counts.

The various itemsets we have seen so far point to associations between people who buy a certain kind of items also buying some of the more frequently occuring items. For example, people who buy a lot of diary products tend to also buy milk, and people how buy a lot of fruits and vegetables also tend to biuy milk and other vegetables, etc. We did not see any rare patterns or patterns amongst itemsets with very low support (some niche products, etc.)