Exercise 2

Kemei Zhuo

August 18, 2018

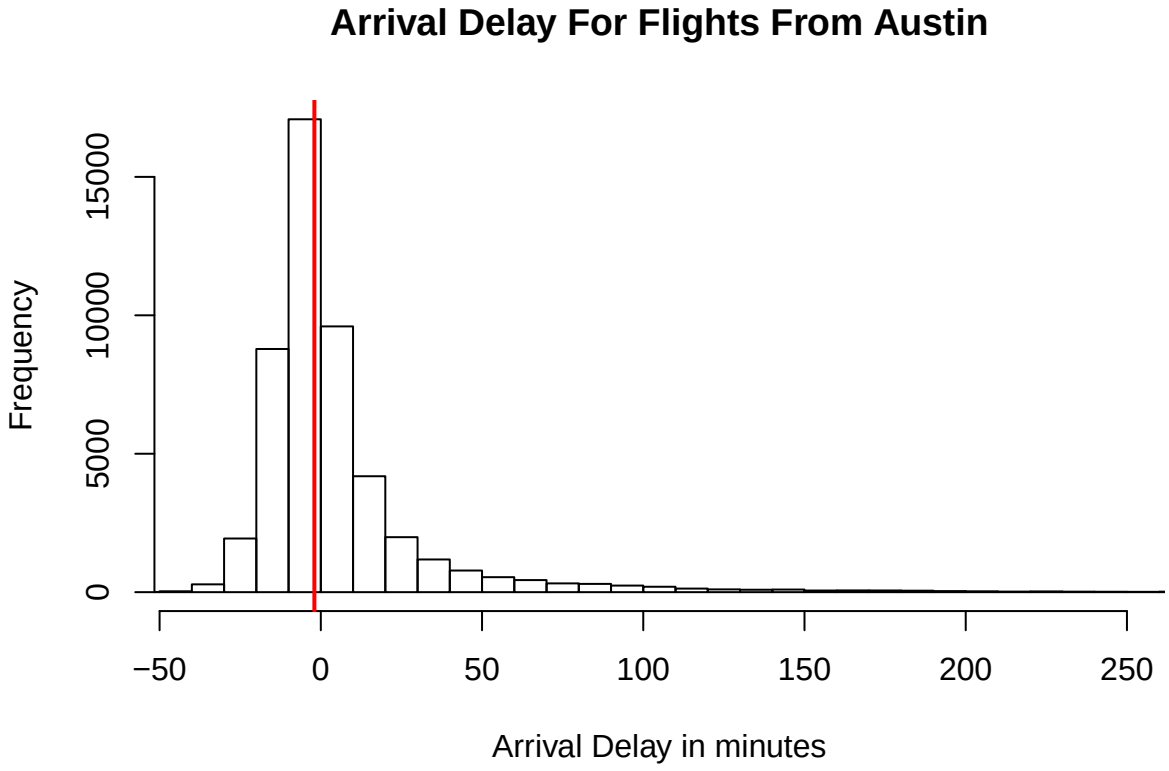
Flights at ABIA

Summary: We should take flights in the first half of month in September, October, and November, avoid the Friday, and choose flight in the morning if we want to fly from Austin. The arrival delay is not related to distance, and arrival airport for both incoming flights and outgoing flights.

Flights from Austin

I separated the dataset into flights that fly from Austin and flights that fly into Austin.

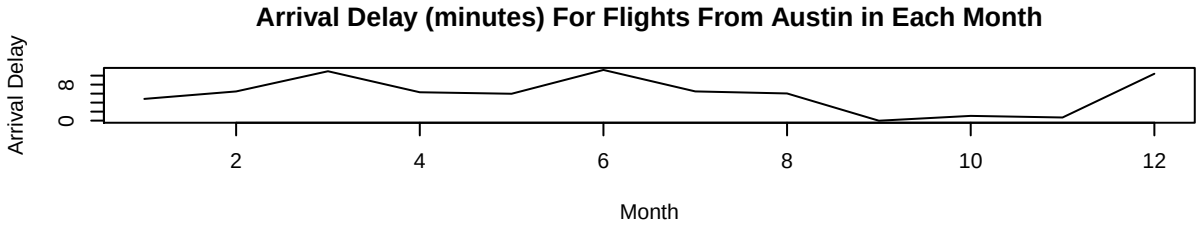
We can see below that arrival delay is not a serious phenomenon because more than half of the flights arrive early. Still, we want to avoid the delay by avoiding the busy hour.

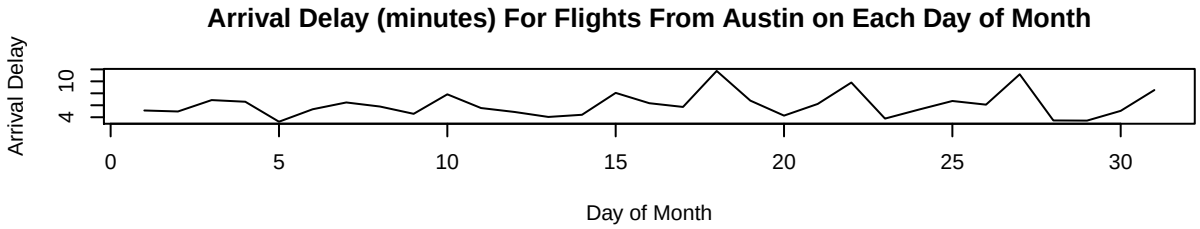


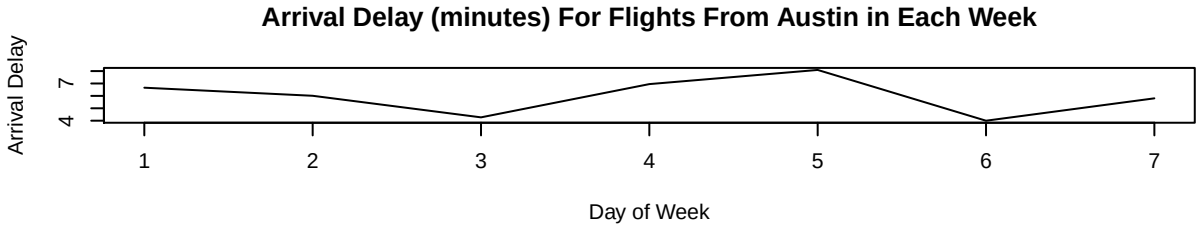
We can see from the histogram of delay in each months that March, June, and December have high delays, but September, October, and November have low delay minutes.

We can aslo see from the histogram of delay in different day of month that the delay increases during the second half of month.

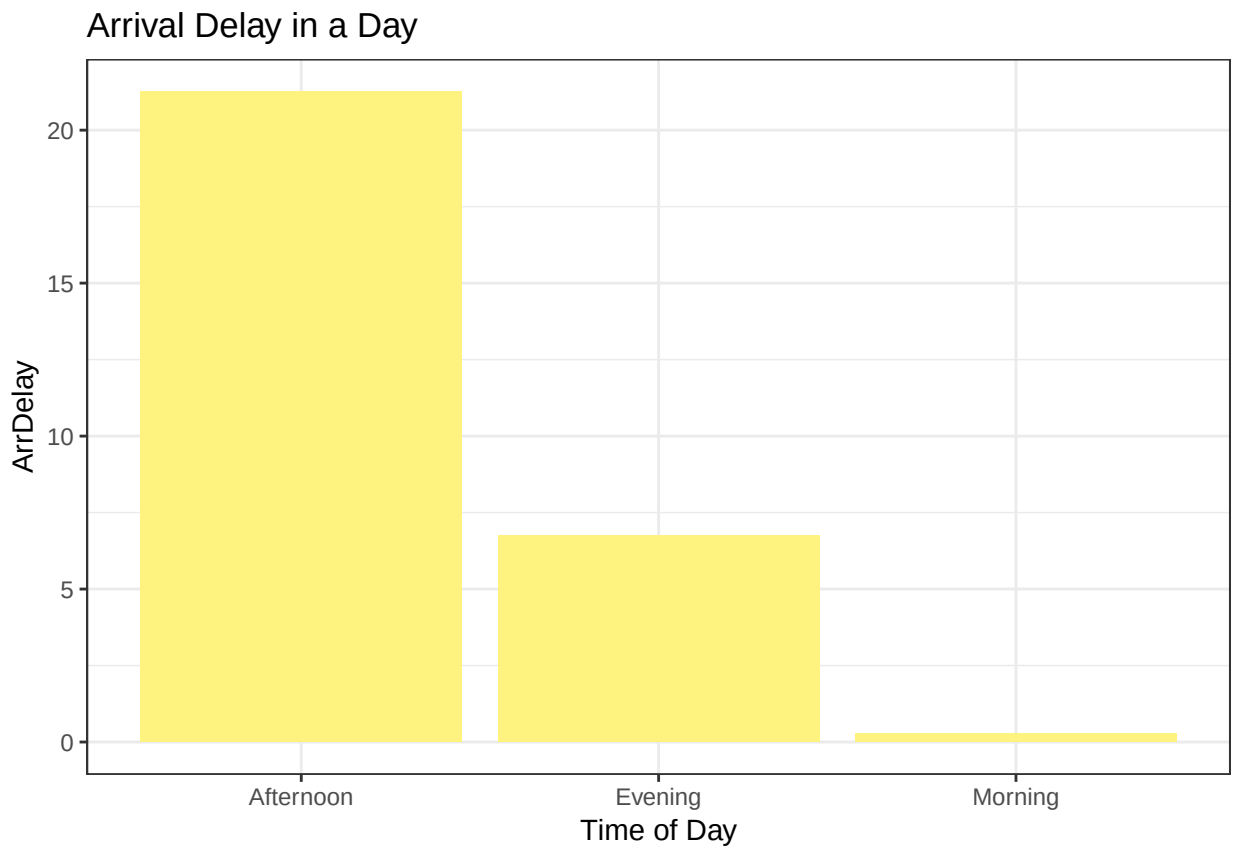
We can also see from histogram of delay in a week that the delay is highest on Friday's flights.







We can also see from histogram of delay in a day that the afternoon flight has highest delay and the flights in morning have lowest delay.



Thus, we should take flights in the first half of month in September, October, and November, avoid the Friday, and choose flight in the morning if we want to fly from Austin.

I am also interested to see whether the delay is related to the number of flights. We can see that

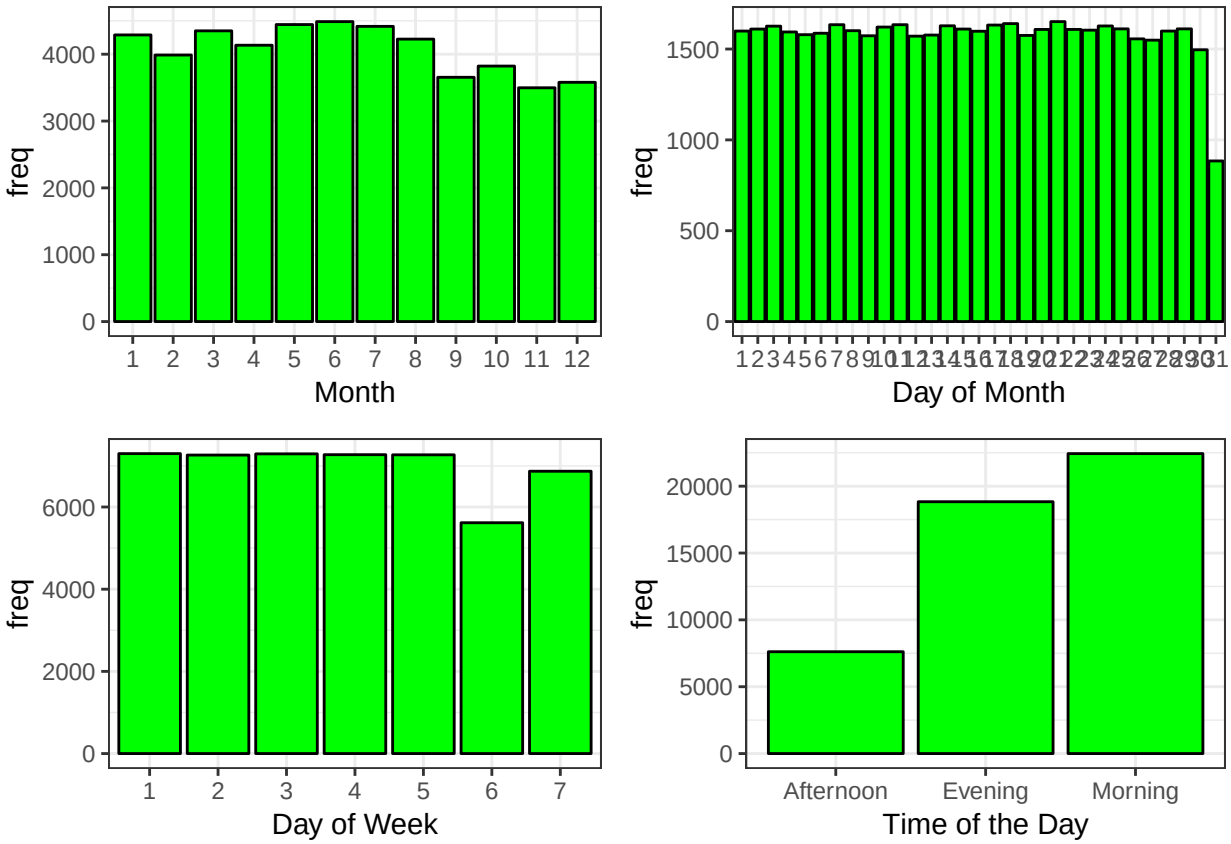
*September, October, November, and December have less number of flights

*The number of flights in each day of month is similiar

*Saturday has less flights.

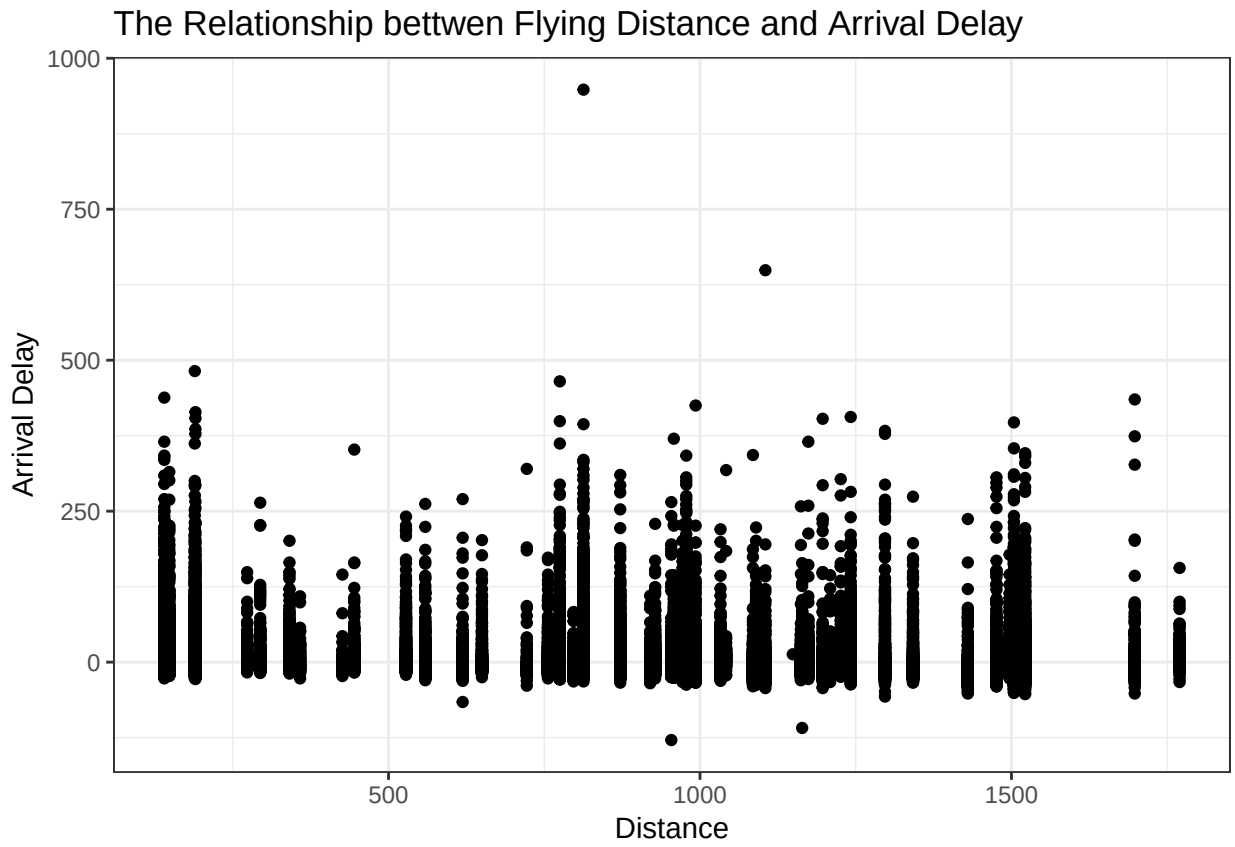
*The flights in morning > the flights in evening > the flgihts in afternoon

Thus, we can conclude that there's no relatinoship between arrival delay and numebr of flights.

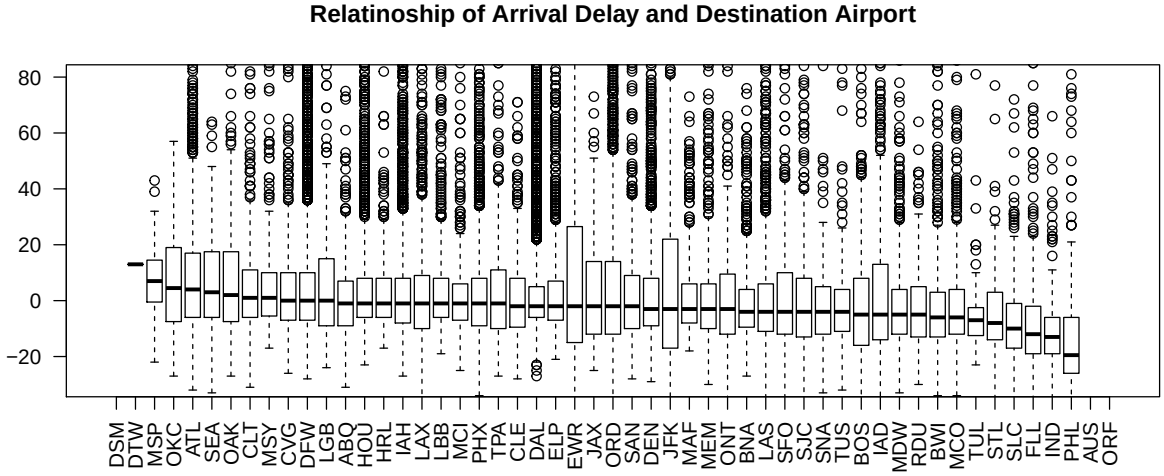


In addition, I wonder if the distance would matter in arrival delay because international flights are almost on time.

*Below we can see that there is no clear relationship.

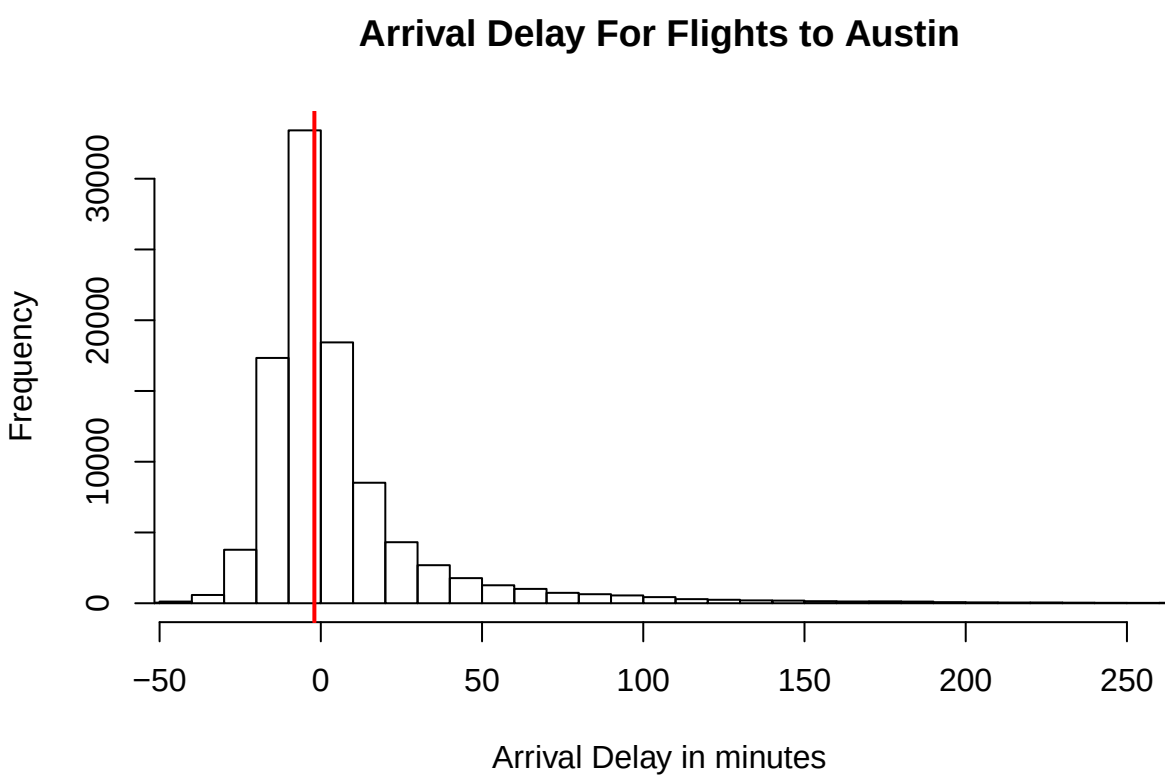


Then, I look into the relationship between Arrival Delay and the destination airport. Last week, I flied from Austin to Antalanta and my flights delayed for 2 hours. We can see that ATL has top arrival delay.

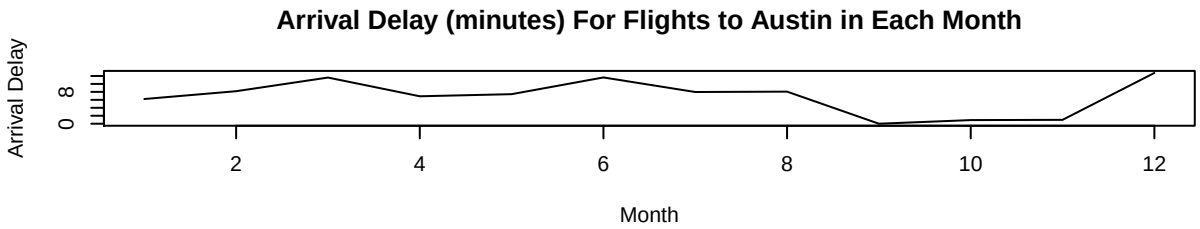


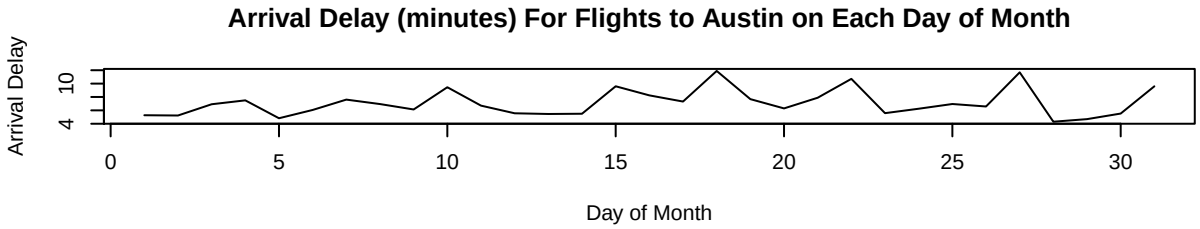
Flights to Austin

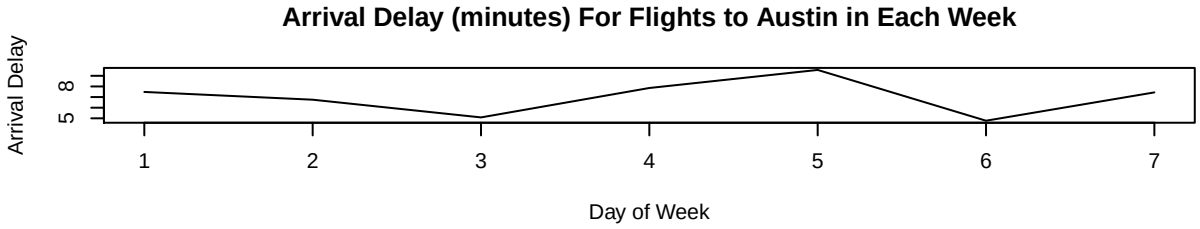
We can see below that arrival delay histogram is similiar.



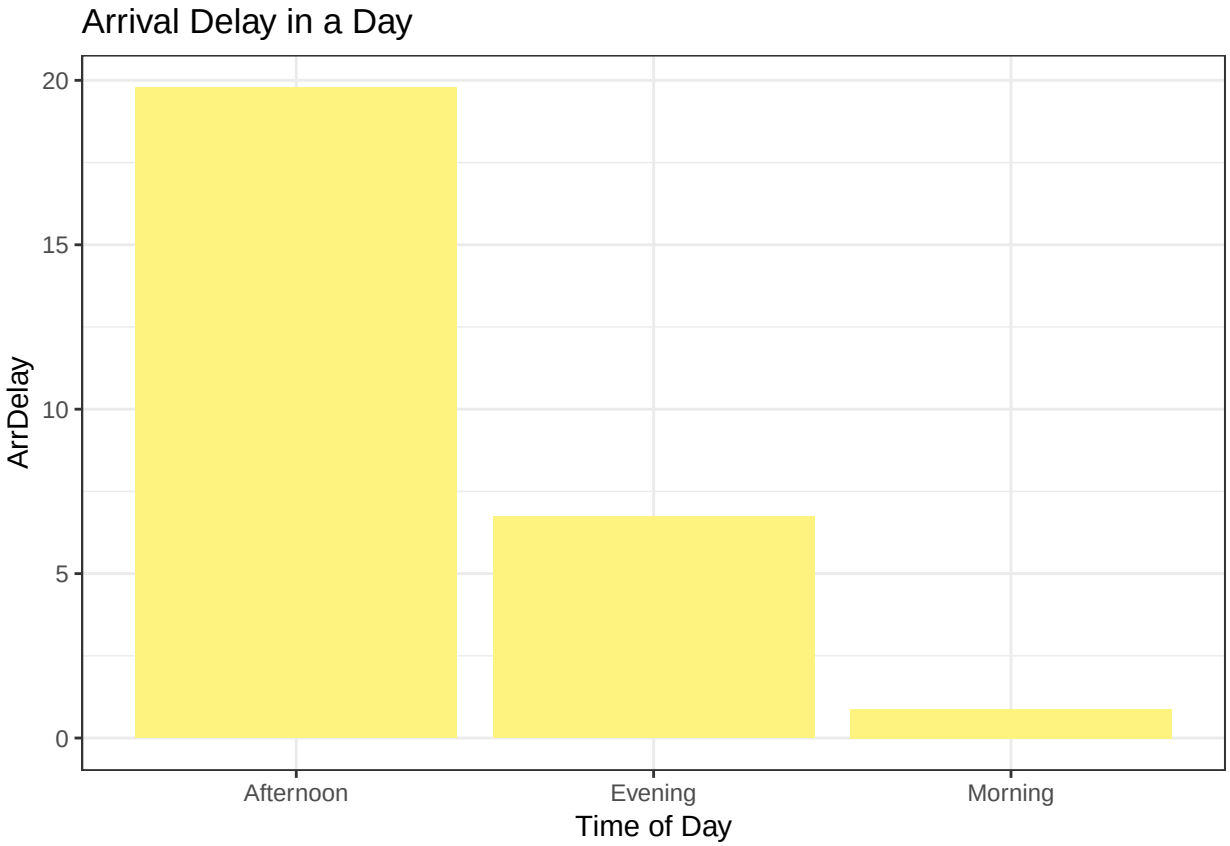
The insights drawn on plots below are very similiar to what we see on flights from Austin.

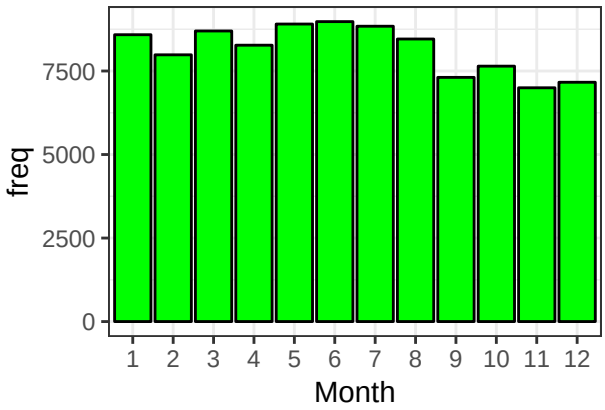


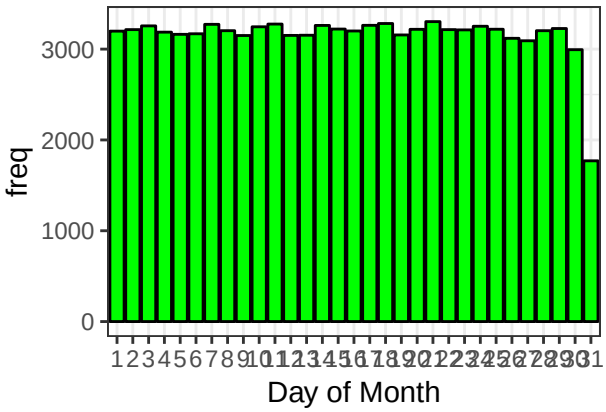




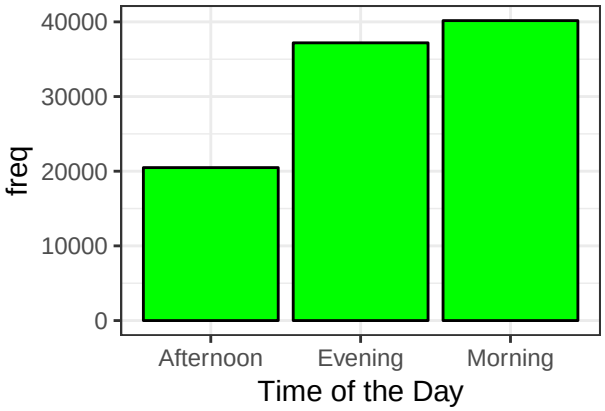
We can also see from histogram of delay in a day that the afternoon flight has highest delay and the flights in morning have lowest delay.

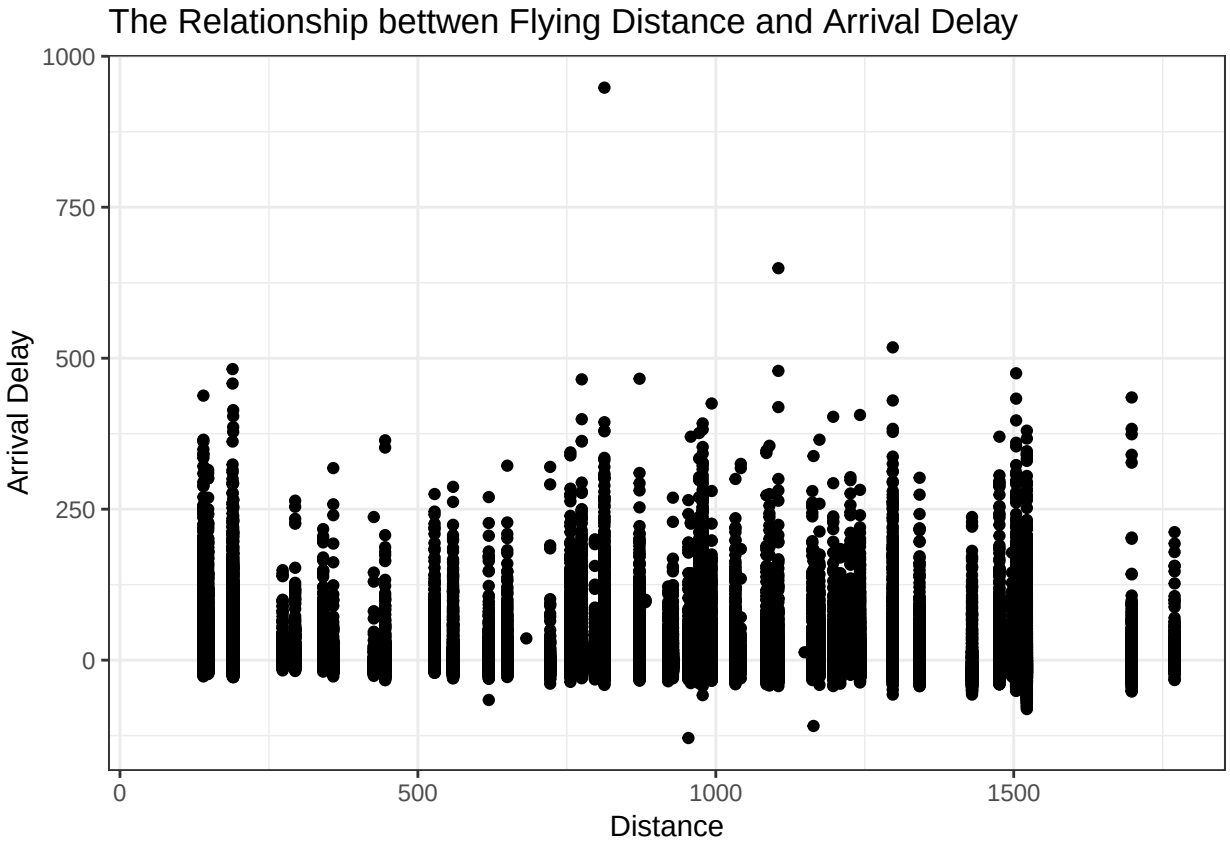

Arrival Delay in a Day
ArrDelay
20
15
10
5
0
Afternoon
Evening
Morning
Time of Day


freq
7500
5000
2500
0
1 2 3 4 5 6 7 8 9 10 11 12
Month

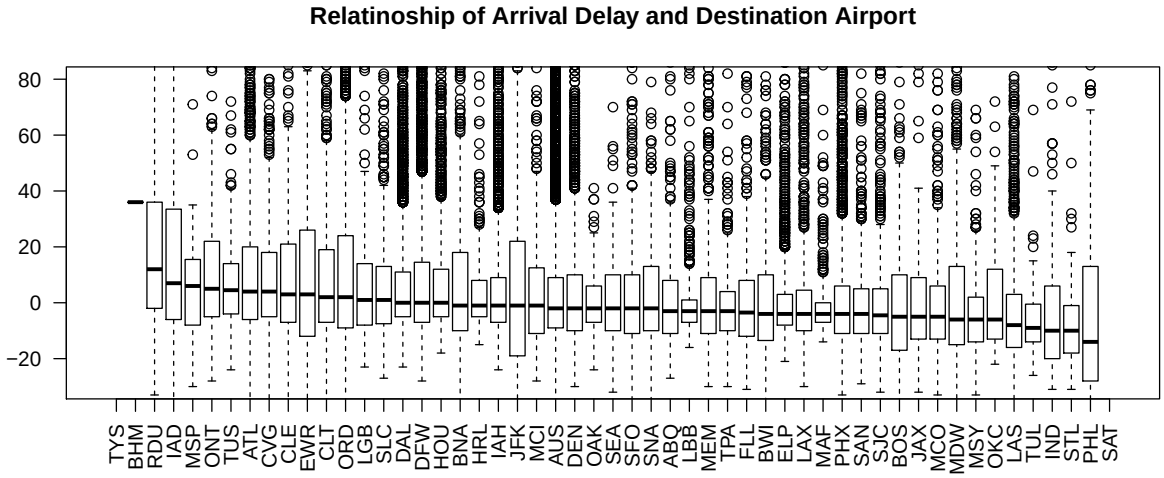

freq
3000
2000
1000
0
Day of Month


freq
15000
10000
5000
0
1 2 3 4 5 6 7
Day of Week


freq
40000
30000
20000
10000
0
Afternoon
Evening
Morning
Time of the Day



We can see that the airports with highest delay are different.



Conclusion: We should take flights in the first half of month in September, October, and November, avoid the Friday, and choose flight in the morning if we want to fly from Austin. The arrival delay is not related to distance. For outgoing flights, MSP, OKC, and ATL has highest arrival delay. For incoming flights, RDU, IAD, and MSP has highest arrival delay.

Author attribution

I used the naivebayes model and random Forest to predict the author identities in the test data. My way of dealing new words in test data is to cut it out in the beginning. I assumed that the same author would have similir word style and thus the new words would not be significant. Thus, I used `control=list(dictionary=sort(Terms(DTM_simon)))` to make sure that the test data has the same column names as the train data, which makes the prediction later much pleasant.

The accuracy of Naivebayes is 0.2828, and the accuracy for randomForest is 0.6168. Thus, randomForest is much better than naivebayes. The ttHillis, idLawder, eMacartney are hard to classify while Gilchrist, ikoFujisaki, and nleyBrowning are easier to classify.

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
library(magrittr)

setwd('C:/Users/zjfhz/Desktop/predictive model/STA380-master')

readerPlain = function(fname){

  readPlain(elem=list(content=readLines(fname)),

            id=fname, language='en') }
# file path

author_dirs1 = Sys.glob('data/ReutersC50/C50train/*')

author_dirs2 = Sys.glob('data/ReutersC50/C50test/*')

file_list = NULL

file_list1 = NULL

labels = NULL

labels1 = NULL

labels2 = NULL

for(author in author_dirs1) {

  files_to_add = Sys.glob(paste0(author, '/*.txt'))

  file_list = append(file_list, files_to_add)

  author_name = substring(author, first=29)
```

```
  labels1 = append(labels1, rep(author_name, length(files_to_add)))
}

for(author in author_dirs2) {

  files_to_add = Sys.glob(paste0(author, '/*.txt'))

  file_list1 = append(file_list1, files_to_add)

  author_name = substring(author, first=28)

  labels2 = append(labels2, rep(author_name, length(files_to_add)))
}

file_list2 = append(file_list,file_list1)

docs1 = lapply(file_list, readerPlain)
docs2 = lapply(file_list1, readerPlain)

name1 = file_list %>%

{ strsplit(., '/', fixed=TRUE) } %>%

{ lapply(., tail, n=2) } %>%

{ lapply(., paste0, collapse = '') } %>%

  unlist

name2 = file_list1 %>%

{ strsplit(., '/', fixed=TRUE) } %>%

{ lapply(., tail, n=2) } %>%

{ lapply(., paste0, collapse = '') } %>%

  unlist
# Rename the articles

names(docs1) = name1
names(docs2) = name2


documents_raw = Corpus(VectorSource(docs1))

my_documents = documents_raw

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

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

my_documents = tm_map(my_documents, content_transformer(removePunctuation)) # remove punctuation
```

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

my_documents = tm_map(my_documents, content_transformer(removeWords), stopwords("en"))

DTM_simon = DocumentTermMatrix(my_documents)

DTM_simon = removeSparseTerms(DTM_simon, 0.95)

X_train = as.matrix(DTM_simon)

documents_raw = Corpus(VectorSource(docs2))

my_documents = documents_raw

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

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

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

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

my_documents = tm_map(my_documents, content_transformer(removeWords), stopwords("en"))

DTM_simon = DocumentTermMatrix(my_documents, control=list(dictionary=sort(Terms(DTM_simon))))

X_test = as.matrix(DTM_simon)

#reorder the column names
X_train = X_train[ , order(colnames(X_train))]
X_test = X_test[ , order(colnames(X_test))]
```

```
library(e1071)
data_trn = cbind(data.frame(X_train), labels1)
model1 = naiveBayes(labels1~., data=data_trn, laplace=1)
X_test = data.frame(X_test)
y_test = labels2
pred_nb = predict(model1, X_test)
t=table(pred_nb, y_test)
sum(diag(t))/length(y_test)
```

```
## [1] 0.2828
```

```
library(randomForest)
```

```
## randomForest 4.6-12
```

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

```
##
## Attaching package: 'randomForest'
```

```
## The following object is masked from 'package:gridExtra':
##
##     combine
```

```
## The following object is masked from 'package:ggplot2':
##
##     margin
```

```
model2 = randomForest(labels1~., data=data_trn)
pred_rf = predict(model2, X_test)
t=table(pred_rf, labels2)
sum(diag(t))/length(y_test)
```

```
## [1] 0.6096
```

```
# look at prediction on each author
sort(diag(t))
```

```
##       ttHillis       cAuchard     eMacartney       idLawder     aFernandes
##              7             10             10             10             15
##        tinWolk   jaminKangLim       uelPerry      liamKazer    xanderSmith
##             15             16             16             18             18
##  renSchuettler  therScoffield        eDickie        dNissen      ahDavison
##             19             19             21             22             24
##    inDrawbaugh     nardHickey    resePoletti        rreTran     stinRidley
##             26             26             27             27             27
##     inMorrison          Ortiz      nMastrini         iaZajc      athanBirt
##             28             28             30             31             32
##       dDorfman        Farrand     haelConnor        Lopatka        nCrosby
##             32             32             34             34             34
##          EeLyn      kBendeich         kLouth   riciaCommins     erHumphrey
##             35             37             37             37             38
##    interbottom         thWeir   celMichelson    neO'Donnell       onCowell
##             38             39             40             41             41
##    hamEarnshaw      erFillion        inSidel       lPenhaul      thewBunce
##             42             43             43             44             45
##   nleyBrowning     onPressman roshKarimkhany    ikoFujisaki      Gilchrist
##             46             46             46             48             50
```

Practice with association rule mining

After I read the data, I created the apriori algorithem that has support higher than 0.005 and confidence larger than .2.

```
## -- Attaching packages --------------------------------- tidyverse 1.2.1 --
```

```
## v tibble  1.4.2     v purrr   0.2.4
## v tidyr   0.7.2     v dplyr   0.7.4
## v readr   1.1.1     v stringr 1.2.0
## v tibble  1.4.2     v forcats 0.2.0
```

```
## -- Conflicts ------------------------------------ tidyverse_conflicts() --
## x NLP::annotate()        masks ggplot2::annotate()
## x dplyr::combine()       masks randomForest::combine(), gridExtra::combine()
## x tidyr::extract()       masks magrittr::extract()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x randomForest::margin() masks ggplot2::margin()
## x purrr::set_names()     masks magrittr::set_names()
```

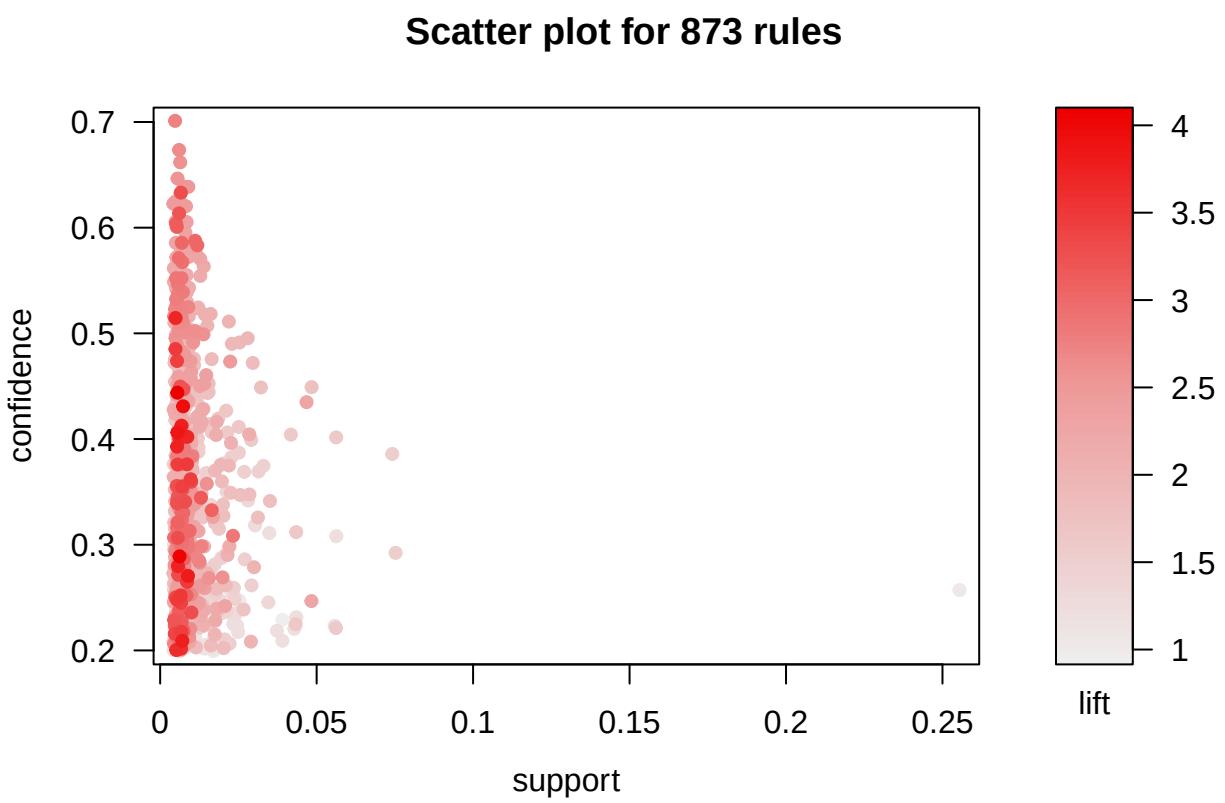
```
## Loading required package: Matrix
##
## Attaching package: 'Matrix'
## The following object is masked from 'package:tidyr':
##
##     expand
##
## Attaching package: 'arules'
## The following object is masked from 'package:dplyr':
##
##     recode
## The following object is masked from 'package:tm':
##
##     inspect
## The following objects are masked from 'package:base':
##
##     abbreviate, write
## Loading required package: grid
```

```
#'apriori' algorithm
rules = apriori(groceries,
                    parameter=list(support=.005, confidence=.2, maxlen=5))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5   0.005      1
##  maxlen target   ext
##       5  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 [0.01s].
## 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 ... [873 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
plot(rules)
```

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

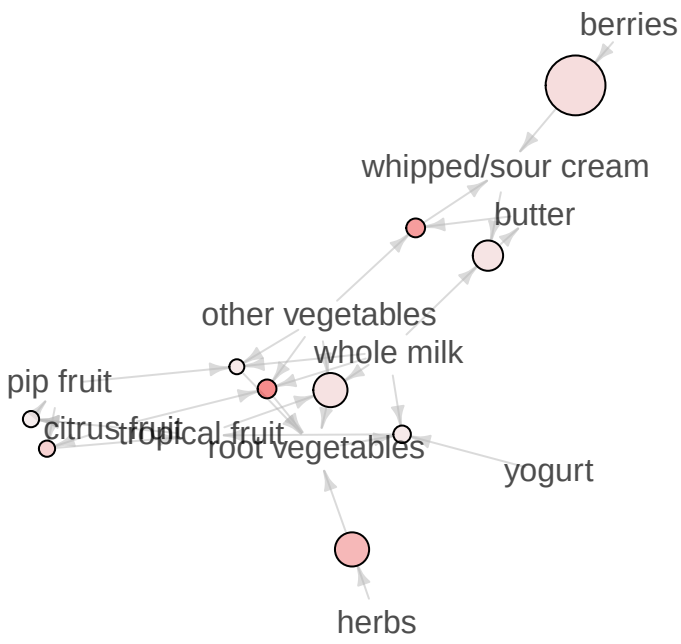


Here we had 873 rules and I'd like to trim down the rules to the ones that are more important.

We can see that

- berries, whipped/sour cream, and better is clustered together. These three are bought together
- citrus fruit, pip fruit, and tropical fruit is close.
- yogurt stands alone
- herbs stands alone
- whole milk, other vegetables, toot vegetables are in the center, which means that most people would buy them when they do grocery shopping.

```
subrules2 = head(sort(rules, by="lift"), 10)
plot(subrules2, method='graph')
```




```
inspect(subrules2)
```

```
##      lhs                     rhs                     support confidence     lift count
## [1]  {citrus fruit,
##       other vegetables,
##       whole milk}         => {root vegetables}    0.005795628  0.4453125 4.085493    57
## [2]  {butter,
##       other vegetables}   => {whipped/sour cream} 0.005795628  0.2893401 4.036397    57
## [3]  {herbs}             => {root vegetables}    0.007015760  0.4312500 3.956477    69
## [4]  {citrus fruit,
##       pip fruit}          => {tropical fruit}     0.005592272  0.4044118 3.854060    55
## [5]  {berries}           => {whipped/sour cream} 0.009049314  0.2721713 3.796886    89
## [6]  {other vegetables,
##       tropical fruit,
##       whole milk}         => {root vegetables}    0.007015760  0.4107143 3.768074    69
## [7]  {whipped/sour cream,
##       whole milk}         => {butter}             0.006710727  0.2082019 3.757185    66
## [8]  {root vegetables,
##       whole milk,
##       yogurt}             => {tropical fruit}     0.005693950  0.3916084 3.732043    56
## [9]  {other vegetables,
##       pip fruit,
##       whole milk}         => {root vegetables}    0.005490595  0.4060150 3.724961    54
## [10] {citrus fruit,
##       tropical fruit}     => {pip fruit}          0.005592272  0.2806122 3.709437    55
```