sta 380 Exercise 1

xueru rong

Flights at ABIA

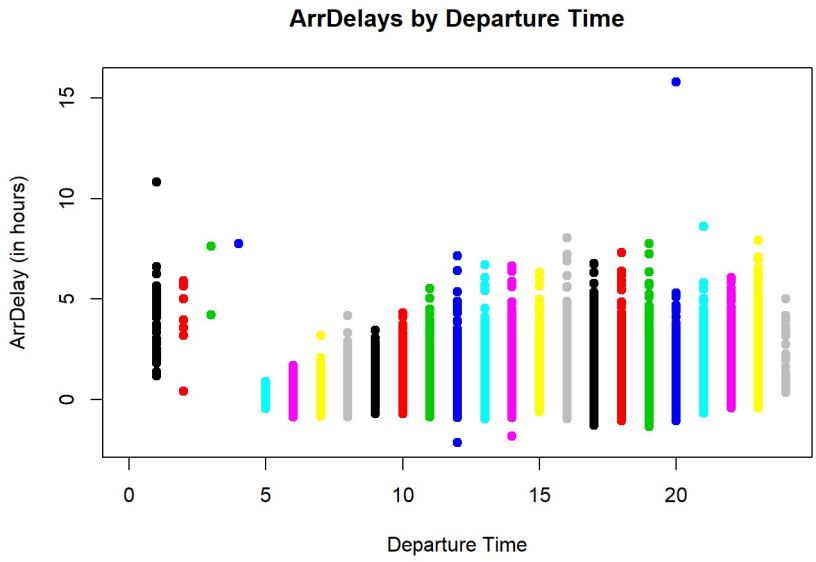
```
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 3.4.4
```

```
par(mfrow=c(2,2))
data=read.csv('ABIA.csv')
data$Austin=ifelse(data$Origin=='AUS',1,0)
```

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

```
data$hour=round(data$DepTime/100)
plot(data$hour, data$ArrDelay/60, pch=19, col=data$hour, xlab='Departure Time', ylab='ArrDelay (in hours)', mai
n='ArrDelays by Departure Time')
```



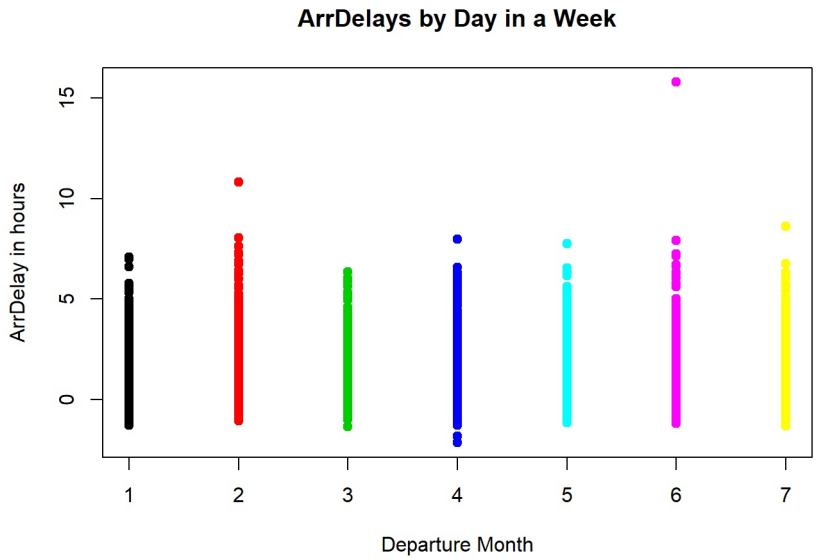

```
median_day=rep(NA,12)
std_day=rep(NA, 12)
for(i in 1:12){
  newsub=subset(data, hour==i)
  median_day[i]<-median(newsub$ArrDelay/60, na.rm=TRUE)
  std_day[i]<-sd(newsub$ArrDelay/60, na.rm=TRUE)

}
choose_day=data.frame(col1=median_day,col2=std_day)
choose_day[
  with(choose_day, order(choose_day$col2),decreasing=FALSE),
]
```

```
##           col1      col2
## 5  -0.01666667 0.1750175
## 6  -0.08333333 0.1908791
## 7  -0.06666667 0.2293273
## 8  -0.06666667 0.2724264
## 9  -0.03333333 0.3566187
## 11 -0.03333333 0.4266867
## 10 -0.03333333 0.4345897
## 12 -0.01666667 0.4677728
## 1   3.75000000 1.7487295
## 2   5.00000000 1.7973188
## 3   4.21666667 1.9726134
## 4   7.76666667        NA
```

What is the best day of week to fly to minimize delays?

```
plot(data$DayOfWeek, data$ArrDelay/60, pch=19, col=data$DayOfWeek, xlab='Departure Month', ylab='ArrDelay in ho
urs', main='ArrDelays by Day in a Week')
```



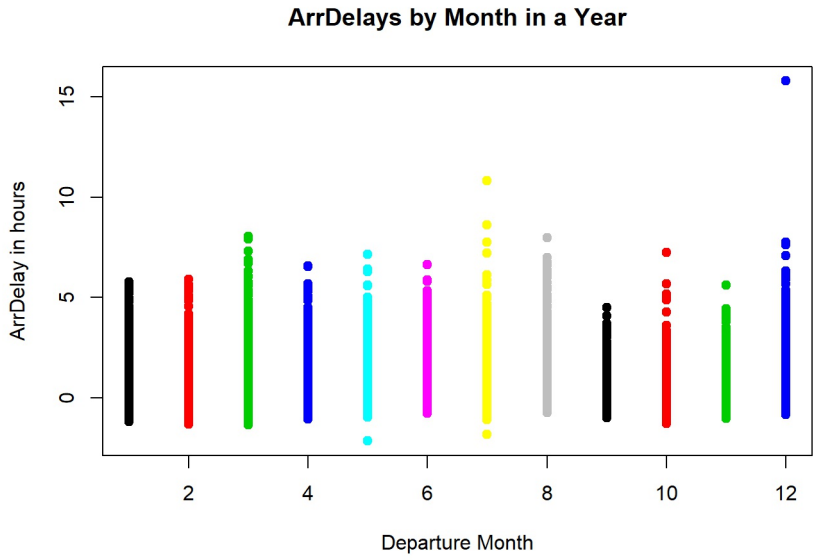

```
median_week=rep(NA,7)
std_week=rep(NA, 7)
for(i in 1:7){
  newsub=subset(data, DayOfWeek==i)
  median_week[i]<-median(newsub$ArrDelay/60, na.rm=TRUE)
  std_week[i]<-sd(newsub$ArrDelay/60, na.rm=TRUE)

}
choose_week=data.frame(col1=median_week,col2=std_week)
choose_week[
  with(choose_week, order(choose_week$col2),decreasing=FALSE),
]
```

```
##          col1      col2
## 3 -0.03333333 0.5035203
## 4 -0.01666667 0.5429525
## 6 -0.05000000 0.5654067
## 1 -0.01666667 0.5684814
## 5  0.00000000 0.5758807
## 2 -0.03333333 0.5830499
## 7 -0.03333333 0.5858247
```

What is the best month of year to fly to minimize delays?

```
plot(data$Month, data$ArrDelay/60, pch=19, col=data$Month, xlab='Departure Month', ylab='ArrDelay in hours', ma
in='ArrDelays by Month in a Year')
```



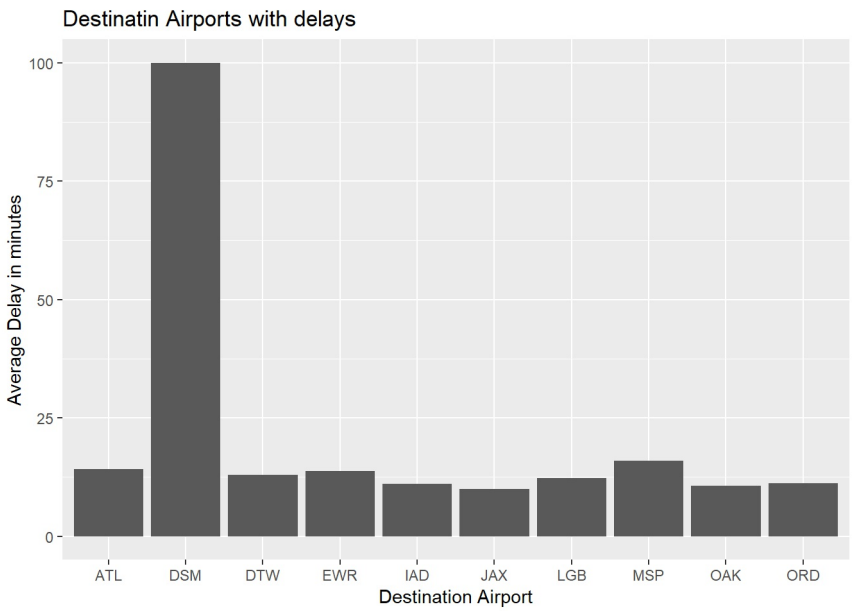
```
median_month=rep(NA,12)
std_month=rep(NA, 12)
for(i in 1:12){
  newsub=subset(data, Month==i)
  median_month[i]<-median(newsub$ArrDelay/60, na.rm=TRUE)
  std_month[i]<-sd(newsub$ArrDelay/60, na.rm=TRUE)

}
choose_month=data.frame(col1=median_month,col2=std_month)
choose_month[
  with(choose_month, order(choose_month$col2),decreasing=FALSE),
]
```

```
##           col1      col2
## 9  -0.06666667 0.3734475
## 10 -0.06666667 0.4066320
## 11 -0.06666667 0.4213849
## 1  -0.03333333 0.5306185
## 5   0.00000000 0.5317505
## 4  -0.01666667 0.5465534
## 2  -0.01666667 0.5549020
## 8  -0.01666667 0.6064537
## 7  -0.03333333 0.6136267
## 6   0.01666667 0.6183460
## 3   0.01666667 0.6417106
## 12  0.00000000 0.7068274
```

What are the bad airports to fly to?

```
airport=aggregate(data$ArrDelay, by=list(Category=data$Dest), FUN=mean, na.rm='True')
airport=airport[order(airport$x,decreasing = TRUE),]
airport=subset(airport,x>10)
ggplot(aes(x = Category, y = x), data = airport ) +geom_bar(stat = "identity") + xlab("Destination Airport") +
  ylab("Average Delay in minutes") + ggtitle("Destinatin Airports with delays")
```



From the plots, we can see that the best time to fly to minimize delay time is round 5 in the morning on Wednesday during September. \ In addition, we found that the Des Moines International Airport (DSM) airport has the most delay time as a destination airport from Austin Airport.\

Author attribution

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.4.4
```

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

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

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

```
library(foreach)
```

```
## Warning: package 'foreach' was built under R version 3.4.4
```

```
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 3.4.4
```

```
## randomForest 4.6-14
```

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

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

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

```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.4.4
```

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

```
## Loaded glmnet 2.0-16
```

```
library(tidyverse)
```

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

```
## v tibble  1.4.2     v purrr   0.2.4
## v tidyr   0.8.0     v dplyr   0.7.6
## v readr   1.1.1     v stringr 1.3.0
## v tibble  1.4.2     v forcats 0.2.0
```

```
## Warning: package 'dplyr' was built under R version 3.4.4
```

```
## -- Conflicts ------------------------------------------------------------------------------- tidyverse_confl
icts() --
## x purrr::accumulate()    masks foreach::accumulate()
## x NLP::annotate()        masks ggplot2::annotate()
## x dplyr::combine()       masks randomForest::combine()
## x tidyr::expand()        masks Matrix::expand()
## x dplyr::filter()        masks stats::filter()
## x dplyr::lag()           masks stats::lag()
## x randomForest::margin() masks ggplot2::margin()
## x purrr::when()          masks foreach::when()
```

```
set.seed("6")

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

train_list = Sys.glob('ReutersC50/C50train/*/*.txt')
test_list = Sys.glob('ReutersC50/C50test/*/*.txt')
file_list = c(train_list,test_list)

alldata = lapply(file_list, readerPlain)

filename = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., paste0, collapse = '') } %>%
  unlist

authorname = file_list %>%
{ strsplit(., '/', fixed=TRUE) } %>%
{ lapply(., tail, n=2) } %>%
{ lapply(., head, n=1) } %>%
  unlist

names(alldata) = filename

documents_raw = Corpus(VectorSource(alldata))

my_documents = documents_raw
my_documents = tm_map(my_documents, content_transformer(tolower))
```

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

```
my_documents = tm_map(my_documents, content_transformer(removeNumbers))
```

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

```
my_documents = tm_map(my_documents, content_transformer(removePunctuation))
```

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

```
my_documents = tm_map(my_documents, content_transformer(stripWhitespace))
```

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

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

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

```
DTM_all = DocumentTermMatrix(my_documents)
DTM_all = removeSparseTerms(DTM_all, 0.95)

tfidf_all = weightTfIdf(DTM_all)

X = as.matrix(tfidf_all)
summary(colSums(X))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    0.00    9.07   12.03   13.49   16.22   64.05
```

```
scrub_cols = which(colSums(X) == 0)
X = X[,-scrub_cols]

pca_train = prcomp(X, scale=TRUE)

X = pca_train$x[1:2500,1:100]
y = authorname[1:2500]

out1 = cv.glmnet(X, y, family='multinomial', type.measure="class")

lambda_hat = out1$lambda.min

paste("The chosen lambda is",lambda_hat)
```

```
## [1] "The chosen lambda is 0.000562636650667486"
```

```
predict.lasso = predict.cv.glmnet(out1,pca_train$x[2501:5000,1:100],s=lambda_hat)

predict.name = vector()
for (i in 1:2500){
  a=which(predict.lasso[i,,1] == max(predict.lasso[i,,1]))
  predict.name = c(predict.name,colnames(predict.lasso)[a])

}

result = predict.name == authorname[2501:5000]
paste("The correction ratio is",sum(result) / length(result))
```

```
## [1] "The correction ratio is 0.5832"
```

```
a=table(predict.name, authorname[2501:5000])
a[1:10,1:10]
```

```
##
## predict.name       AaronPressman AlanCrosby AlexanderSmith BenjaminKangLim
##   AaronPressman               34          0              2               0
##   AlanCrosby                   0         25              0               0
##   AlexanderSmith               0          0             25               0
##   BenjaminKangLim              0          0              0              17
##   BernardHickey                0          0              0               0
##   BradDorfman                  1          0              0               0
##   DarrenSchuettler             0          0              0               0
##   DavidLawder                  0          0              0               0
##   EdnaFernandes                0          0              1               0
##   EricAuchard                  0          0              0               0
##
## predict.name       BernardHickey BradDorfman DarrenSchuettler DavidLawder
##   AaronPressman                0           0                1           0
##   AlanCrosby                   0           0                0           0
##   AlexanderSmith               0           0                0           0
##   BenjaminKangLim              0           0                0           0
##   BernardHickey               36           0                0           0
##   BradDorfman                  0          32                0           9
##   DarrenSchuettler             0           0                9           0
##   DavidLawder                  0           2                0           6
##   EdnaFernandes                0           0                0           0
##   EricAuchard                  0           0                0           0
##
## predict.name       EdnaFernandes EricAuchard
##   AaronPressman                0           0
##   AlanCrosby                   0           0
##   AlexanderSmith               4           0
##   BenjaminKangLim              0           0
##   BernardHickey                0           0
##   BradDorfman                  0           2
##   DarrenSchuettler             0           0
##   DavidLawder                  0           0
##   EdnaFernandes               15           0
##   EricAuchard                  0          22
```

```
############
p=2500-1
mtryv = sqrt(p)
ntreev = 600

a=data.frame(authorname[1:2500],pca_train$x[1:2500,1:50])
colnames(a)[1]="author"

temprf = randomForest(author ~., data=a,mtry=mtryv,ntree=ntreev)
predict.name = predict(temprf,newdata=pca_train$x[2501:5000,1:50])
result = predict.name == authorname[2501:5000]
paste("The correction ratio is", sum(result) / length(result))
```

```
## [1] "The correction ratio is 0.4628"
```

```
t=table(predict.name, authorname[2501:5000])
t[1:10,1:10]
```

```
##
## predict.name       AaronPressman AlanCrosby AlexanderSmith BenjaminKangLim
##   AaronPressman               31          0              0               0
##   AlanCrosby                   0         18              0               0
##   AlexanderSmith               0          0             22               0
##   BenjaminKangLim              0          0              0              12
##   BernardHickey                0          0              0               0
##   BradDorfman                  1          0              0               0
##   DarrenSchuettler             2          0              2               0
##   DavidLawder                  0          0              0               0
##   EdnaFernandes                0          0              1               0
##   EricAuchard                  0          0              0               0
##
## predict.name       BernardHickey BradDorfman DarrenSchuettler DavidLawder
##   AaronPressman                2           1                0           0
##   AlanCrosby                   0           0                0           0
##   AlexanderSmith               0           0                0           0
##   BenjaminKangLim              0           0                0           0
##   BernardHickey               22           0                0           0
##   BradDorfman                  0          27                1           1
##   DarrenSchuettler             0           0                8           1
##   DavidLawder                  0           0                0           7
##   EdnaFernandes                0           0                0           0
##   EricAuchard                  0           0                0           1
##
## predict.name       EdnaFernandes EricAuchard
##   AaronPressman                0           0
##   AlanCrosby                   0           0
##   AlexanderSmith               1           0
##   BenjaminKangLim              0           0
##   BernardHickey                0           0
##   BradDorfman                  0           1
##   DarrenSchuettler             0           0
##   DavidLawder                  0           0
##   EdnaFernandes               14           0
##   EricAuchard                  0          12
```

```
k=0
i1=0
j1=0
for (i in 1:50) {
  for (j in 1:50) {
      if(i != j){
        if(t[i,j]>k){
          k=t[i,j]
          i1=i
          j1=j
        }
      }
  }
}

t[12:14,6:8]
```

```
##
## predict.name       BradDorfman DarrenSchuettler DavidLawder
##   GrahamEarnshaw             0                0           0
##   HeatherScoffield           0               35           0
##   JaneMacartney              0                0           0
```

From the result we can see that the Lasso regression model is better than random forest. DarrenSchuettler and HeatherScoffield are most easily be distinguished.