Assignment 2

Abhishek Sanghavi

Tuesday, August 18, 2015

Answer 1

Reading the data

```
delay_data<-read.csv("ABIA.csv")

require(quantmod)

## Loading required package: quantmod

## Warning: package 'quantmod' was built under R version 3.2.2

## Loading required package: xts

## Warning: package 'xts' was built under R version 3.2.2

## Loading required package: zoo

## Warning: package 'zoo' was built under R version 3.2.2

##
## Attaching package: 'zoo'
##
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
##
## Loading required package: TTR

## Warning: package 'TTR' was built under R version 3.2.2

## Version 0.4-0 included new data defaults. See ?getSymbols.

require(ggplot2)
```

```
## Loading required package: ggplot2

require(reshape2)

## Loading required package: reshape2

require(plyr)

## Loading required package: plyr

## Warning: package 'plyr' was built under R version 3.2.2

require(scales)

## Loading required package: scales

names(delay_data)

##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay"
```

Adding required dummy variables

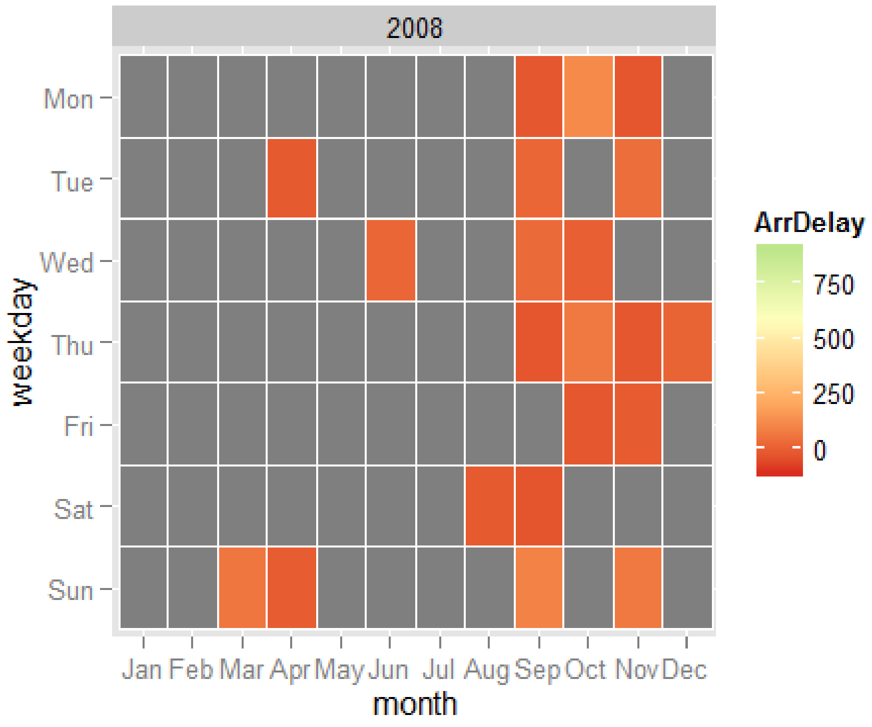
```
delay_data$month<-factor(delay_data$Month,levels=as.character(1:12),labels=c("Jan","
Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"),ordered=TRUE)
delay_data$weekday<-factor(delay_data$DayOfWeek,levels=rev(1:7),labels=rev(c("Mon","
Tue","Wed","Thu","Fri","Sat","Sun")),ordered=TRUE)
delay_data$yearmonthf<-factor(delay_data$Month)
```

Plotting

```
ggplot(delay_data, aes(month, weekday, fill = ArrDelay)) +

  geom_tile(colour = "white") +

  scale_fill_gradientn(colours = c("#D61818","#FFAE63","#FFFFBD","#B5E384")) +
  facet_wrap(~ Year, ncol = 1)
```



The plot does not take into consideration time delays if they are infrequent

It can be seen from the plot that the boxes with a lighter hue are the ones which had a larger time delay.

September, October and November have high time delays.

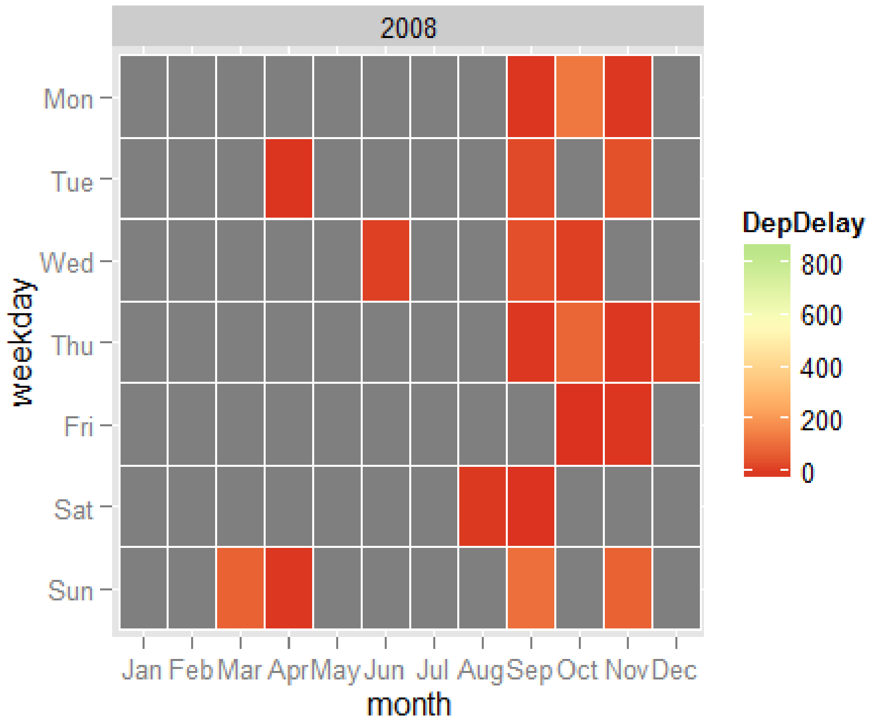
Plotting

```
ggplot(delay_data, aes(month, weekday, fill = DepDelay)) +

  geom_tile(colour = "white") +

  scale_fill_gradientn(colours = c("#D61818","#FFAE63","#FFFFBD","#B5E384")) +

  facet_wrap(~ Year, ncol = 1)
```



- The plot does not take into consideration time delays if they are infrequent
- It can be seen from the plot that the boxes with a lighter hue are the ones which had a larger time delay.
- September, October and November have high time delays.
- It can be seen that days of the month when Arrival delay is high Departure delay is also high.
- Sundays of the month March, September and November, Mondays and Thursdays of October have both high arrival and departure delays

Answer 2

Loading the library and the reader wrapper function

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 3.2.2

## Loading required package: NLP

## Warning: package 'NLP' was built under R version 3.2.2
```

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

Making a single corpus from 2 directories i.e Test and Train

- This is done to take care of the fact that there may be some words in the test data set which donot appear in the training data set
- After the data has been cleaned, the merged data (Train+Test) will be separated to use it in the models.

```
author_c50train = Sys.glob('../data/ReutersC50/C50train/*')
author_c50test = Sys.glob('../data/ReutersC50/C50test/*')

file_list = NULL
file_list1 = NULL
labels = NULL
labels1 = NULL
labels2 = NULL

for(author in author_c50train) {
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
}

for(author in author_c50test) {
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list1 = append(file_list1, files_to_add)
}

file_list2 = append(file_list,file_list1)
```

Adding meta data

```
for(author in author_c50train) {
  author_name = substring(author, first=29)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  labels1 = append(labels1, rep(author_name, length(files_to_add)))
}

for(author in author_c50test) {
  author_name = substring(author, first=28)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  labels2 = append(labels2, rep(author_name, length(files_to_add)))
}

labels <- unique(append(labels1, labels2))
```

Reading the files from the file list

```
all_docs = lapply(file_list2, readerPlain)
names(all_docs) = file_list2
names(all_docs) = sub('.txt', '', names(all_docs))

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

Preprocessing the data

```
my_corpus = tm_map(my_corpus, content_transformer(tolower)) # make everything lowerc
ase
my_corpus = tm_map(my_corpus, content_transformer(removeNumbers)) # remove numbers
my_corpus = tm_map(my_corpus, content_transformer(removePunctuation)) # remove punct
uation
```

```
my_corpus = tm_map(my_corpus, content_transformer(stripWhitespace)) ## remove excess
 white-space
my_corpus = tm_map(my_corpus, content_transformer(removeWords), stopwords("SMART"))

DTM = DocumentTermMatrix(my_corpus)
DTM # some basic summary statistics

## <<DocumentTermMatrix (documents: 5000, terms: 44234)>>
## Non-/sparse entries: 858721/220311279
## Sparsity           : 100%
## Maximal term length: 45
## Weighting          : term frequency (tf)

class(DTM)  # a special kind of sparse matrix format

## [1] "DocumentTermMatrix"    "simple_triplet_matrix"

DTM = removeSparseTerms(DTM, 0.975)
```

The sparsity of the Document Terms matrix is very high. We use the tm package function to reduce the sparsity by a defined parameter

Naive Bayes model

The 'weight vector' for each Author after Laplace smoothing is named weight. These are the word weights (for each author) that are multiplied with the word frequencies in the Test data to calculate the log probabilities.

```
X = as.matrix(DTM)

X_train <- X[1:2500,]


labels <- unique(labels)
```

```
smooth_count = 1/nrow(X_train)

for(i in 1:50)
{
  weight_label <- paste("w",labels[i], sep = "_")
  weight <- colSums(X_train[(50*i-49):(50*i),] + smooth_count)
  assign(weight_label, weight/sum(weight))
}
```

Predictions using Naive Bayes

```
X_test <- X[2501:5000,]

result = matrix(, nrow = 2500, ncol = 51)
for(i in 1:2500)
{ for(j in 1:50)
{
  weight_label <- paste("w",labels[j], sep = "_")

  result[i,j] = sum(X_test[i,]*log(get(weight_label)))
}
}
```

Assigning author based on calculated probablities

```
for (i in 1:2500)
{
  result[i,51] = which.max(result[i,])
}

result1 = NULL
result1 = cbind((rep(1:50, each=50)),result[,51])
result1$auth <- rep(1:50, each=50)

## Warning in result1$auth <- rep(1:50, each = 50): Coercing LHS to a list
```

```
result1$pred_auth <- result[,51]
```

Using the library caret for calculating prediction accuracy

Please note that to keep the length of the document short the confusion matrix has not been completely shown.

```
library(caret)

## Loading required package: lattice
## Loading required package: ggplot2
##
## Attaching package: 'ggplot2'
##
## The following object is masked from 'package:NLP':
##
##     annotate

confusionMatrix(result1$pred_auth,result1$auth)

## Confusion Matrix and Statistics
##

##             Reference
## Prediction  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19 20 21
##          1 42  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          2  0 25  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0
##          3  0  0 20  0  2  0  0  0  3  0  0  0  0  0  0  0  3  5  0  2  0
##          4  0  0  0 11  0  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0
##          5  0  0  0  0 27  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##          6  1  0  0  0  0 43  0  8  0  1  0  0  0  0  0  0  0  0  0  0  0
##          7  1  0  0  0  0  0 14  0  0  0  0  0  7  0  0  0  0  0  0  0  0
##          8  0  0  0  0  0  0  0  7  0  0  0  0  0  0  0  0  0  0  0  0  0
##          9  0  0  0  0  0  0  0  0 20  0  0  0  0  0  0  0  3  0  0  1  0
##         10  0  0  0  0  0  2  0  0  0 25  0  0  0  0  0  0  0  0  0  0  0
##         11  0  0  0  0  0  0  0  0  0  0 49  0  0  0  0  0  0  0  0  0  0
##         12  0  0  0  2  0  0  0  0  0  0  0 39  0  0  2  0  0  0  0  0  0
##         13  0  0  0  0  3  0 36  0  0  0  0  0 17  0  0  0  0  0  0  0  0
##         14  0  8  0  0  0  0  0  0  0  0  0  0  1 26  0  0  0  0 11  0  0
##         15  0  0  0 13  0  0  0  1  0  0  0  3 12  0 18  0  0  0  0  0  0
##         16  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0 50  0  0  0  0  0
##         17  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0 33  0  0  2  0
##         18  1  0 28  0  1  0  0  1  0  0  0  0  0  0  0  0  1 39  0  0  0
##         19  0 17  0  0  0  0  0  0  0  0  0  0  3 23  0  0  0  0 32  0  0
##         20  0  0  1  0  0  0  0  0  1  0  0  0  0  0  0  0  2  0  0 37  0
##         21  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  2  0 47
##         22  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  1  0
##         23  0  0  0  0  0  1  0  3  0  3  0  0  0  0  0  0  0  0  0  0  0
##         24  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         25  0  0  0  0  0  0  0  0  5  2  0  0  0  0  0  0  0  0  0  1  0
##         26  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0
##         27  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         28  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         29  0  0  0  0  1  0  0  0  0  0  0  0  1  0  0  0  0  0  0  1  0
##         30  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         31  0  0  0  0  3  0  0  0  1  0  0  0  4  0  0  0  0  0  0  0  0
##         32  0  0  0  0  0  1  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0
##         33  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2
##         34  0  0  0  0  2  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0
##         35  0  0  0  3  0  0  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0
##         36  0  0  0  0  0  0  0  0  0 10  0  0  0  0  0  0  0  0  0  0  0
##         37  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         38  0  0  0  3  0  0  0  0  0  0  0  0  1  0  0  0  0  0  1  0  0
##         39  0  0  0  0  0  0  0  0  4  0  0  0  0  0  0  0  0  0  1  0  0
##         40  0  0  0  0  0  1  0  4  0  1  0  0  0  0  0  0  0  0  0  0  0
##         41  0  0  0  0  0  0  0  0  0  1  0  0  1  0  0  0  0  0  0  0  0
##         42  3  0  1  0  0  1  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0
##         43  0  0  0  0  4  0  0  0  0  0  1  4  0  0  0  0  0  1  0  0  0
##         44  0  0  0  7  0  0  0  0  0  0  0  1  1  0 14  0  0  0  1  0  1
##         45  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  2  0
##         46  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
##         47  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0
##         48  0  0  0  0  2  0  0  0  6  0  0  0  0  0  0  0  8  1  0  3  0
##         49  0  0  0  0  0  1  0 23  6  0  0  0  0  0  0  0  0  0  1  0  0
##         50  2  0  0 11  0  0  0  0  0  0  0  2  2  0  2  0  0  0  0  0  0
## Overall Statistics
##
##                Accuracy : 0.6024
##                  95% CI : (0.5829, 0.6217)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.5943
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.8400   0.5000   0.4000   0.2200   0.5400   0.8600
## Specificity            0.9963   0.9996   0.9869   0.9939   0.9955   0.9861
## Pos Pred Value         0.8235   0.9615   0.3846   0.4231   0.7105   0.5584
## Neg Pred Value         0.9967   0.9899   0.9877   0.9842   0.9907   0.9971
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0168   0.0100   0.0080   0.0044   0.0108   0.0172
## Detection Prevalence   0.0204   0.0104   0.0208   0.0104   0.0152   0.0308
## Balanced Accuracy      0.9182   0.7498   0.6935   0.6069   0.7678   0.9231
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.2800   0.1400   0.4000    0.5000    0.9800
## Specificity            0.9947   0.9947   0.9951    0.9869    0.9988
## Pos Pred Value         0.5185   0.3500   0.6250    0.4386    0.9423
## Neg Pred Value         0.9854   0.9827   0.9878    0.9898    0.9996
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0056   0.0028   0.0080    0.0100    0.0196
## Detection Prevalence   0.0108   0.0080   0.0128    0.0228    0.0208
## Balanced Accuracy      0.6373   0.5673   0.6976    0.7435    0.9894
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.7800    0.3400    0.5200    0.3600    1.0000
## Specificity             0.9931    0.9820    0.9918    0.9633    0.9984
## Pos Pred Value          0.6964    0.2787    0.5652    0.1667    0.9259
## Neg Pred Value          0.9955    0.9865    0.9902    0.9866    1.0000
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0156    0.0068    0.0104    0.0072    0.0200
## Detection Prevalence    0.0224    0.0244    0.0184    0.0432    0.0216
## Balanced Accuracy       0.8865    0.6610    0.7559    0.6616    0.9992
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.6600    0.7800    0.6400    0.7400    0.9400
## Specificity             0.9967    0.9824    0.9824    0.9931    0.9959
## Pos Pred Value          0.8049    0.4756    0.4267    0.6852    0.8246
## Neg Pred Value          0.9931    0.9955    0.9926    0.9947    0.9988
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0132    0.0156    0.0128    0.0148    0.0188
## Detection Prevalence    0.0164    0.0328    0.0300    0.0216    0.0228
## Balanced Accuracy       0.8284    0.8812    0.8112    0.8665    0.9680
##                      Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity             0.7400    0.5200    0.5600    0.6800    0.6400
## Specificity             0.9927    0.9873    0.9943    0.9918    0.9984
## Pos Pred Value          0.6727    0.4561    0.6667    0.6296    0.8889
## Neg Pred Value          0.9947    0.9902    0.9910    0.9935    0.9927
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0148    0.0104    0.0112    0.0136    0.0128
## Detection Prevalence    0.0220    0.0228    0.0168    0.0216    0.0144
## Balanced Accuracy       0.8663    0.7537    0.7771    0.8359    0.8192
##                      Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity             0.6200    0.7800    0.9800    0.6000    0.4200
## Specificity             1.0000    0.9988    0.9959    0.9943    0.9951
## Pos Pred Value          1.0000    0.9286    0.8305    0.6818    0.6364
## Neg Pred Value          0.9923    0.9955    0.9996    0.9919    0.9882
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0156    0.0196    0.0120    0.0084
## Detection Prevalence    0.0124    0.0168    0.0236    0.0176    0.0132
## Balanced Accuracy       0.8100    0.8894    0.9880    0.7971    0.7076
##                      Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity             0.5000    0.8400    0.7800    0.2600    0.8200
## Specificity             0.9967    0.9992    0.9947    0.9939    0.9931
## Pos Pred Value          0.7576    0.9545    0.7500    0.4643    0.7069
## Neg Pred Value          0.9899    0.9967    0.9955    0.9850    0.9963
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0100    0.0168    0.0156    0.0052    0.0164
## Detection Prevalence    0.0132    0.0176    0.0208    0.0112    0.0232
## Balanced Accuracy       0.7484    0.9196    0.8873    0.6269    0.9065
##                      Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity            0.6200    0.6600    0.6800    0.8000    0.7600
## Specificity            0.9943    0.9829    0.9955    0.9976    0.9988
## Pos Pred Value         0.6889    0.4400    0.7556    0.8696    0.9268
## Neg Pred Value         0.9923    0.9930    0.9935    0.9959    0.9951
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0124    0.0132    0.0136    0.0160    0.0152
## Detection Prevalence   0.0180    0.0300    0.0180    0.0184    0.0164
## Balanced Accuracy      0.8071    0.8214    0.8378    0.8988    0.8794
##                     Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity            0.6400    0.5400    0.2600    0.5600    0.4000
## Specificity            0.9873    0.9865    0.9816    0.9951    0.9890
## Pos Pred Value         0.5079    0.4500    0.2241    0.7000    0.4255
## Neg Pred Value         0.9926    0.9906    0.9848    0.9911    0.9878
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0128    0.0108    0.0052    0.0112    0.0080
## Detection Prevalence   0.0252    0.0240    0.0232    0.0160    0.0188
## Balanced Accuracy      0.8137    0.7633    0.6208    0.7776    0.6945
##                     Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity            0.5200    0.7800    0.4000    0.3400
## Specificity            0.9902    0.9902    0.9849    0.9865
## Pos Pred Value         0.5200    0.6190    0.3509    0.3400
## Neg Pred Value         0.9902    0.9955    0.9877    0.9865
## Prevalence             0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0104    0.0156    0.0080    0.0068
## Detection Prevalence   0.0200    0.0252    0.0228    0.0200
## Balanced Accuracy      0.7551    0.8851    0.6924    0.6633
```

- As can be seen the model above gives an accuracy of 60.24%
- Defining the threshold of predicting an author unambiguosly at 70%,
- Authors 4, 7, 8, 13, 15, 35, 44, 50 are difficult to very accurately predict. author 8 is the most difficult to predict with an accuracy of only about 55%
- It is also seen from the matrix that authors 13 and 15 are difficult to tell from each other as can be seen from the confusion matrix
- It is also be seen that author 8 David Lawder has a prediction accuracy that is very low because of author 8 not being distinguishable from author 49 Todd Nissen . Most of the articles

written by author 8 are being attributed to author 49 , this is because they write about similar topics.

Data preparation for other models

Creating author labels

```
author_name=rep(rep(1:50,each=50),2)
```

Converting the document term matrix into a dataframe

```
author = as.data.frame(X)
```

Giving each column a column name

```
colnames(author) = make.names(colnames(author))
```

Adding dependent variable as a factor variable in the dataframe

```
author$author_name = author_name
author$author_name=as.factor(author$author_name)
```

Dividing data into test and train

```
author_train=author[1:2500,]
author_test=author[2501:5000,]
```

Using regression tree for the prediction

```
library(rpart)
authorcart=rpart(author_name~.,data=author_train,method="class",cp=0.0011)
predictauthor=predict(authorcart,newdata=author_test,type="class")
confusionMatrix(predictauthor,author_test$author_name)

## Confusion Matrix and Statistics
##
## Overall Statistics
##
##                Accuracy : 0.4172
##                  95% CI : (0.3978, 0.4368)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.4053
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.5200   0.4200   0.0800   0.3000   0.3400   0.6600
## Specificity            0.9943   0.9967   0.9931   0.9698   0.9967   0.9127
## Pos Pred Value         0.6500   0.7241   0.1905   0.1685   0.6800   0.1336
## Neg Pred Value         0.9902   0.9883   0.9814   0.9855   0.9867   0.9925
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0104   0.0084   0.0016   0.0060   0.0068   0.0132
## Detection Prevalence   0.0160   0.0116   0.0084   0.0356   0.0100   0.0988
## Balanced Accuracy      0.7571   0.7084   0.5365   0.6349   0.6684   0.7863
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.3000   0.0800   0.0800    0.1400    0.7800
## Specificity            0.9853   0.9914   0.9878    0.9882    0.9992
## Pos Pred Value         0.2941   0.1600   0.1176    0.1944    0.9512
## Neg Pred Value         0.9857   0.9814   0.9813    0.9825    0.9955
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0060   0.0016   0.0016    0.0028    0.0156
## Detection Prevalence   0.0204   0.0100   0.0136    0.0144    0.0164
## Balanced Accuracy      0.6427   0.5357   0.5339    0.5641    0.8896
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.5400    0.2200    0.6000    0.2400    0.7800
## Specificity             0.9935    0.9857    0.9878    0.9955    0.9967
## Pos Pred Value          0.6279    0.2391    0.5000    0.5217    0.8298
## Neg Pred Value          0.9906    0.9841    0.9918    0.9847    0.9955
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0108    0.0044    0.0120    0.0048    0.0156
## Detection Prevalence    0.0172    0.0184    0.0240    0.0092    0.0188
## Balanced Accuracy       0.7667    0.6029    0.7939    0.6178    0.8884
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.4600    0.4400    0.5000    0.2400    0.7400
## Specificity            0.9947    0.9653    0.9882    0.9886    0.9841
## Pos Pred Value         0.6389    0.2056    0.4630    0.3000    0.4868
## Neg Pred Value         0.9890    0.9883    0.9898    0.9846    0.9946
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0092    0.0088    0.0100    0.0048    0.0148
## Detection Prevalence   0.0144    0.0428    0.0216    0.0160    0.0304
## Balanced Accuracy      0.7273    0.7027    0.7441    0.6143    0.8620
##                     Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity            0.5400    0.4400    0.5800    0.2000    0.4800
## Specificity            0.9927    0.9763    0.9894    0.9971    0.9914
## Pos Pred Value         0.6000    0.2750    0.5273    0.5882    0.5333
## Neg Pred Value         0.9906    0.9884    0.9914    0.9839    0.9894
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0108    0.0088    0.0116    0.0040    0.0096
## Detection Prevalence   0.0180    0.0320    0.0220    0.0068    0.0180
## Balanced Accuracy      0.7663    0.7082    0.7847    0.5986    0.7357
##                     Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity            0.5400    0.8000    0.7400    0.6200    0.6400
## Specificity            0.9996    0.9959    0.9996    0.9898    0.9935
## Pos Pred Value         0.9643    0.8000    0.9737    0.5536    0.6667
## Neg Pred Value         0.9907    0.9959    0.9947    0.9922    0.9927
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0108    0.0160    0.0148    0.0124    0.0128
## Detection Prevalence   0.0112    0.0200    0.0152    0.0224    0.0192
## Balanced Accuracy      0.7698    0.8980    0.8698    0.8049    0.8167
##                     Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity            0.2800    0.8200    0.4000    0.3800    0.2200
## Specificity            0.9947    0.9906    0.9841    0.9898    0.9910
## Pos Pred Value         0.5185    0.6406    0.3390    0.4318    0.3333
## Neg Pred Value         0.9854    0.9963    0.9877    0.9874    0.9842
## Prevalence             0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate         0.0056    0.0164    0.0080    0.0076    0.0044
## Detection Prevalence   0.0108    0.0256    0.0236    0.0176    0.0132
## Balanced Accuracy      0.6373    0.9053    0.6920    0.6849    0.6055
##                     Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity            0.1600    0.3400    0.6000    0.5400    0.4800
## Specificity            0.9980    0.9894    0.9902    0.9939    0.9918
## Pos Pred Value         0.6154    0.3953    0.5556    0.6429    0.5455
## Neg Pred Value          0.9831    0.9866    0.9918    0.9906    0.9894
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0032    0.0068    0.0120    0.0108    0.0096
## Detection Prevalence    0.0052    0.0172    0.0216    0.0168    0.0176
## Balanced Accuracy       0.5790    0.6647    0.7951    0.7669    0.7359
##                      Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity             0.3600    0.5000    0.0600    0.1600    0.3800
## Specificity             0.9820    0.9841    0.9890    0.9959    0.9824
## Pos Pred Value          0.2903    0.3906    0.1000    0.4444    0.3065
## Neg Pred Value          0.9869    0.9897    0.9810    0.9831    0.9873
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0072    0.0100    0.0012    0.0032    0.0076
## Detection Prevalence    0.0248    0.0256    0.0120    0.0072    0.0248
## Balanced Accuracy       0.6710    0.7420    0.5245    0.5780    0.6812
##                      Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity             0.3600    0.2600    0.3800    0.1400
## Specificity             0.9865    0.9837    0.9845    0.9833
## Pos Pred Value          0.3529    0.2453    0.3333    0.1458
## Neg Pred Value          0.9869    0.9849    0.9873    0.9825
## Prevalence              0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0072    0.0052    0.0076    0.0028
## Detection Prevalence    0.0204    0.0212    0.0228    0.0192
## Balanced Accuracy       0.6733    0.6218    0.6822    0.5616
```

- The prediction accuracy of the model is 41% which is much lower than the naive bayes model
- To improve accuracy we try a randomForest model instead of a tree.

Using randomForest for the prediction

```
library(randomForest)

## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.

authorforest=randomForest(author_name~.,data=author_train)
predictauthor=predict(authorforest,newdata=author_test)
confusionMatrix(predictauthor,author_test$author_name)
```

```
## Confusion Matrix and Statistics
##
## Overall Statistics
##
##                Accuracy : 0.6224
##                  95% CI : (0.6031, 0.6415)
##     No Information Rate : 0.02
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.6147
##  Mcnemar's Test P-Value : NA
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity            0.9200   0.7000   0.1800   0.4600   0.4800   0.5600
## Specificity            0.9927   0.9967   0.9992   0.9804   0.9955   0.9914
## Pos Pred Value         0.7188   0.8140   0.8182   0.3239   0.6857   0.5714
## Neg Pred Value         0.9984   0.9939   0.9835   0.9889   0.9895   0.9910
## Prevalence             0.0200   0.0200   0.0200   0.0200   0.0200   0.0200
## Detection Rate         0.0184   0.0140   0.0036   0.0092   0.0096   0.0112
## Detection Prevalence   0.0256   0.0172   0.0044   0.0284   0.0140   0.0196
## Balanced Accuracy      0.9563   0.8484   0.5896   0.7202   0.7378   0.7757
##                      Class: 7 Class: 8 Class: 9 Class: 10 Class: 11
## Sensitivity            0.2800   0.1600   0.3600    0.3200    1.0000
## Specificity            0.9808   0.9959   0.9951    0.9980    0.9967
## Pos Pred Value         0.2295   0.4444   0.6000    0.7619    0.8621
## Neg Pred Value         0.9852   0.9831   0.9870    0.9863    1.0000
## Prevalence             0.0200   0.0200   0.0200    0.0200    0.0200
## Detection Rate         0.0056   0.0032   0.0072    0.0064    0.0200
## Detection Prevalence   0.0244   0.0072   0.0120    0.0084    0.0232
## Balanced Accuracy      0.6304   0.5780   0.6776    0.6590    0.9984
##                      Class: 12 Class: 13 Class: 14 Class: 15 Class: 16
## Sensitivity             0.9400    0.3600    0.5000    0.2000    1.0000
## Specificity             0.9918    0.9837    0.9939    0.9943    0.9971
## Pos Pred Value          0.7015    0.3103    0.6250    0.4167    0.8772
## Neg Pred Value          0.9988    0.9869    0.9898    0.9838    1.0000
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0188    0.0072    0.0100    0.0040    0.0200
## Detection Prevalence    0.0268    0.0232    0.0160    0.0096    0.0228
## Balanced Accuracy       0.9659    0.6718    0.7469    0.5971    0.9986
##                      Class: 17 Class: 18 Class: 19 Class: 20 Class: 21
## Sensitivity             0.8200    0.5600    0.6400    0.7200    0.9200
## Specificity             0.9951    0.9820    0.9869    0.9922    0.9922
## Pos Pred Value          0.7736    0.3889    0.5000    0.6545    0.7077
## Neg Pred Value          0.9963    0.9909    0.9926    0.9943    0.9984
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0164    0.0112    0.0128    0.0144    0.0184
## Detection Prevalence    0.0212    0.0288    0.0256    0.0220    0.0260
## Balanced Accuracy       0.9076    0.7710    0.8135    0.8561    0.9561
##                      Class: 22 Class: 23 Class: 24 Class: 25 Class: 26
## Sensitivity             0.8000    0.5800    0.5800    0.4800    0.8400
## Specificity             0.9959    0.9906    0.9927    0.9996    0.9865
## Pos Pred Value          0.8000    0.5577    0.6170    0.9600    0.5600
## Neg Pred Value          0.9959    0.9914    0.9914    0.9895    0.9967
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0160    0.0116    0.0116    0.0096    0.0168
## Detection Prevalence    0.0200    0.0208    0.0188    0.0100    0.0300
## Balanced Accuracy       0.8980    0.7853    0.7863    0.7398    0.9133
##                      Class: 27 Class: 28 Class: 29 Class: 30 Class: 31
## Sensitivity             0.6200    0.7800    0.9400    0.9800    0.8000
## Specificity             0.9992    0.9992    0.9980    0.9869    0.9918
## Pos Pred Value          0.9394    0.9512    0.9038    0.6049    0.6667
## Neg Pred Value          0.9923    0.9955    0.9988    0.9996    0.9959
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0124    0.0156    0.0188    0.0196    0.0160
## Detection Prevalence    0.0132    0.0164    0.0208    0.0324    0.0240
## Balanced Accuracy       0.8096    0.8896    0.9690    0.9835    0.8959
##                      Class: 32 Class: 33 Class: 34 Class: 35 Class: 36
## Sensitivity             0.4200    0.9200    0.6400    0.4200    0.8800
## Specificity             1.0000    0.9967    0.9914    0.9943    0.9894
## Pos Pred Value          1.0000    0.8519    0.6038    0.6000    0.6286
## Neg Pred Value          0.9883    0.9984    0.9926    0.9882    0.9975
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0084    0.0184    0.0128    0.0084    0.0176
## Detection Prevalence    0.0084    0.0216    0.0212    0.0140    0.0280
## Balanced Accuracy       0.7100    0.9584    0.8157    0.7071    0.9347
##                      Class: 37 Class: 38 Class: 39 Class: 40 Class: 41
## Sensitivity             0.7000    0.8200    0.4400    0.9200    0.7800
## Specificity             0.9910    0.9849    0.9988    0.9914    0.9963
## Pos Pred Value          0.6140    0.5256    0.8800    0.6866    0.8125
## Neg Pred Value          0.9939    0.9963    0.9887    0.9984    0.9955
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0140    0.0164    0.0088    0.0184    0.0156
## Detection Prevalence    0.0228    0.0312    0.0100    0.0268    0.0192
## Balanced Accuracy       0.8455    0.9024    0.7194    0.9557    0.8882
##                      Class: 42 Class: 43 Class: 44 Class: 45 Class: 46
## Sensitivity             0.4200    0.4600    0.2600    0.8800    0.5200
## Specificity             0.9939    0.9910    0.9878    0.9939    0.9865
## Pos Pred Value          0.5833    0.5111    0.3023    0.7458    0.4407
## Neg Pred Value          0.9882    0.9890    0.9849    0.9975    0.9902
## Prevalence              0.0200    0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0084    0.0092    0.0052    0.0176    0.0104
## Detection Prevalence    0.0144    0.0180    0.0172    0.0236    0.0236
## Balanced Accuracy       0.7069    0.7255    0.6239    0.9369    0.7533
##                      Class: 47 Class: 48 Class: 49 Class: 50
## Sensitivity             0.5800    0.7400    0.5400    0.3000
## Specificity             0.9914    0.9927    0.9829    0.9882
## Pos Pred Value          0.5800    0.6727    0.3913    0.3409
## Neg Pred Value          0.9914    0.9947    0.9905    0.9857
## Prevalence              0.0200    0.0200    0.0200    0.0200
## Detection Rate          0.0116    0.0148    0.0108    0.0060
## Detection Prevalence    0.0200    0.0220    0.0276    0.0176
## Balanced Accuracy       0.7857    0.8663    0.7614    0.6441
```

- The prediction accuracy of the model is 62.24%
- Defining the threshold of predicting an author unambiguosly at 70%,
- Authors 3, 7, 8, 9, 10, 13, 15 , 44, 50 are difficult to very accurately predict. author 3, 8 are the most difficult to predict.

Answer 3

```
library(arules)
library(reshape)
```

- The data is read and the transaction numbers is added to the list
- The data frame is reshaped get all the variables in the row(unstacked) format
- NA's are omittted and the transaction number is changed to a categorical variable

```
colmax= max(count.fields("../data/groceries.txt",sep=','))
groceries <- read.csv("../data/groceries.txt", header = FALSE,col.names = paste0("V",
seq_len(colmax)),fill = TRUE)



row_series<-1:nrow(groceries)
groceries<-cbind(row_series,groceries)

groceries<-melt(groceries,id=c("row_series"))
groceries<-groceries[order(groceries$row_series),]

groceries[groceries==""] <- NA
groceries <- na.omit(groceries)



groceries$row_series <- factor(groceries$row_series)
```

- Creating a list of baskets: vectors of items by trasaction
- Duplicates are removed and cast as a special arules "transaction class"

```
# First split data into a list of items for each transaction
groceries <- split(x=groceries$value, f=groceries$row_series)

## Removing duplicates
groceries <- lapply(groceries, unique)



groceries_trans <- as(groceries, "transactions")
```

```r
# Using the 'apriori' algorithm, examine the rules with support > .05 & confidence >.
5
groceries_assoc_rules <- apriori(groceries_trans, parameter=list(support=.05, confid
ence=.5, maxlen=3))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE    0.05      1      3
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [28 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [0 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Look at the output
inspect(groceries_assoc_rules)
```

```
## NULL
```

- We notice that we do not get any set for support of .05.

- Reducing it to .01 still keeping the confidence of 0.5 which would give us a better quality of the rules .

-

```r
groceries_assoc_rules1 <- apriori(groceries_trans, parameter=list(support=.01, confi
dence=.5, maxlen=3))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE    0.01      1      3
##  target   ext
##   rules FALSE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## apriori - find association rules with the apriori algorithm
## version 4.21 (2004.05.09)        (c) 1996-2004   Christian Borgelt
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
# Look at the output
inspect(groceries_assoc_rules1)
```

```
##    lhs                        rhs                      support confidence     lift
## 1  {curd,
##     yogurt}                => {whole milk}       0.01006609  0.5823529 2.279125
## 2  {butter,
##     other vegetables}      => {whole milk}       0.01148958  0.5736041 2.244885
## 3  {domestic eggs,
##     other vegetables}      => {whole milk}       0.01230300  0.5525114 2.162336
## 4  {whipped/sour cream,
##     yogurt}                => {whole milk}       0.01087951  0.5245098 2.052747
## 5  {other vegetables,
##     whipped/sour cream} => {whole milk}       0.01464159  0.5070423 1.984385
## 6  {other vegetables,
##     pip fruit}             => {whole milk}       0.01352313  0.5175097 2.025351
## 7  {citrus fruit,
##     root vegetables}    => {other vegetables} 0.01037112  0.5862069 3.029608
## 8  {root vegetables,
##     tropical fruit}     => {other vegetables} 0.01230300  0.5845411 3.020999
## 9  {root vegetables,
##     tropical fruit}     => {whole milk}       0.01199797  0.5700483 2.230969
## 10 {tropical fruit,
##     yogurt}             => {whole milk}       0.01514997  0.5173611 2.024770
## 11 {root vegetables,
##     yogurt}             => {other vegetables} 0.01291307  0.5000000 2.584078
## 12 {root vegetables,
##     yogurt}             => {whole milk}       0.01453991  0.5629921 2.203354
## 13 {rolls/buns,
##     root vegetables}    => {other vegetables} 0.01220132  0.5020921 2.594890
## 14 {rolls/buns,
##     root vegetables}    => {whole milk}       0.01270971  0.5230126 2.046888
## 15 {other vegetables,
##     yogurt}             => {whole milk}       0.02226741  0.5128806 2.007235
```

- Whole milk being bought with {Curd, Yogurt} or {Butter, Yogurt} or `{whipped/sour cream, yogurt}` is understandable as they are all dairy products.
- Other Vegetables' which is a category has various kind of vegetables is bought with vegetables and fruits. Here we can see that these are bought with {Citrus Fruit, Root vegetables},{Root Vegetables, Tropical Fruit}. There are other categories like {rolls/buns, root vegetables} but other vegetables is only bought when atleast one of the other things bought is a fruit or a vegetable.
- People who do their weekly shop at the retail store tend to buy everything together which will explain the association of dairy products like Milk to categories like vegetables, fruits, buns etc.