STA380 Homework 2

Francisco Sananez

August 17, 2015

Flights at ABIA

To begin the "ABIA.csv" data was loaded and saved as flights. We used the *head()* function to get a feel of how the data is portrayed. We can identify specific flights that are either arriving to or departing from Austin by looking at the *Origin* and *Dest* columns.

```
library(ggplot2)
library(plyr)
flights <- read.csv("ABIA.csv")
flights <- subset(flights, flights$Dest != "DSM")
head(flights)
```

```
##   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime
CRSArrTime
## 1 2008     1          1         2     120       1935     309
2130
## 2 2008     1          1         2     555        600     826
835
## 3 2008     1          1         2     600        600     728
729
## 4 2008     1          1         2     601        605     727
750
## 5 2008     1          1         2     601        600     654
700
## 6 2008     1          1         2     636        645     934
932
##   UniqueCarrier FlightNum TailNum ActualElapsedTime CRSElapsedTime
AirTime
## 1            9E      5746  84129E                109             115
88
## 2            AA      1614  N438AA                151             155
133
## 3            YV      2883  N922FJ                148             149
125
## 4            9E      5743  89189E                 86             105
70
## 5            AA      1157  N4XAAA                 53              60
38
## 6            NW      1674   N967N                178             167
145
```

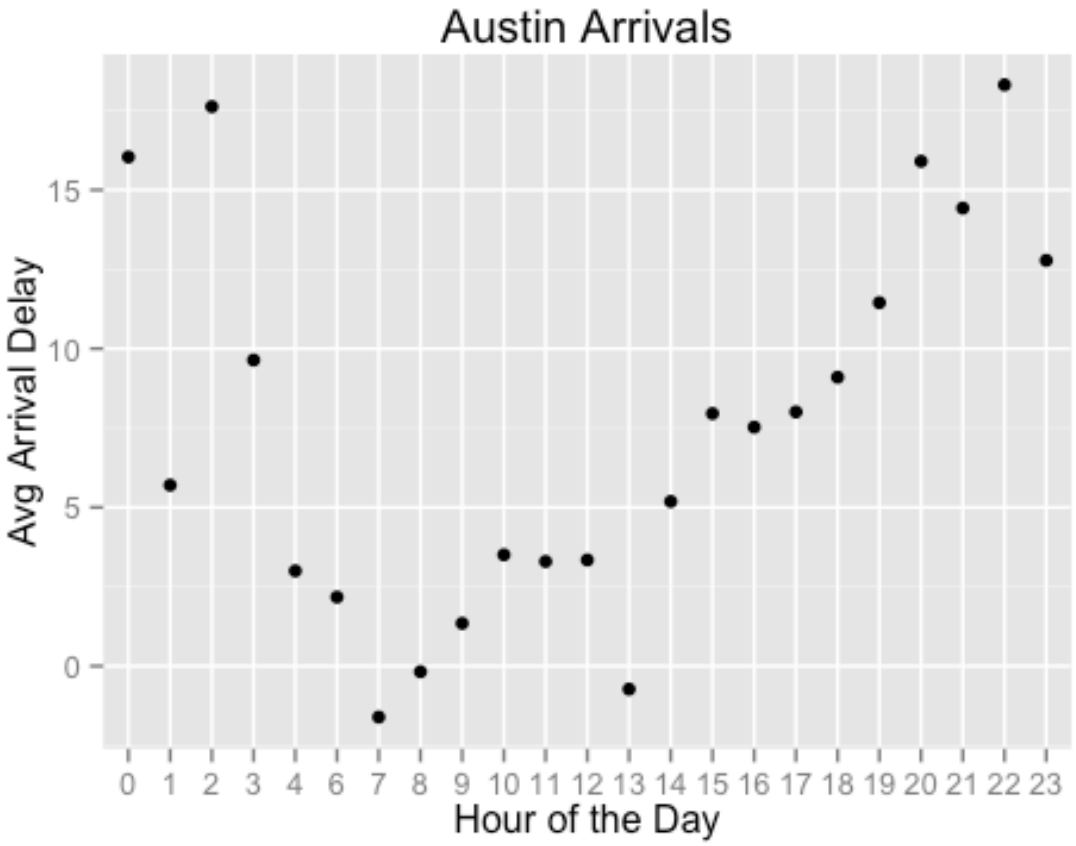
```
##   ArrDelay DepDelay Origin Dest Distance TaxiIn TaxiOut Cancelled
## 1      339      345    MEM  AUS      559      3      18         0
## 2       -9       -5    AUS  ORD      978      7      11         0
## 3       -1        0    AUS  PHX      872      7      16         0
## 4      -23       -4    AUS  MEM      559      4      12         0
## 5       -6        1    AUS  DFW      190      5      10         0
## 6        2       -9    AUS  MSP     1042     11      22         0
##   CancellationCode Diverted CarrierDelay WeatherDelay NASDelay
## 1                        0          339            0        0
## 2                        0           NA           NA       NA
## 3                        0           NA           NA       NA
## 4                        0           NA           NA       NA
## 5                        0           NA           NA       NA
## 6                        0           NA           NA       NA
##   SecurityDelay LateAircraftDelay
## 1             0                 0
## 2            NA                NA
## 3            NA                NA
## 4            NA                NA
## 5            NA                NA
## 6            NA                NA
```

In order to get a better view of the data, it was broken into two sets. The arriving flights set as *arrflights* and the departing flights set as *depflights*.

Arriving Flights

Hourly Delays

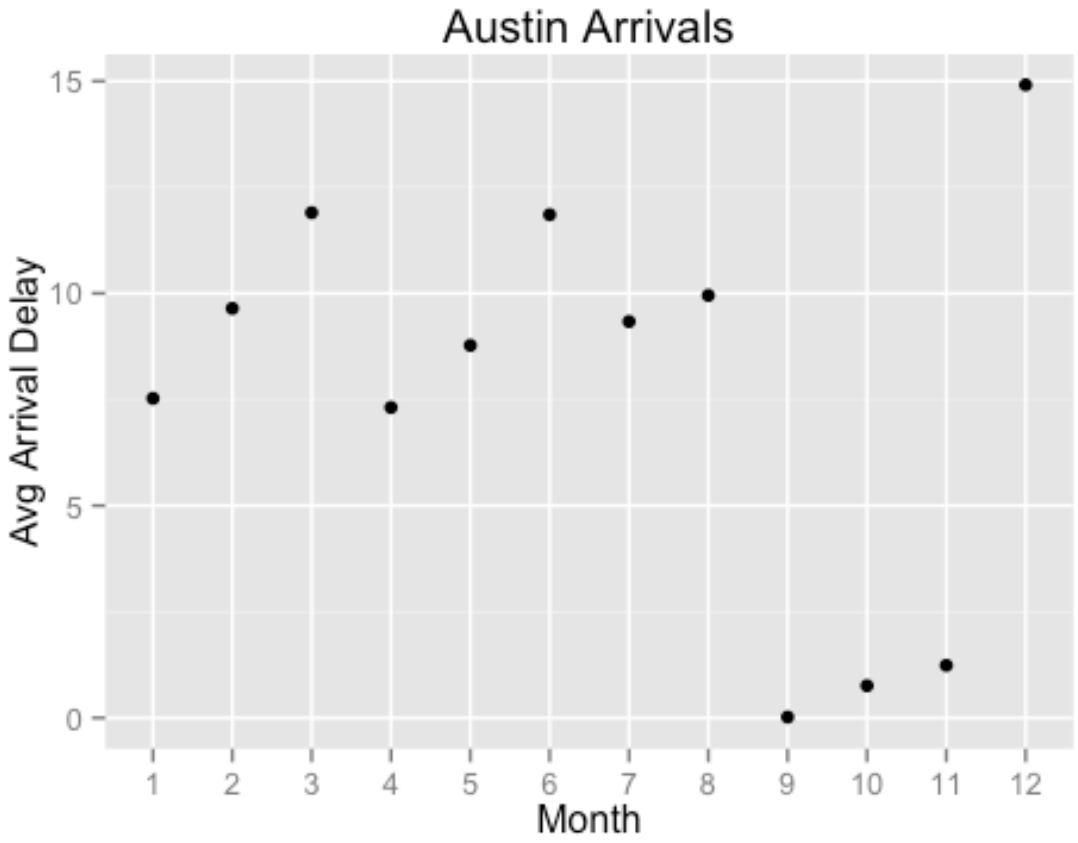
Here we can see which hours of the day are better to have as the scheduled arrival time, ignoring where you are coming from.



We clearly see that having a schedule time of arrival in the early morning is the best bet as arrival delays through these hours (4 am - 9 am) are used to be very low.

Monthly Delays

Here we can see which months are best and worst when it comes to arrival delays.

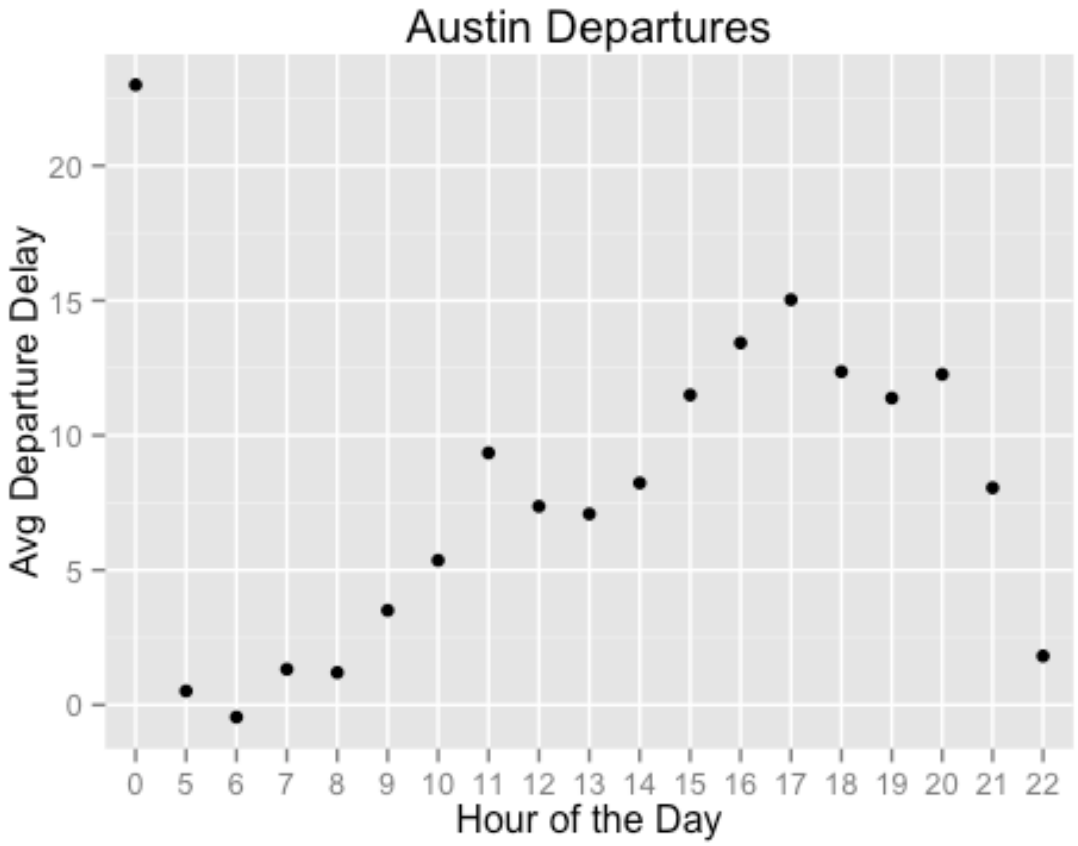


We see, as one would expect, that December is the worst month to fly to Austin, since on average we have a 15 minute arrival delay. On the other hand, September/October are months that usually don't have the enormous volume of passengers that December has, therefore, we can see that delays during these months are no big deal.

Departing Flights

Hourly Delays

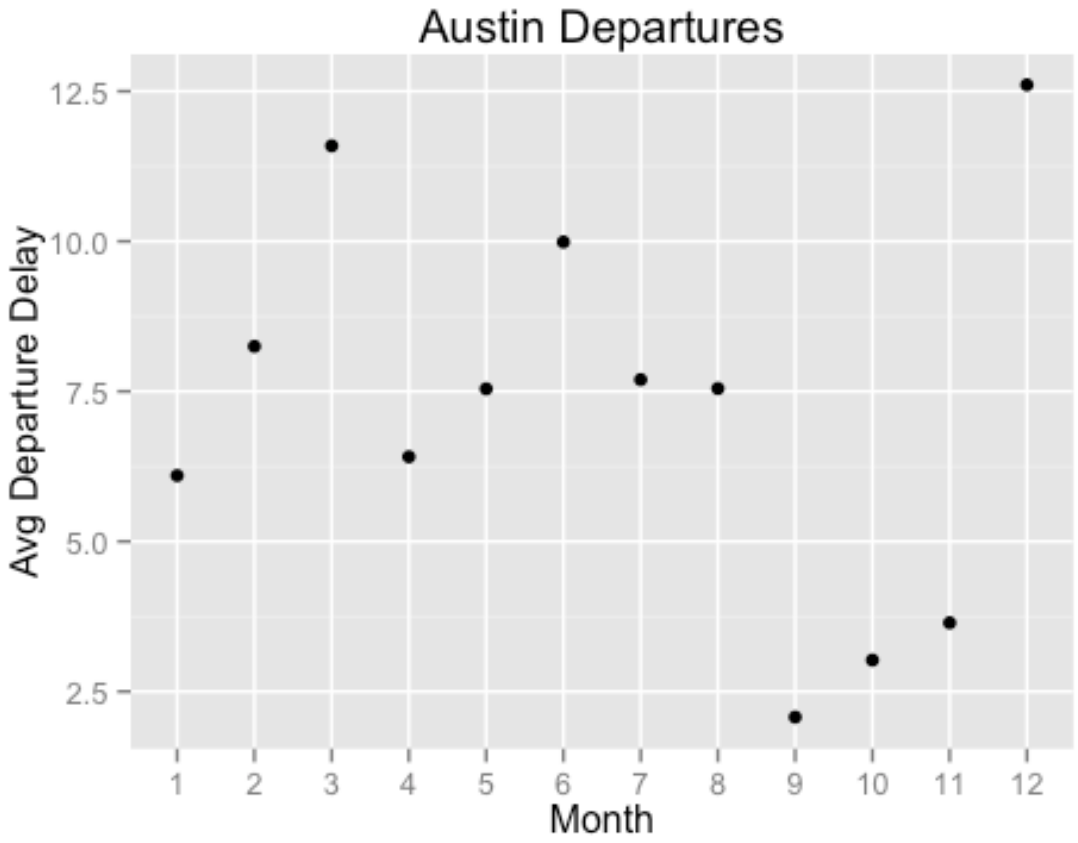
Here we can see which hours of the day are better to fly departing from Austin.



It is interesting to see that there are no flights departing from Austin between 1:00 am and 5:00 am. Also, one would think that departures are usually delayed more as the hours pass by, which we see that it holds true. Early in the morning we don't see much delays, but as we progress in the we see a stable increase in average delays.

Monthly Delays

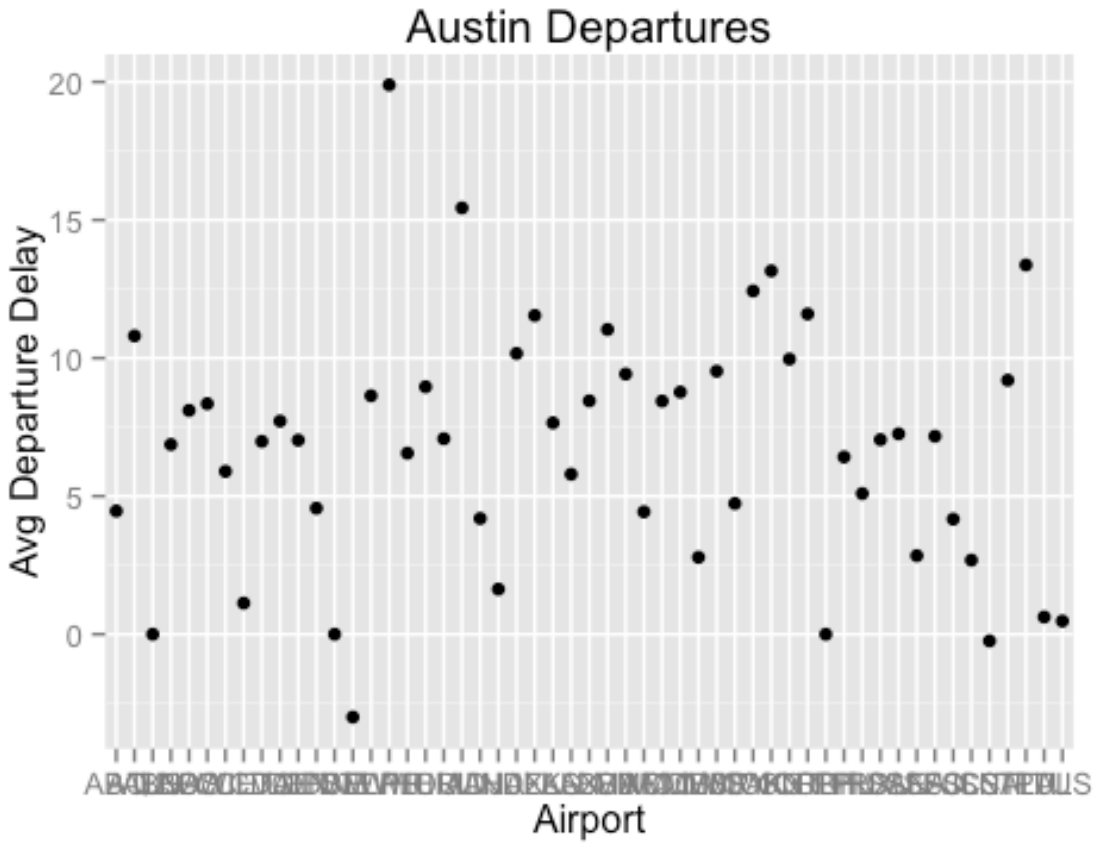
Here we can see which months are best and worst when flying from Austin.



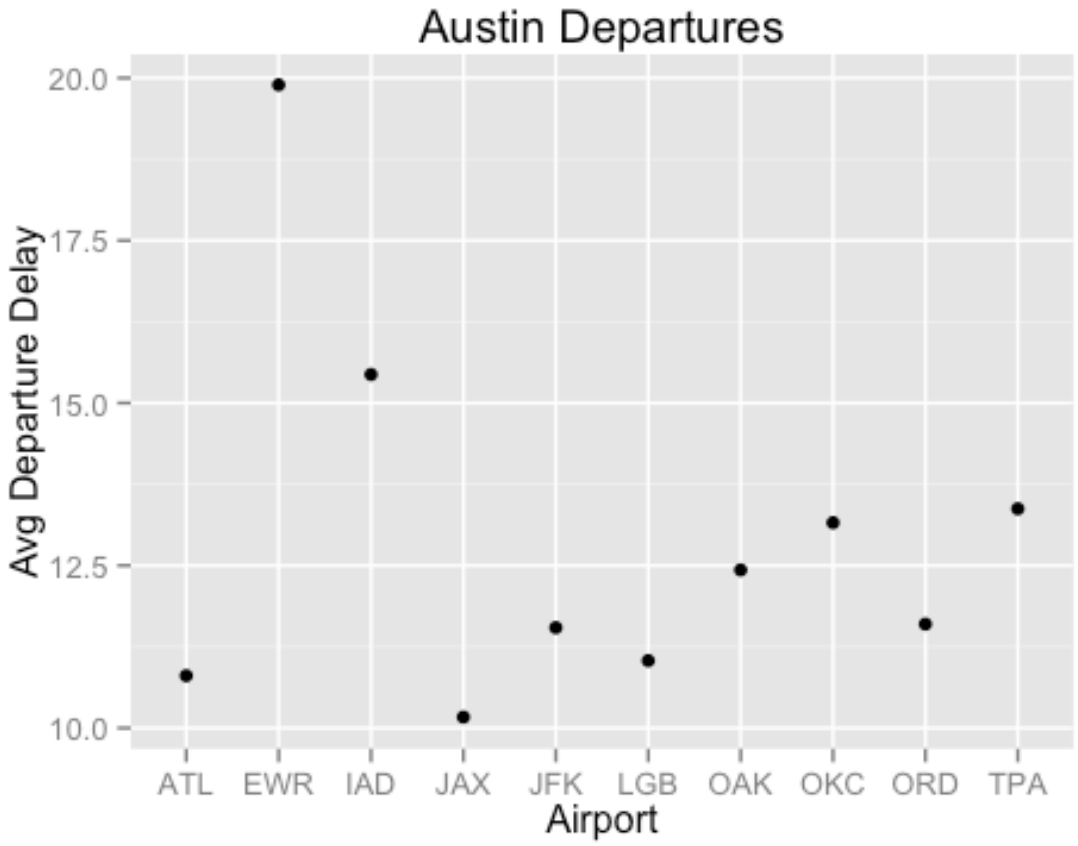
As we saw in the figure of monthly arrival delays, December was the worst month to travel, for departures this is no exception. We see that December again leads the worst average delays, and again September/October are the best months to travel. One interesting point to study is the fact that for departures March has higher delays almost reaching December's average delays. A hypothesis for this phenomenon is the fact that Austin is home to numerous Universities, and March represent the month when spring break usually occurs, therefore a lot of students are flying out of Austin, increasing in some level the average delays.

Worst Airports to Travel to

Here we see the average delays to each airport that someone can travel from Austin.



For display purposes the figure has been cut to just show the airports that have on average delays over 10 minutes



As one would expect big connecting airport like Newark Liberty International Airport or Washington Dulles International Airport, to name the top delays, have more volume of passengers and therefore are expected to have greater delays.

Author Attribution

First we read in our training data and create the corresponding corpus for it. When then proceed to clean the corpus by removing nuisance like numbers, punctuation and words that have no real descriptive meaning. Finally we create the Document Term Matrix(DTM) for the cleaned corpus.

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

## <<DocumentTermMatrix (documents: 2500, terms: 31423)>>
## Non-/sparse entries: 425955/78131545
```

```
## Sparsity           : 99%
## Maximal term length: 36
## Weighting          : term frequency (tf)
```

With these information we see that the sparsity of the data provided is 99% so we must remove some sparse terms in order to provide a DTM that is easier to handle, computationally cost speaking. A maximum of *0.97* was decided in order to remove the sparse terms.

```
## <<DocumentTermMatrix (documents: 2500, terms: 1141)>>
## Non-/sparse entries: 229853/2622647
## Sparsity           : 92%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

With a 92% we are more than happy to continue our study, therefore, we convert the DTM to a Dense Matrix (*X_train*). For the test data we run the same process that was applied to the the training data. This means creating the corresponding corpus, cleaning it and creating the DTM. Also, removing the sparse terms by a maximum of *0.97* since the first sparsity was also 99% and converting it to a Dense Matrix (*X_test*).

```
## <<DocumentTermMatrix (documents: 2500, terms: 32264)>>
## Non-/sparse entries: 432766/80227234
## Sparsity           : 99%
## Maximal term length: 45
## Weighting          : term frequency (tf)

## <<DocumentTermMatrix (documents: 2500, terms: 1166)>>
## Non-/sparse entries: 234051/2680949
## Sparsity           : 92%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

Naive Bayes

Now with both Dense Matrix we can proceed by running a Naive Bayes model. First we intersected both column names of each Dense Matrix in order to obtain the common words between both of them. Then we trimmed each Dense Matrix, based on these common words, in order to have them both with the same dimensions and column names (words).

```
## [1] 1166

## [1] 1141

## [1] 1049

## [1] 1049
```

We even see that at the beginning each Dense Matrix has a different number of columns but after trimming them they had the same number.

Now we smooth out each author, in order to account for the possibility that a word that hasn't been used by such author, can indeed be used in the future by the same author. We calculate the log probabilities under the Naive Bayes model in order to compare them and predict based on them the author of the Test data.

```
## [1] 59.12
```

We calculate the accuracy of the Naive Bayes model and see that we got a 59%, which is not as great as one would expect but at least is higher than just a plain 50:50 chance. Now For the sake of it we went and show specifically which Authors where easier to predict and which ones where just nearly impossible.

```
##   AaronPressman  FumikoFujisaki  GrahamEarnshaw    JimGilchrist 
##              82              98              80              96 
##     KarlPenhaul  LynneO'Donnell LynnleyBrowning    MatthewBunce 
##              88              80              98              84 
##   MichaelConnor       NickLouth      RobinSidel 
##              80              86              80 

## DavidLawder ScottHillis 
##          12          16
```

The above percentages show the accuracy of prediction for each individual author. We have just listed the ones that had an accuracy of 80% or higher and the ones that had a 20% or lower.

Multinomial Regression

Comparing Naive Bayes to a Multinomial regression we must first calculate the accuracy of the multinomial regression at predicting such authors.

Therefore, we first create a train model based on our *DTM_train* and vectors on the train corpus names. Also, since we have seen that this data has a lot of sparsity we decide to implement a Stochastic gradient descent efficiently which will estimate the maximum likelihood logistic regression coefficients from our sparse data. Moving forward we also run a prediction based on our training model and using the DTM_test as the testing DTM.

AS we did with the Naive Bayes model, based on our prediction we now calculate the accuracy of our multinomial regression model.

```
## [1] 5.2
```

We clearly see that using a multinomial regression model wont do us any good since its accuracy is only 5.2%, we must then decide to go with the Naive Bayes model

Practice with Association Rule Mining

When running the Association Rule Mining model it has been decided to go for a minimum support of *0.009* since this means that we are looking at at least 0.9% of the transactions have a certain itemset, this means around 90 baskets contained a certain itemset making it interesting to capture. Also, it has been decided to use a confidence minimum of *0.5*, meaning that we can say that that all rules will have a confidence of 50% or higher of actually happening.

```
library(arules)
```

```
## Loading required package: Matrix
##
## Attaching package: 'arules'
##
## The following object is masked from 'package:tm':
##
##     inspect
##
## The following objects are masked from 'package:base':
##
##     %in%, write
```

```
baskets <- read.transactions("groceries.txt", format = c("basket"), sep
= ',')
grocerie_rules <- apriori(baskets, parameter=list(support=0.009,
confidence=0.5))
```

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen
maxlen
##         0.5    0.1    1 none FALSE            TRUE   0.009      1
10
##  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 ... [93 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [25 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
inspect(grocerie_rules)
```

```
##    lhs                          rhs                    support
confidence     lift
## 1  {baking powder}         => {whole milk}       0.009252669
0.5229885 2.046793
## 2  {frozen vegetables,
##     other vegetables}      => {whole milk}       0.009659380
0.5428571 2.124552
## 3  {curd,
##     yogurt}                => {whole milk}       0.010066090
0.5823529 2.279125
## 4  {curd,
##     other vegetables}      => {whole milk}       0.009862735
0.5739645 2.246296
## 5  {brown bread,
##     other vegetables}      => {whole milk}       0.009354347
0.5000000 1.956825
## 6  {butter,
##     yogurt}                => {whole milk}       0.009354347
0.6388889 2.500387
## 7  {butter,
##     other vegetables}      => {whole milk}       0.011489578
0.5736041 2.244885
## 8  {domestic eggs,
##     other vegetables}      => {whole milk}       0.012302999
0.5525114 2.162336
## 9  {fruit/vegetable juice,
##     yogurt}                => {whole milk}       0.009456024
0.5054348 1.978094
## 10 {root vegetables,
##     whipped/sour cream}    => {whole milk}       0.009456024
0.5535714 2.166484
## 11 {whipped/sour cream,
##     yogurt}                => {whole milk}       0.010879512
0.5245098 2.052747
## 12 {other vegetables,
##     whipped/sour cream}    => {whole milk}       0.014641586
0.5070423 1.984385
## 13 {pip fruit,
##     yogurt}                => {whole milk}       0.009557702
0.5310734 2.078435
## 14 {other vegetables,
##     pip fruit}             => {whole milk}       0.013523132
0.5175097 2.025351
## 15 {pastry,
##     yogurt}                => {whole milk}       0.009150991
0.5172414 2.024301
## 16 {citrus fruit,
##     root vegetables}       => {other vegetables} 0.010371124
```

```
0.5862069 3.029608
## 17 {citrus fruit,
##     root vegetables}       => {whole milk}       0.009150991
0.5172414 2.024301
## 18 {root vegetables,
##     tropical fruit}        => {other vegetables} 0.012302999
0.5845411 3.020999
## 19 {root vegetables,
##     tropical fruit}        => {whole milk}       0.011997966
0.5700483 2.230969
## 20 {tropical fruit,
##     yogurt}                => {whole milk}       0.015149975
0.5173611 2.024770
## 21 {root vegetables,
##     yogurt}                => {other vegetables} 0.012913066
0.5000000 2.584078
## 22 {root vegetables,
##     yogurt}                => {whole milk}       0.014539908
0.5629921 2.203354
## 23 {rolls/buns,
##     root vegetables}       => {other vegetables} 0.012201322
0.5020921 2.594890
## 24 {rolls/buns,
##     root vegetables}       => {whole milk}       0.012709710
0.5230126 2.046888
## 25 {other vegetables,
##     yogurt}                => {whole milk}       0.022267412
0.5128806 2.007235
```

By running the model we have gotten 25 rules that are interesting to study, just to mention the the one with the highest confidence is the rule that states the following:

```
                {butter,yogurt} => {whole milk}
```

with a confidence of 63.88% and a support of around 0.94%.