Homework 2

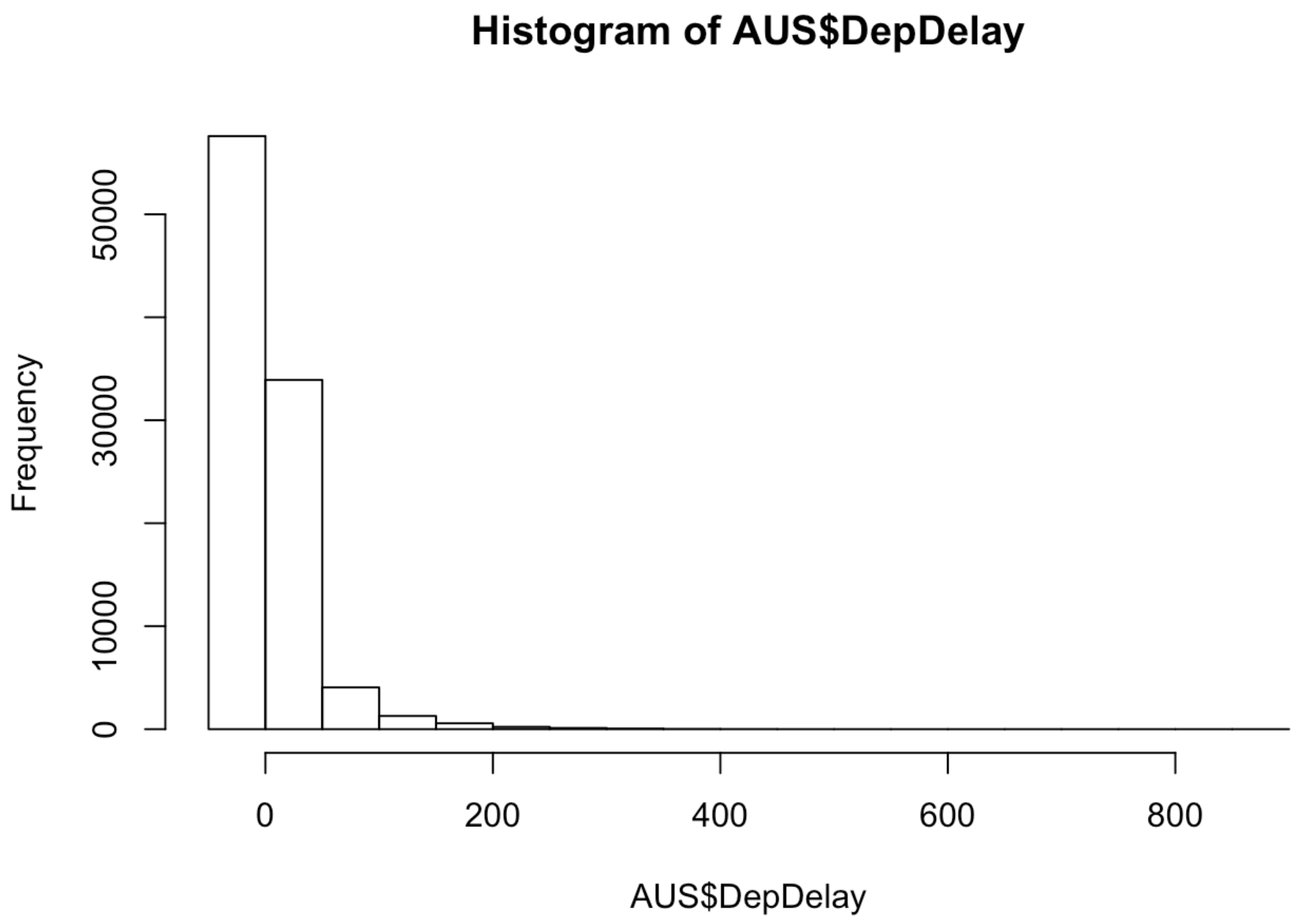
Julian Ghadially

August 17, 2015

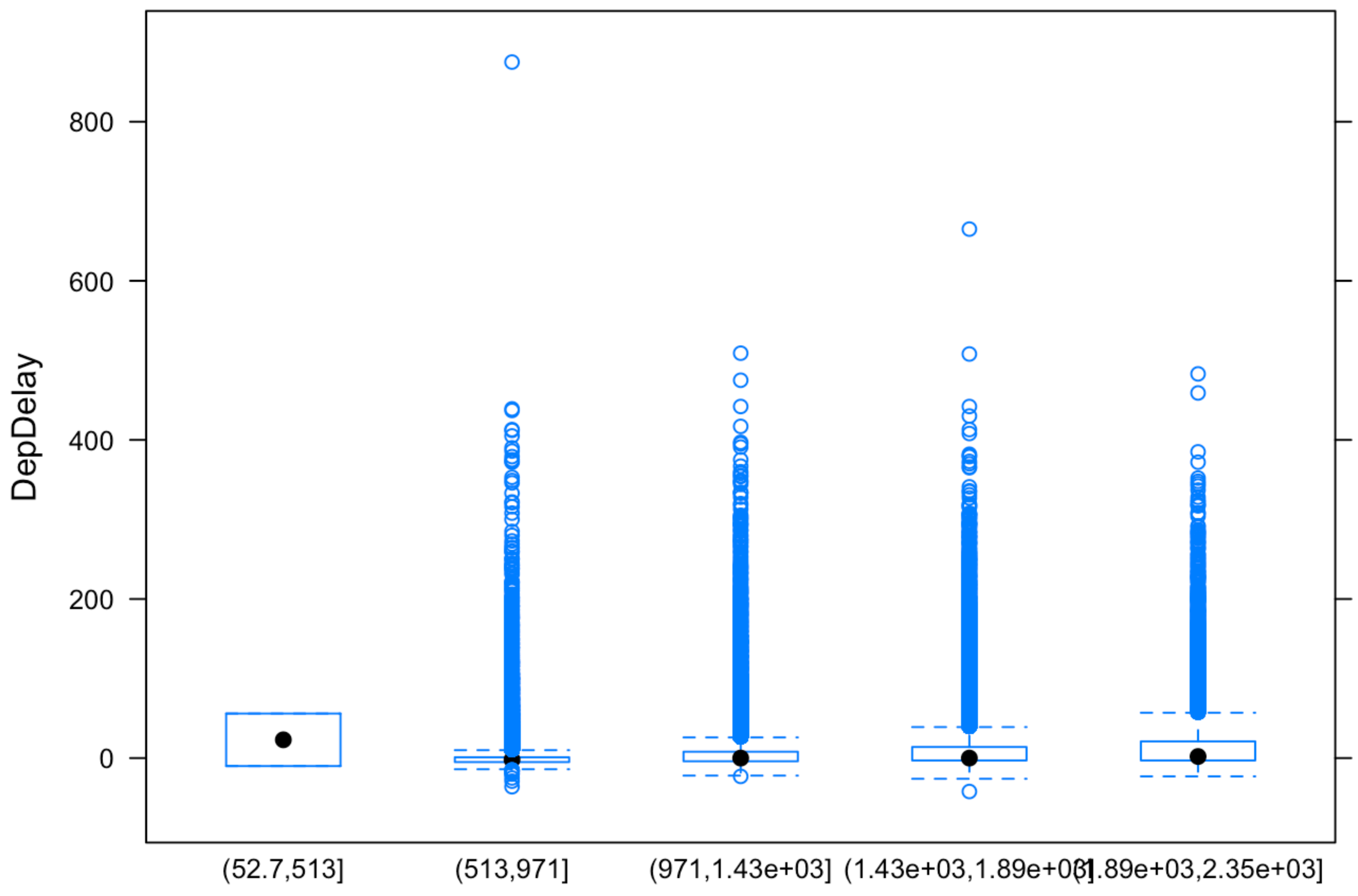
Problem 1: Flights at Austin Bergstrom International Airport

Do a time delay plot by airline.

```
## Loading required package: car
## Loading required package: dplyr
##
## Attaching package: 'dplyr'
##
## The following objects are masked from 'package:stats':
##
##     filter, lag
##
## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
##
## Loading required package: lattice
## Loading required package: mosaicData
##
## Attaching package: 'mosaic'
##
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
##
## The following object is masked from 'package:car':
##
##     logit
##
## The following object is masked from 'package:timeDate':
##
##     sample
##
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cov, D, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var
##
## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```



```
## [1] 45 55  0 56  1 51
```



Problem 2: Reuters Author Attribution

Bag of Words Model

In this exercise, works from 50 authors are analysed in order to attribute authorship to unknown text files from the same set of authors. Text is represented using a bag of words model. In the bag of words model, each document is a bag with counts associated with each word from the corpora of all authors. In order to best represent the text using bag of words, common words are ommitted, since they don't add as much value as the uncommon words. Sparse words are also ommitted.

Calculating Multinomial Probability Vector

The multinomial probability vector is made up of the likelihood of each word for each author involved. In order to do this, the document term matrix is condensed into counts of words per author. These counts are converted into likelihoods by dividing by the total corpora word count. A laplacian smoothing term of 1/2500 is added to each count in order to estimate the likelihood of words unseen by the training corpora.

Handling words in the test set that are not present in the training data

The naive-bayes model is trained solely on documents from the training text, however, words from the test set are included in the document term matrices for the training set. the likelihoods of the unseen test words in the training set are equal to the value dictated by Laplacian smoothing. Because this is likelihood we would assign them if they were in seperate document term matrices, it is acceptable to include them in the set of training words. Another way to handle the unseen test words would be to use only the intersection of words between the two corpora. However, this biases the training set to include information from the test set through omission of words unique to the training set.

Naive-Bayes Performance

The Naive-Bayes model described above attributed the correct author **57%** of the time. This is very good compared to a 2% chance of getting the correct author with no model at all. Also, author attribution varied for different authors. the table below displays accuracy for each author. Six authors were very troublesome to predict, with classification scores of 20% or less.

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

```
## [1] 0.5728
```

```
goodclassifications = 0
misclassifications = 0
misclass_per_author = NULL
for (j in 1:2500) {
  Bayes = NULL
  for (i in 1:50) {
    Bayes = append(Bayes, sum(X_test[j,]*w_authors_train[i,]))
  }
  attributed.author = which(Bayes == max(Bayes))
  attributed.author = authors_unique[attributed.author]
  #print(attributed.author)
  if (attributed.author == test.authors[j]) {
    goodclassifications = goodclassifications + 1
  }
  else{misclassifications = misclassifications + 1}
  if (j%%50 == 0) {
    misclass_per_author = append(misclass_per_author, goodclassifications/sum(goodcla
ssifications, misclassifications))
    misclassifications = 0
    goodclassifications = 0
  }
}
mpa.df = data.frame(x = authors_unique, y = misclass_per_author)
mpa.df
```

```
##                     x    y
## 1      AaronPressman 0.70
## 2          AlanCrosby 0.44
## 3      AlexanderSmith 0.96
## 4     BenjaminKangLim 0.16
## 5       BernardHickey 0.50
## 6         BradDorfman 0.86
## 7     DarrenSchuettler 0.36
## 8         DavidLawder 0.14
## 9       EdnaFernandes 0.08
## 10        EricAuchard 0.58
## 11     FumikoFujisaki 0.98
## 12     GrahamEarnshaw 0.74
## 13   HeatherScoffield 0.32
## 14      JaneMacartney 0.64
## 15          JanLopatka 0.32
## 16        JimGilchrist 0.92
## 17            JoeOrtiz 0.36
## 18        JohnMastrini 0.78
## 19        JonathanBirt 0.50
## 20      JoWinterbottom 0.74
## 21         KarlPenhaul 0.54
## 22           KeithWeir 0.72
## 23     KevinDrawbaugh 0.42
## 24      KevinMorrison 0.76
## 25      KirstinRidley 0.72
## 26 KouroshKarimkhany 0.36
## 27           LydiaZajc 0.62
## 28     LynneO'Donnell 0.80
## 29    LynnleyBrowning 1.00
## 30    MarcelMichelson 0.68
## 31       MarkBendeich 0.36
## 32         MartinWolk 0.50
## 33       MatthewBunce 0.84
## 34      MichaelConnor 0.60
## 35         MureDickie 0.32
## 36          NickLouth 0.92
## 37    PatriciaCommins 0.56
## 38      PeterHumphrey 0.88
## 39         PierreTran 0.68
## 40         RobinSidel 0.78
## 41       RogerFillion 0.70
## 42        SamuelPerry 0.52
## 43       SarahDavison 0.62
## 44        ScottHillis 0.06
## 45        SimonCowell 0.52
## 46           TanEeLyn 0.08
## 47     TheresePoletti 0.62
## 48         TimFarrand 0.76
## 49         ToddNissen 0.42
## 50       WilliamKazer 0.20
```

Random Forest Model

Naive Bayes did well predicting authors. Random Forests can also be applied to the same bag of words analysis. In order to do so, the bag of words model needs less variables. We can choose a subset of variables using principal component analysis (PCA). PCA generates 1422 principal components. However, 1422 variables for 2500 observations would produce too much variability error. Thus, we select the top principal components. The number of principal components should be selected using cross validation, for increased accuracy. Using 50 principal components, the random forest model is fit to the training data and used to predict on the test data. The percent of correct author attributions obtained is **66%.** This is higher than the Naive Bayes because Random Forest is using more important components dictated by PCA. As a result, the five authors that were difficult to predict are less difficult to predict. shown below is a table of prediction accuracy for each author by Random Forests.

```
## randomForest 4.6-10
## Type rfNews() to see new features/changes/bug fixes.
##
## Attaching package: 'randomForest'
##
## The following object is masked from 'package:dplyr':
##
##     combine
```

```
## [1] 0.6664
```

```
##                    x    y
## 1      AaronPressman 0.82
## 2         AlanCrosby 0.80
## 3     AlexanderSmith 0.56
## 4    BenjaminKangLim 0.46
## 5      BernardHickey 0.72
## 6        BradDorfman 0.36
## 7   DarrenSchuettler 0.76
## 8        DavidLawder 0.82
## 9      EdnaFernandes 0.48
## 10       EricAuchard 0.50
## 11    FumikoFujisaki 0.94
## 12    GrahamEarnshaw 0.70
## 13  HeatherScoffield 0.86
## 14     JaneMacartney 0.52
## 15        JanLopatka 0.72
## 16      JimGilchrist 0.94
## 17          JoeOrtiz 0.56
## 18      JohnMastrini 0.54
## 19      JonathanBirt 0.66
## 20    JoWinterbottom 0.80
## 21       KarlPenhaul 0.58
## 22         KeithWeir 0.66
## 23    KevinDrawbaugh 0.60
## 24     KevinMorrison 0.56
## 25     KirstinRidley 0.52
## 26 KouroshKarimkhany 0.74
## 27         LydiaZajc 0.92
## 28    LynneO'Donnell 0.98
## 29   LynnleyBrowning 0.96
## 30   MarcelMichelson 0.84
## 31       MarkBendeich 0.62
## 32         MartinWolk 0.66
## 33       MatthewBunce 0.90
## 34     MichaelConnor 0.60
## 35         MureDickie 0.34
## 36          NickLouth 0.76
## 37   PatriciaCommins 0.60
## 38     PeterHumphrey 0.74
## 39         PierreTran 0.72
## 40         RobinSidel 0.74
## 41       RogerFillion 0.86
## 42        SamuelPerry 0.58
## 43       SarahDavison 0.52
## 44        ScottHillis 0.28
## 45        SimonCowell 0.64
## 46           TanEeLyn 0.40
## 47    TheresePoletti 0.50
## 48         TimFarrand 0.74
## 49         ToddNissen 0.92
## 50       WilliamKazer 0.32
```

Problem 3: Association Rule Mining

Association between grocery items is important for grocery stores because it can help them understand what items are being bought together, which can in tern help them manage supply and appropriately place inventory and promotions. Associations for groceries from groceries.txt were gathered using a minimum support of .01 and confidence of .5. The associations found are displayed below. This analysis points to two common items, milk and vegetables associated with particular types of products. purchase of milk is associated with eggs and various dairy products, and purchase of “other vegetables”" is associated with purchase of various produce. These all have a support over .01, which means that 10% of all grocery carts contained these associations. They also all have a confidence greater than .5, which means that of grocery carts containing itemset A, over 50% contained the righthand side itemset.

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

```
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport support minlen maxlen
##         0.5    0.1    1 none FALSE            TRUE    0.01      1      4
##  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 4 done [0.00s].
## writing ... [15 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
##    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
```