HW2_McNabb

Catherine McNabb

August 20, 2018

Problem 1) Flight Info at Austin Bergstrom

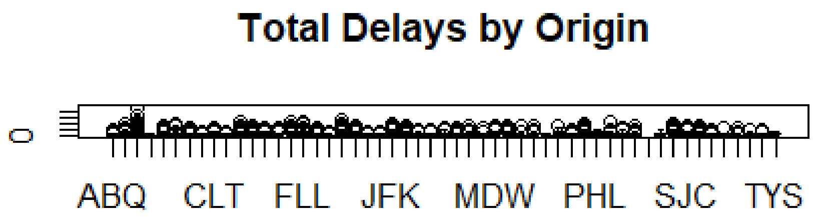
First, I'm going to read in the data.

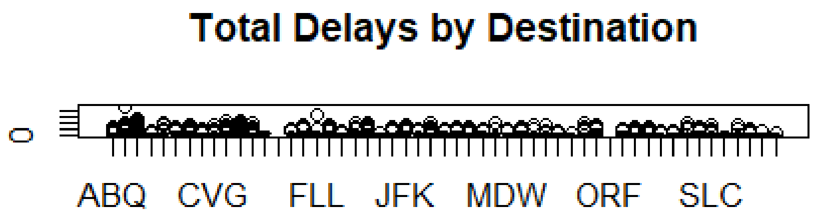
```r
abia = read.csv("ABIA.csv")
```

Then, I am going to create a new column called "total delays", so we can look at all of the delays together.

```r
abia$totaldelays <- rowSums(abia[,25:29])
attach(abia)
```

Let's compare total delays by origin and destination to and from Austin Bergstrom.

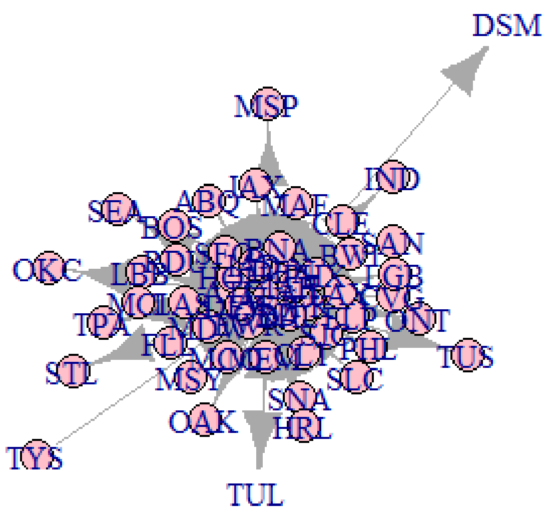




It appears that origin and destination delays are pretty comparable for each airport.

Now, I'm going to make a graph that shows the network of airports with flights going TO AUS. The distance the node from AUS indicates how common that airport is, with the closer ones being more common, and the size of the node indicates how many delays, with bigger nodes indicating bigger delays.

```
## Warning: package 'igraph' was built under R version 3.5.1

##
## Attaching package: 'igraph'

## The following objects are masked from 'package:stats':
##
##     decompose, spectrum

## The following object is masked from 'package:base':
##
##     union
```



Problem 2) Predicting Authors

First, I need to load in the data and read it in. I will start with training data.

```
## Loading required package: NLP

##
## Attaching package: 'proxy'

## The following objects are masked from 'package:stats':
##
##     as.dist, dist
## The following object is masked from 'package:base':
##
##     as.matrix
```

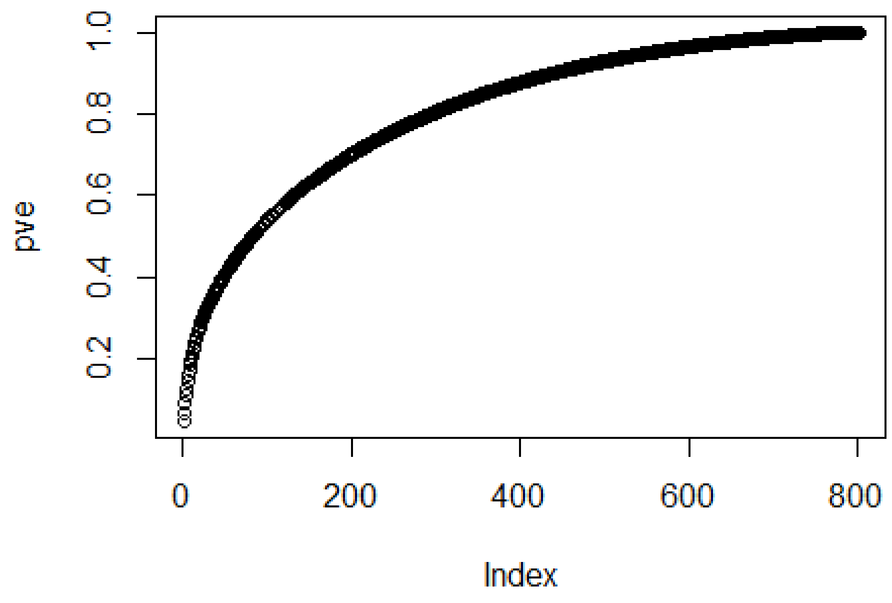
Then, I need to clean up the names of the documents and authors.

Then, I will clean up the data, so get rid of common words, numbers, weird symbols, etc.

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

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

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

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

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

Then do the same with test, including taking care of any words that aren't in the training data.

Now, I need to do PCA to get the right amount of components.



This looks like cutting the data at 100 will do.

Now, we need to turn this into models.

First, random forest:

```
## Warning: package 'randomForest' was built under R version 3.5.1

## randomForest 4.6-14

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

## [1] 0.9768

## [1] 0.022
```

Unfortunately, seems like we did not predict very well on the test set. We did a LOT better on the training set, which isn't really helpful.

Now, I'll try naive bayes.

```
## Warning: package 'gbm' was built under R version 3.5.1

## Loading required package: survival

## Loading required package: lattice

## Loading required package: splines

## Loading required package: parallel

## Loaded gbm 2.1.3

## [1] 0.7736

## [1] 0.0176
```

Again, very, very poor showing for the the testing set. I probably did something wrong, but this does not bode well for predicting authors.

Problem 3) Groceries List

First, I'm going to read in the data and correct libraries.

```
library(tidyverse)

## -- Attaching packages -------------------

## v ggplot2 3.0.0     v purrr   0.2.5
## v tibble  1.4.2     v dplyr   0.7.6
## v tidyr   0.8.1     v stringr 1.3.1
## v readr   1.1.1     v forcats 0.3.0
## -- Conflicts -- tidyverse_conflicts() --
## x ggplot2::annotate()    masks NLP::annotate()
## x dplyr::as_data_frame() masks tibble::as_data_frame(),
igraph::as_data_frame()
## x dplyr::combine()       masks randomForest::combine()
## x purrr::compose()       masks igraph::compose()
## x tidyr::crossing()      masks igraph::crossing()
## x tidyr::extract()       masks magrittr::extract()
## x dplyr::filter()        masks stats::filter()
## x dplyr::groups()        masks igraph::groups()
## x dplyr::lag()           masks stats::lag()
## x ggplot2::margin()      masks randomForest::margin()
## x purrr::set_names()     masks magrittr::set_names()
## x purrr::simplify()      masks igraph::simplify()
```

```r
library(arules)  # has a big ecosystem of packages built around it
```

```
## 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
```

```r
library(arulesViz)
```

```
## Loading required package: grid
```

```r
groceries_raw = read.csv("groceries.txt",header=FALSE)
```

Then, I will set the data up to be in the correct format.

```r
x = matrix(0,nrow= nrow(groceries_raw)*4,ncol=2)

counter = 1
x = data.frame(x)
```

```r
for (i in 1:15296) {
  g = groceries_raw[i,]
  for (j in 1:length(g)) {
    x[counter,1] = i
    x[counter,2] = as.character(g[[j]])
    counter = counter+1
  }
}
```

Now, I will format the data so we can graph using different support and confidence rules.

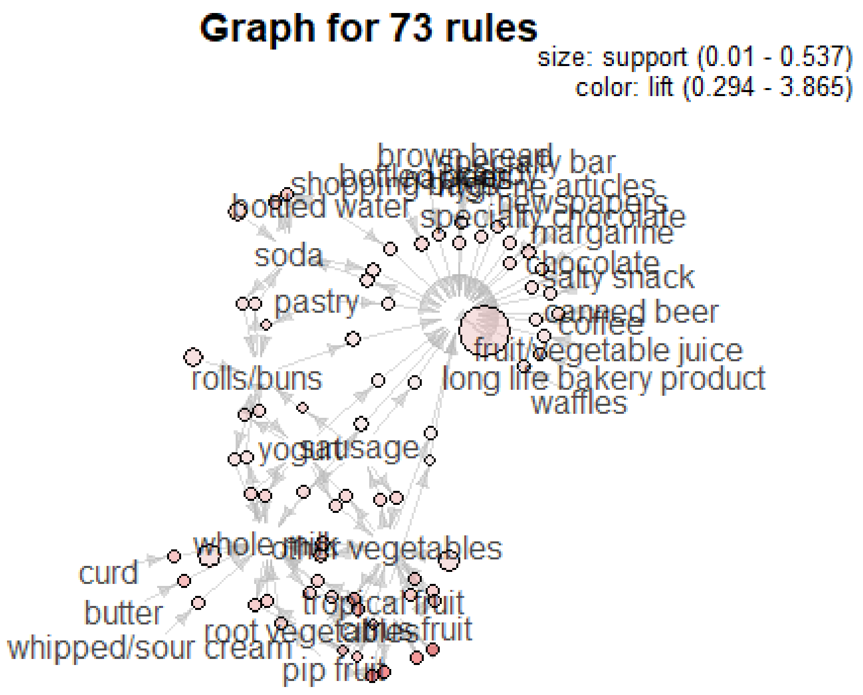
```r
x$X1 = factor(x$X1)
baskets = split(x=x$X2, f=x$X1)

## Remove duplicates ("de-dupe")
baskets = lapply(baskets, unique)

#make into special arules class
shoptrans = as(baskets, "transactions")

grocerylist = apriori(shoptrans,
                      parameter=list(support=.005, confidence=.1, maxlen=5))

## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    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: 76
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[170 item(s), 15296 transaction(s)] done [0.00s].
## sorting and recoding items ... [102 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [173 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

I wanted to look at the results for multiple combinations of support and confidence, so I tried several.

```r
sub1 = subset(grocerylist, subset=confidence > 0.01 & support > 0.005)
sub2 = subset(grocerylist, subset=confidence > 0.05 & support > 0.01)
sub3 = subset(grocerylist, subset=confidence > 0.0075 & support > 0.0045)
```

And then I found that the one with the most lax rules - confidence > 0.05 and support > 0.01 seemed to be the ones with the best grouping. Perhaps because there are some items that are bought so often, ‘strict’ rules seem to overfit, and the ‘looser’ rules work better.

```r
plot(sub2, method='graph')
```


Graph for 73 rules
size: support (0.01 - 0.537)
color: lift (0.294 - 3.865)
brown bread
specialty bar
bottled beer
shopping bags
hygiene articles
bottled water
newspapers
specialty chocolate
margarine
soda
chocolate
salty snack
pastry
canned beer
coffee
fruit/vegetable juice
rolls/buns
long life bakery product
waffles
yogurt
sausage
whole milk
other vegetables
curd
butter
tropical fruit
whipped/sour cream
root vegetables
citrus fruit
pip fruit