Intro to Machine Learning (STA380) - Part 2

Christian Alfonso, Musmin Zar, Satya Pal, Vinay Pahwa

16/08/2021

Link to Github repository: "https://github.com/satyapal07/ML2_STA380_exercises"

```
library(mosaic)
```

```
## Warning: package 'mosaic' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'mosaic':
##   method                           from
##   fortify.SpatialPolygonsDataFrame ggplot2
```

```
##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
```

```
##
## Attaching package: 'mosaic'
```

```
## The following objects are masked from 'package:dplyr':
##
##     count, do, tally
```

```
## The following object is masked from 'package:Matrix':
##
##     mean
```

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

```
## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, 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
```

```
library(quantmod)
```

```
## Warning: package 'quantmod' was built under R version 4.0.5
```

```
## Loading required package: xts
```

```
## Warning: package 'xts' was built under R version 4.0.5
```

```
## Loading required package: zoo
```

```
## Warning: package 'zoo' was built under R version 4.0.5
```

```
##
## Attaching package: 'zoo'
```

```
## The following objects are masked from 'package:base':
##
##     as.Date, as.Date.numeric
```

```
##
## Attaching package: 'xts'
```

```
## The following objects are masked from 'package:dplyr':
##
##     first, last
```

```
## Loading required package: TTR
```

```
## Warning: package 'TTR' was built under R version 4.0.5
```

```
## Registered S3 method overwritten by 'quantmod':
##   method            from
##   as.zoo.data.frame zoo
```

```
library(foreach)
```

```
## Warning: package 'foreach' was built under R version 4.0.5
```

```
library(tm)
```

```
## Warning: package 'tm' was built under R version 4.0.5
```

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

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

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

##
## Attaching package: 'tm'

## The following object is masked from 'package:mosaic':
##
##     inspect
```

```
library(magrittr)
```

```
## Warning: package 'magrittr' was built under R version 4.0.5
```

```
library(e1071)
```

```
## Warning: package 'e1071' was built under R version 4.0.5
```

```
library(caret)
```

```
## Warning: package 'caret' was built under R version 4.0.5

##
## Attaching package: 'caret'

## The following object is masked from 'package:mosaic':
##
##     dotPlot
```

```
library(dplyr)
library(doParallel)
```

```
## Warning: package 'doParallel' was built under R version 4.0.5

## Loading required package: iterators

## Warning: package 'iterators' was built under R version 4.0.5

## Loading required package: parallel
```

```
library(foreach)
library(randomForest)
```

```
## Warning: package 'randomForest' was built under R version 4.0.5

## randomForest 4.6-14
```

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

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

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

```
library(plyr)
```

```
## Warning: package 'plyr' was built under R version 4.0.5

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following object is masked from 'package:mosaic':
##
##     count

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize
```

```
library(arules) ## install.packages('arules')
```

```
## Warning: package 'arules' was built under R version 4.0.5

##
## Attaching package: 'arules'

## The following object is masked from 'package:tm':
##
##     inspect

## The following objects are masked from 'package:mosaic':
##
##     inspect, lhs, rhs
```

```
## The following object is masked from 'package:dplyr':
##
##     recode

## The following objects are masked from 'package:base':
##
##     abbreviate, write
```

```
library(LICORS)
```

```
## Warning: package 'LICORS' was built under R version 4.0.5
```

```
library(foreach)
library(plotly)
```

```
## Warning: package 'plotly' was built under R version 4.0.5

##
## Attaching package: 'plotly'

## The following objects are masked from 'package:plyr':
##
##     arrange, mutate, rename, summarise

## The following object is masked from 'package:mosaic':
##
##     do

## The following object is masked from 'package:ggplot2':
##
##     last_plot

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

```
library(ggplot2)
library(tidyverse)
```

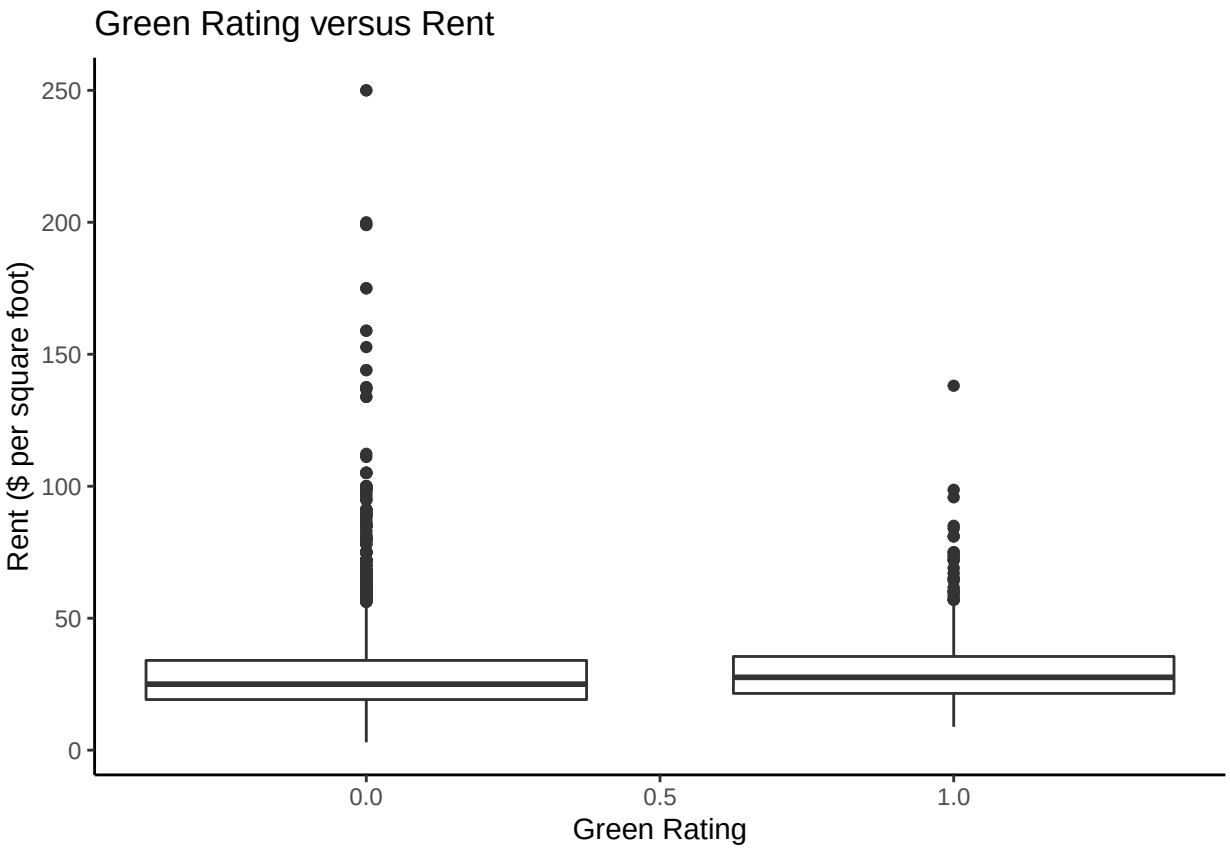
```
## Warning: package 'tidyverse' was built under R version 4.0.5

## -- Attaching packages --------------------------------------- tidyverse 1.3.1 --

## v tibble  3.1.3     v purrr   0.3.4
## v tidyr   1.1.3     v stringr 1.4.0
## v readr   2.0.0     v forcats 0.5.1
```

```
## Warning: package 'tibble' was built under R version 4.0.5

## Warning: package 'tidyr' was built under R version 4.0.5

## Warning: package 'readr' was built under R version 4.0.5

## Warning: package 'purrr' was built under R version 4.0.5

## Warning: package 'stringr' was built under R version 4.0.5

## Warning: package 'forcats' was built under R version 4.0.5

## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x purrr::accumulate()        masks foreach::accumulate()
## x NLP::annotate()            masks ggplot2::annotate()
## x plotly::arrange()          masks plyr::arrange(), dplyr::arrange()
## x randomForest::combine()    masks dplyr::combine()
## x purrr::compact()           masks plyr::compact()
## x plyr::count()              masks mosaic::count(), dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x plotly::do()               masks mosaic::do(), dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x tidyr::extract()           masks magrittr::extract()
## x plyr::failwith()           masks dplyr::failwith()
## x plotly::filter()           masks dplyr::filter(), stats::filter()
## x xts::first()               masks dplyr::first()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x plyr::id()                 masks dplyr::id()
## x dplyr::lag()               masks stats::lag()
## x xts::last()                masks dplyr::last()
## x purrr::lift()              masks caret::lift()
## x randomForest::margin()     masks ggplot2::margin()
## x plotly::mutate()           masks plyr::mutate(), dplyr::mutate()
## x tidyr::pack()              masks Matrix::pack()
## x arules::recode()           masks dplyr::recode()
## x plotly::rename()           masks plyr::rename(), dplyr::rename()
## x purrr::set_names()         masks magrittr::set_names()
## x mosaic::stat()             masks ggplot2::stat()
## x plotly::summarise()        masks plyr::summarise(), dplyr::summarise()
## x plyr::summarize()          masks dplyr::summarize()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
## x purrr::when()              masks foreach::when()
```

Question 1: Visual Story Telling (Part 1): Green Building

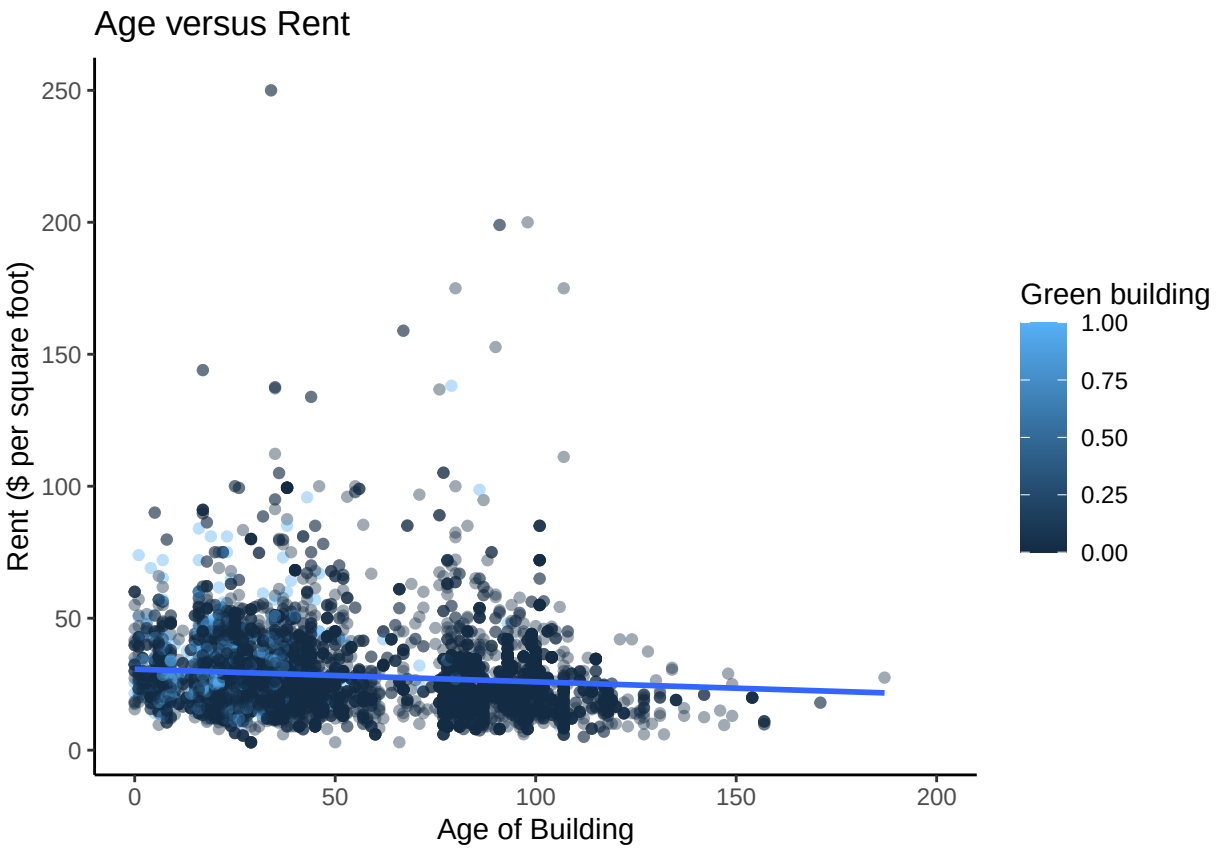
I do not agree with the stats guru. For starters, only 216 houses have an occupancy of less than 10%. This is a small amount of buildings, and there is no way to know what is happening in this buildins, so we shouldn't just write them off.

To begin, we have the plot of the data the Guru uses.



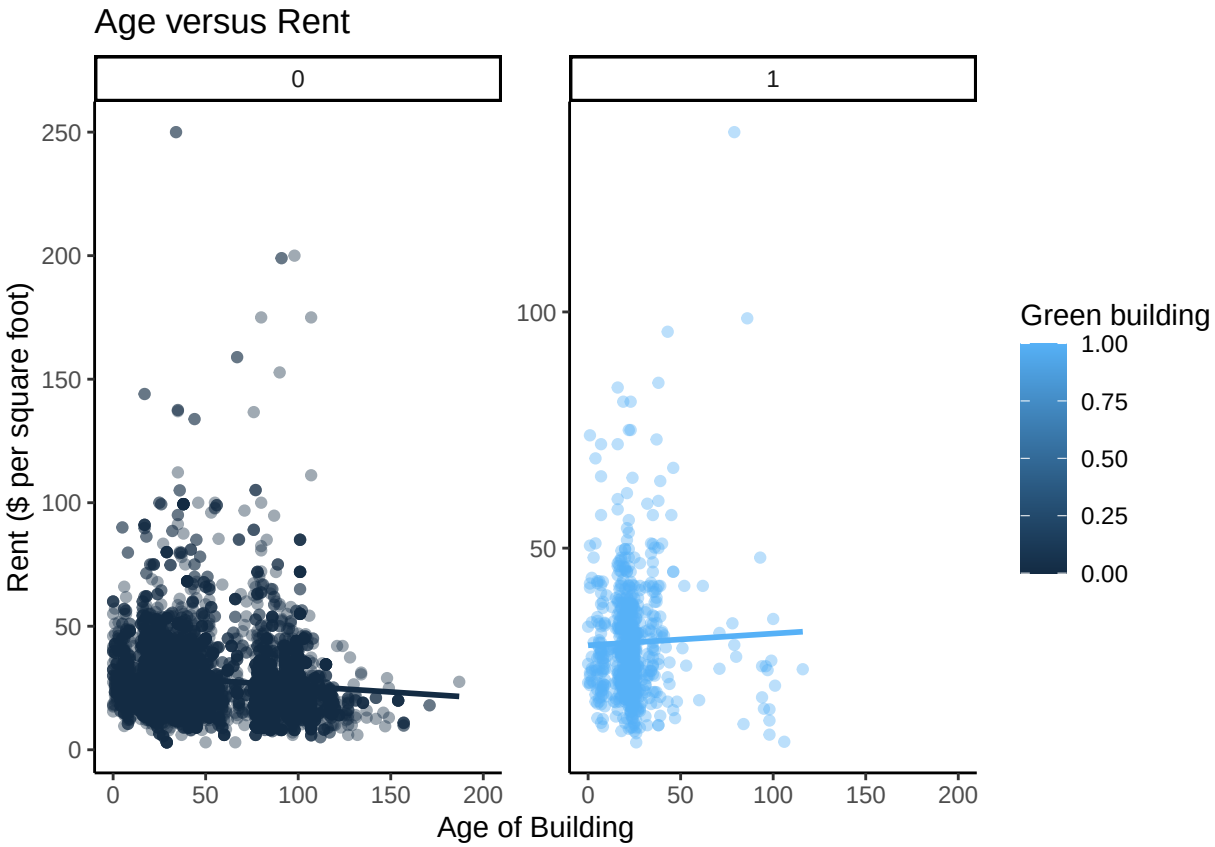
He is saying the rent is higher because the buildings are green. He seems right if you use only this data, but lets look at some more.

```
## `geom_smooth()` using formula 'y ~ x'
```

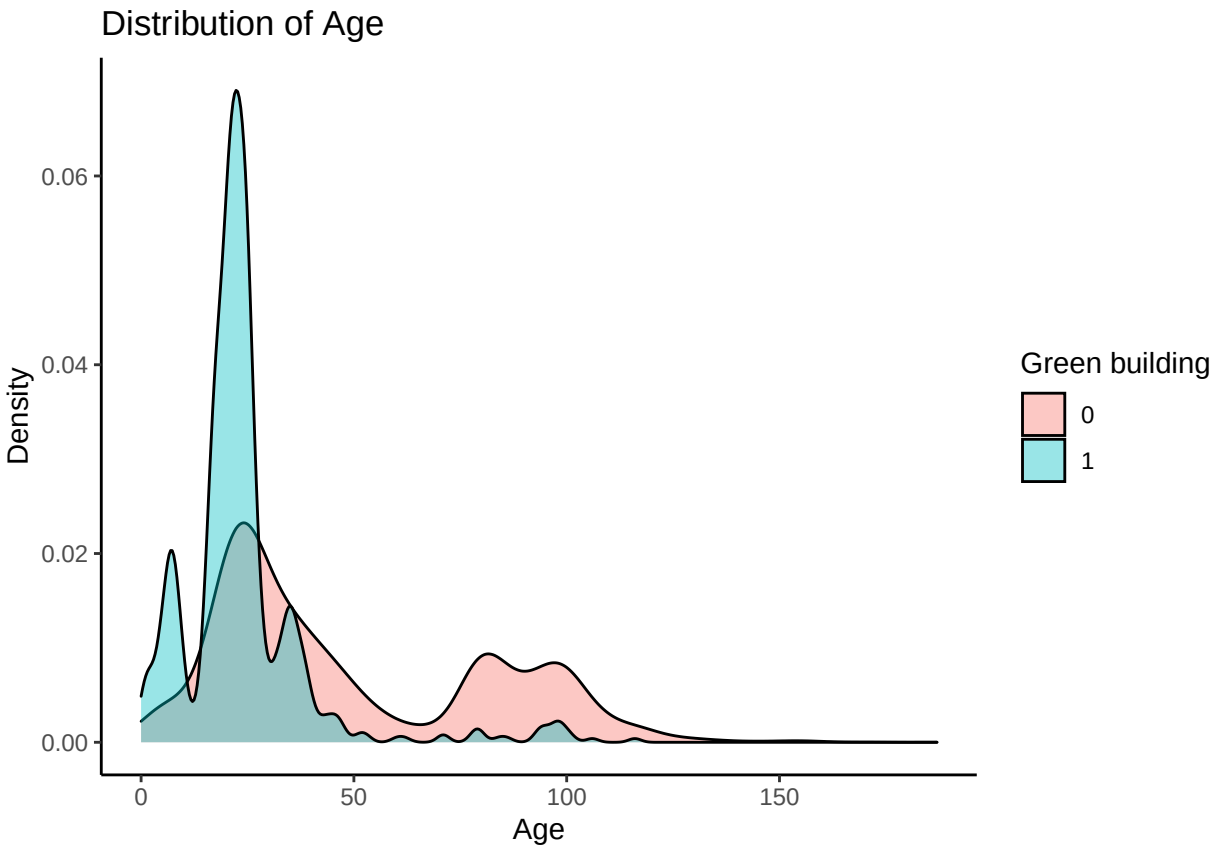


Here, I am looking at the relationship between Age and Rent. It is a bit hard to see so we will separate it.

```
## `geom_smooth()` using formula 'y ~ x'
```



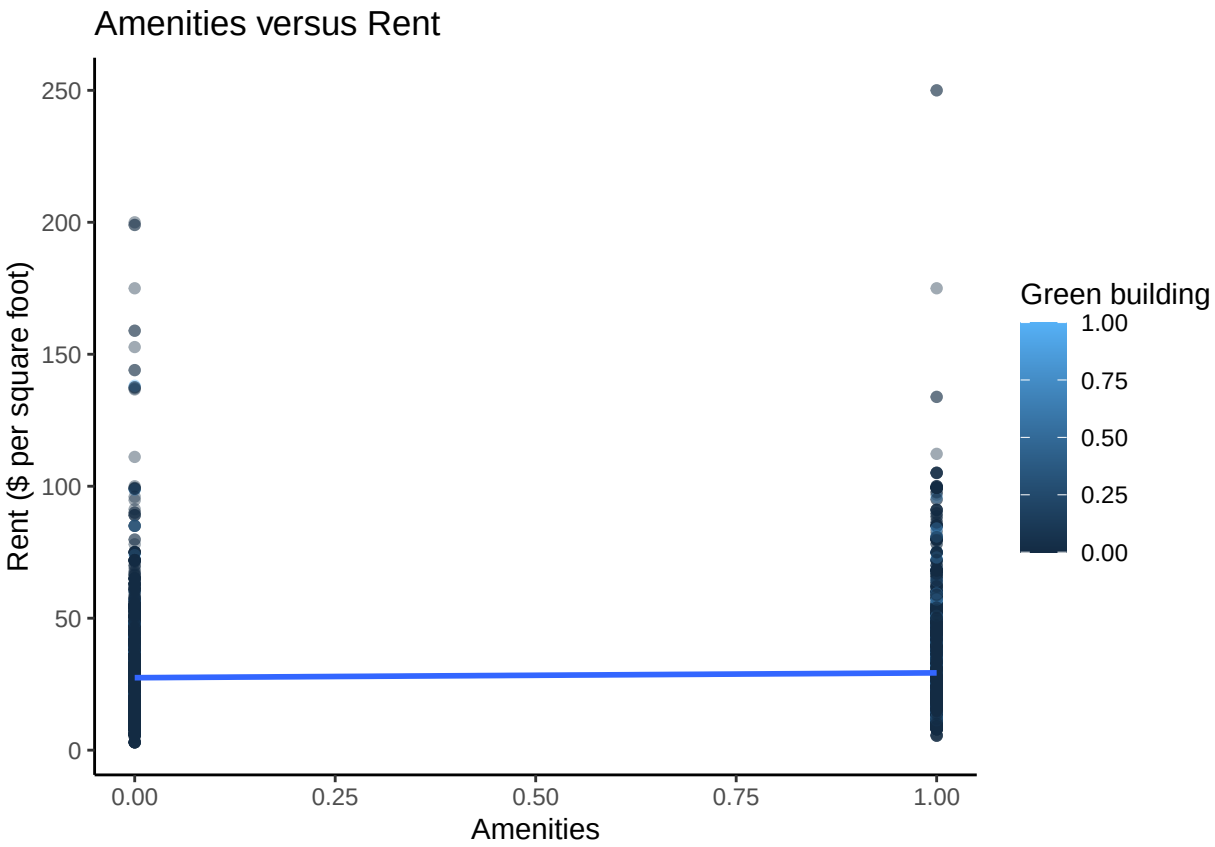
Now we can see how rent goes down as age goes up for non-green buildings, but the inverse is true for green buildings. We can already start to see a little of how the Guru is wrong.



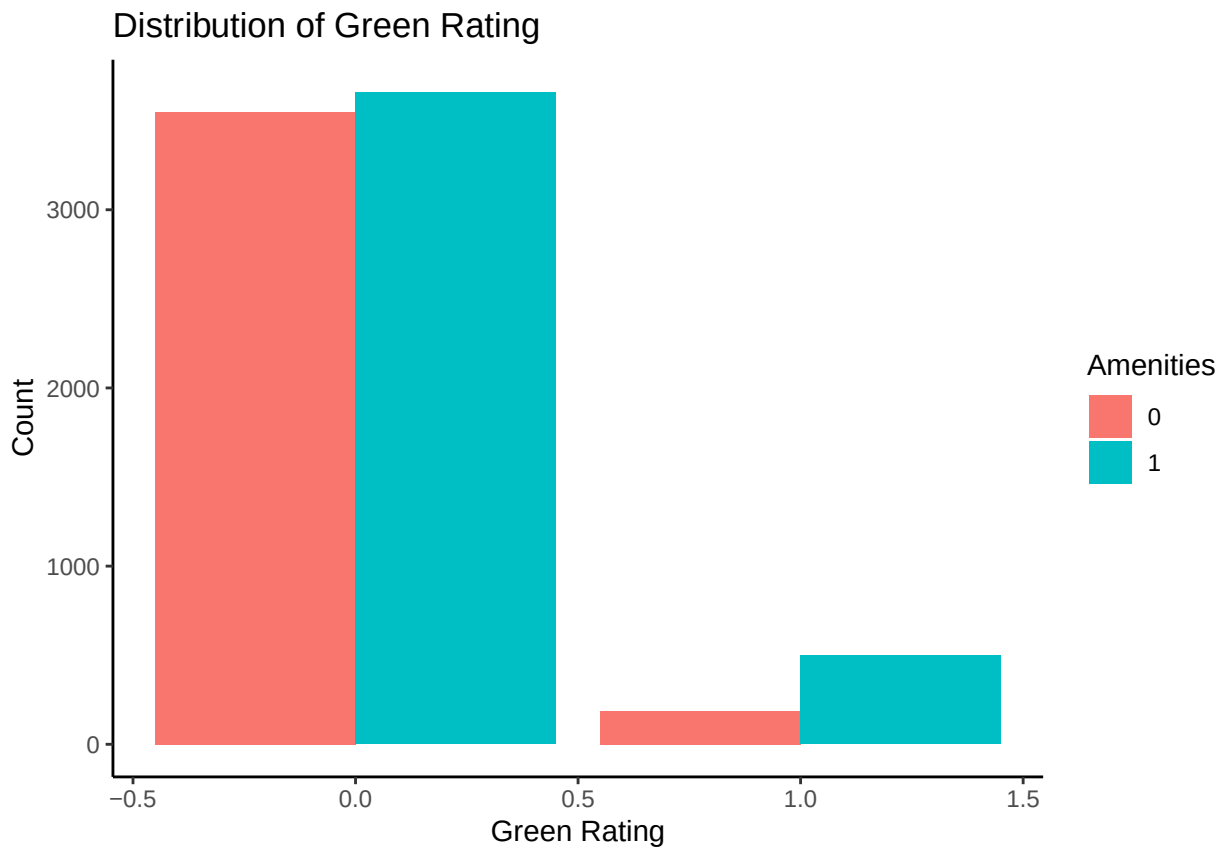
Here we can see a majority of the green buildings are newer. This shows us the Guru can also be wrong because he assumes the rent is higher because the buildings are green, when it could also be because the buildings are newer. Age is a confounding variable, and should have been taken into consideration.

The assumption made by the guru is wrong, the building would pay for itself faster as time when on and rent increased gradually.

```
## `geom_smooth()` using formula 'y ~ x'
```



To see if I could grab another confounding variable, I attempted using amenities. The plot doesn't go so well, but I am saying the cost of rent is higher where there are more amenities. Next I have a bar chart to show green rating.



Here, we can see the green buildings have more amenities, and rent is higher in both green buildings, and buildings with more amenities, making amenities another confounding variable.

All and all, the Guru was wrong for making quick assumptions of the data without trying to find if his correlation was true regarding other variables and if other variables play a factor.

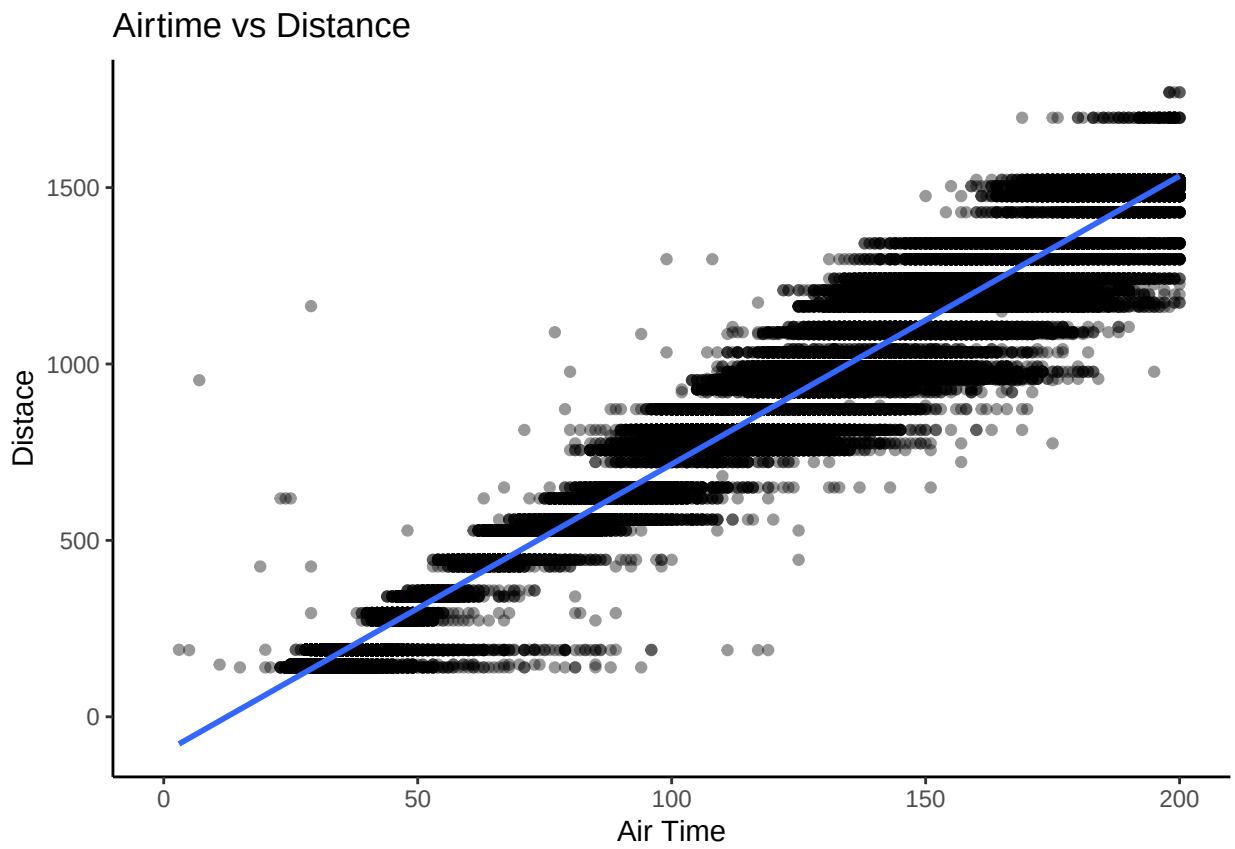
Question 2: Visual Story Telling (Part 2): flights at ABIA

To start, I wanted to see how Airtime and Distance were related. I assumed the more Airtime, the longer the Distance.

```
## `geom_smooth()` using formula 'y ~ x'
```

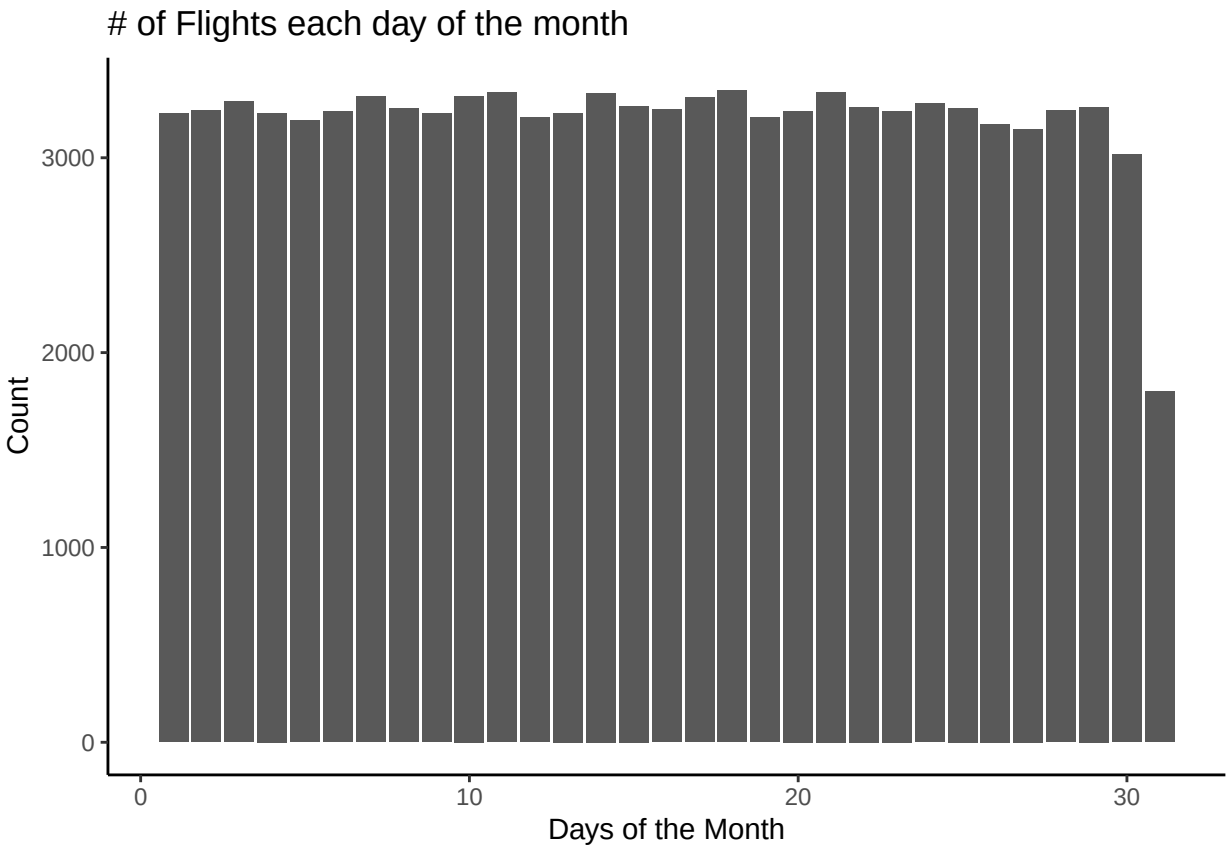
```
## Warning: Removed 6757 rows containing non-finite values (stat_smooth).
```

```
## Warning: Removed 6757 rows containing missing values (geom_point).
```

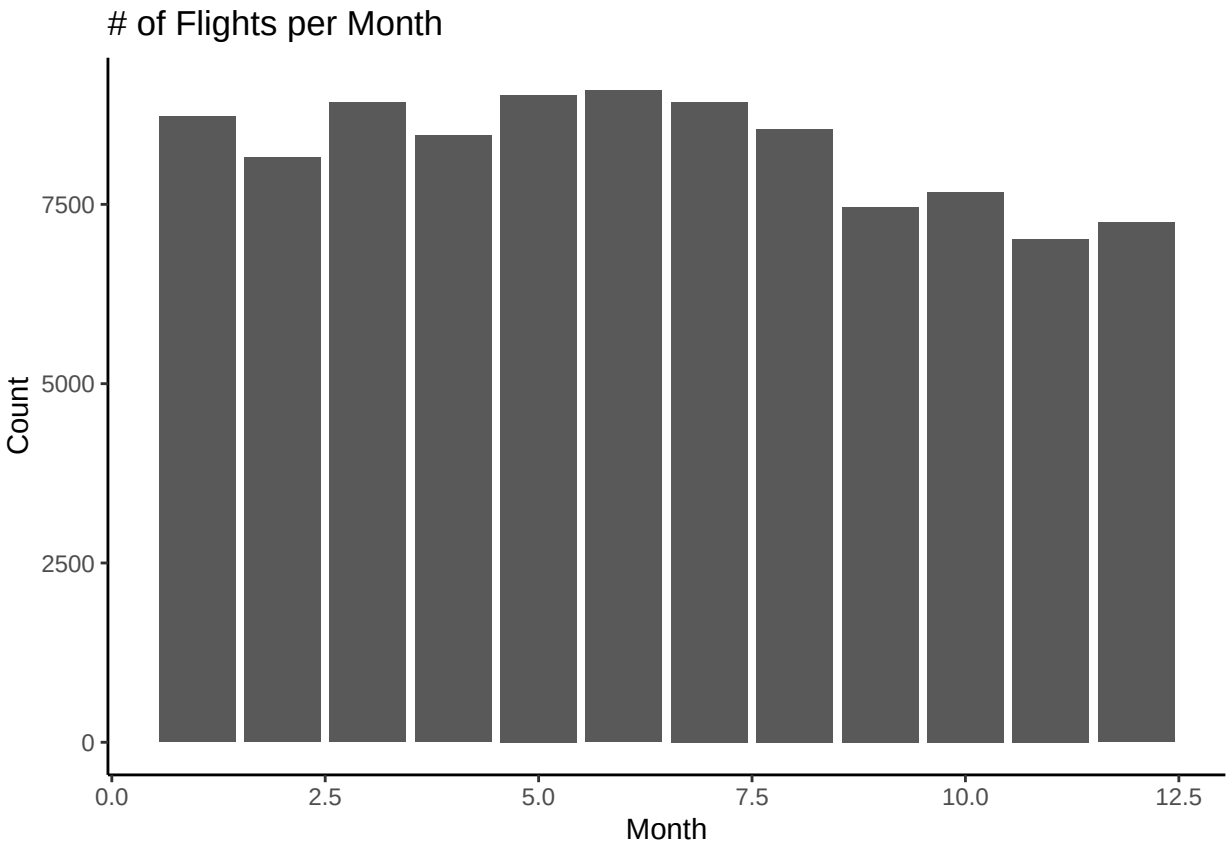


And here we can see that is true. So far, no fancy changes or crazy assumptions.

Next I wanted to plot the number of flights per day on each day of the month.



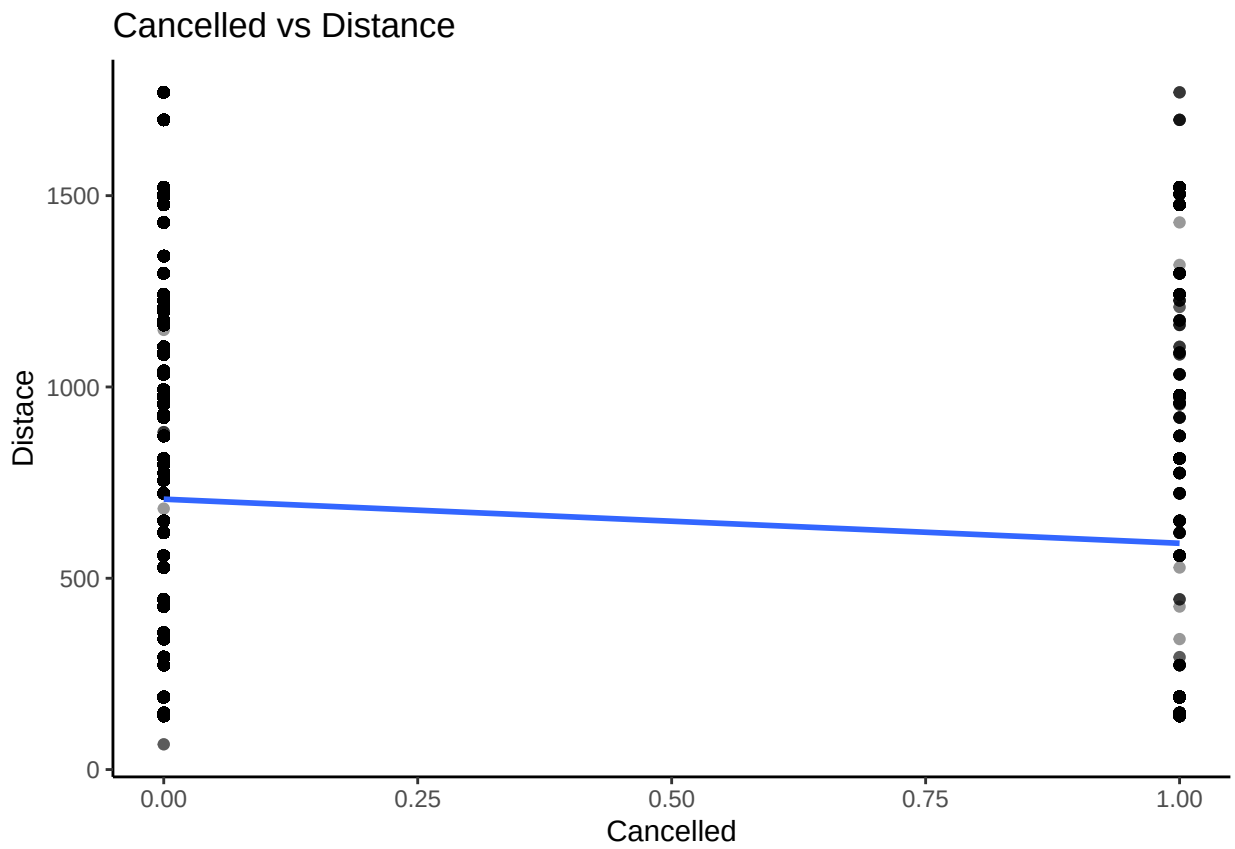
I had to use a bar chart, as my other charts were not playing nice. Here it looks like we have a lower rate of flights on the 31st. If you didn't think about it much, you might assume this means not many people fly at the end of the month, but really, its more of a factor when you think of how many months of a 31st day in them compared to others. Only 7 out of the 12 months of the year have a 31st day and here we can see that data align well as its a little over half of the rest of the days.



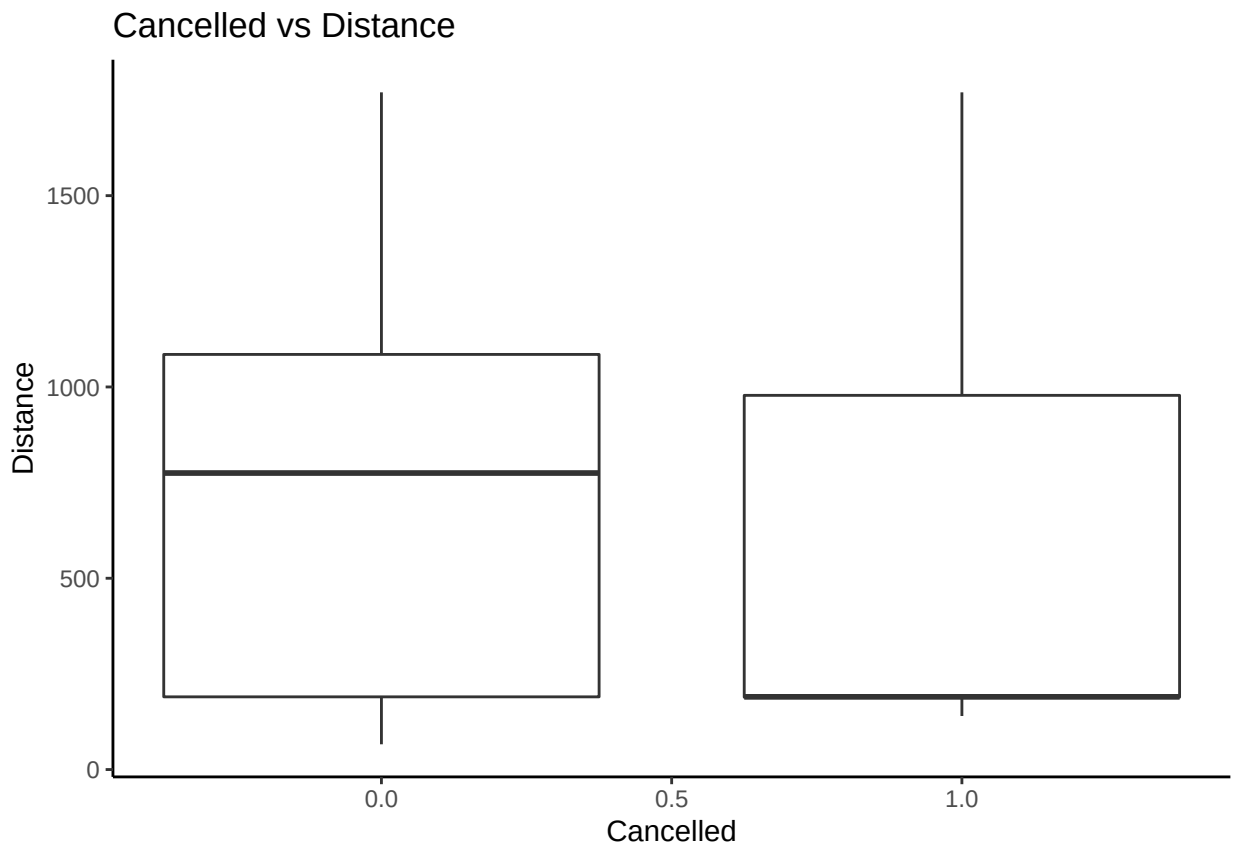
So we did days of the month, here is flights per month. We can see general trends, like increased traffic during holiday times as well as summer time.

Now we have my favorite change. I wanted to plot cancelled flights against distance. I assumed the longer the flight, the more likely it was to get cancelled. The longer a plane is in the air, the high its chance of other variables coming in and causing problems, like weather or traffic delays.

```
## `geom_smooth()` using formula 'y ~ x'
```



Here we see the opposite. Shorter flights actually have higher chances of being cancelled.



It becomes more apparent in the boxplot. This is different then I was expecting, meaning there are many more variables at play when it comes to short distance flights while long distance flights don't seem to be effected as much.

Question 3: Portfolio Modeling

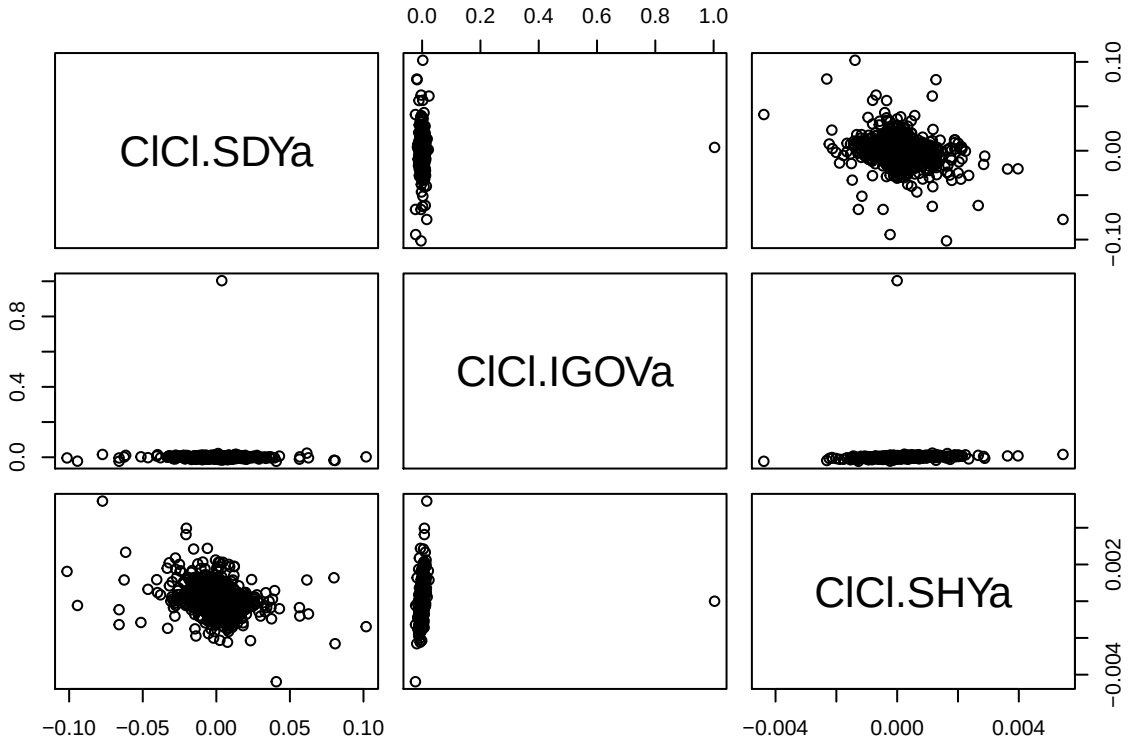
We have decided to choose three different portfolios with the following breakdowns:

Portfolio 1 (Low Risk Portfolio):

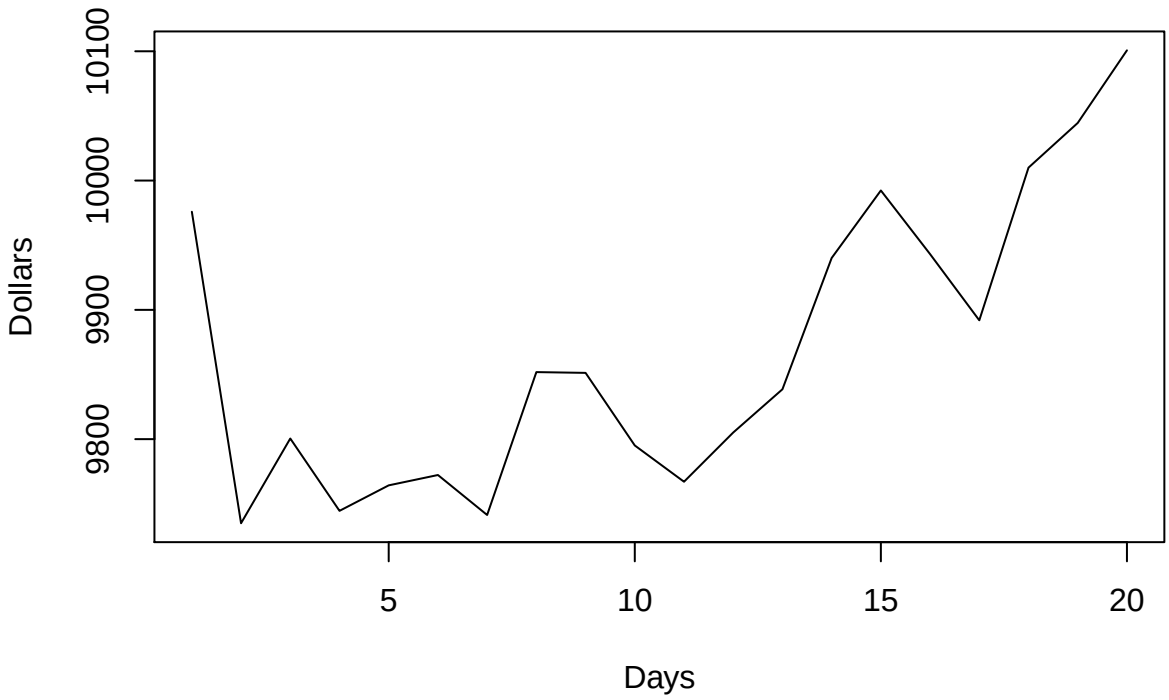
This portfolio is made up of index ETF's and bond ETF's. It is meant to represent a low risk investment strategy.

- SDY - SPDR S&P Dividend ETF (80%)
- IGOV - iShares International Treasury Bond ETF (10%)
- SHY - iShares 1-3 Year Treasury Bond ETF (10%)

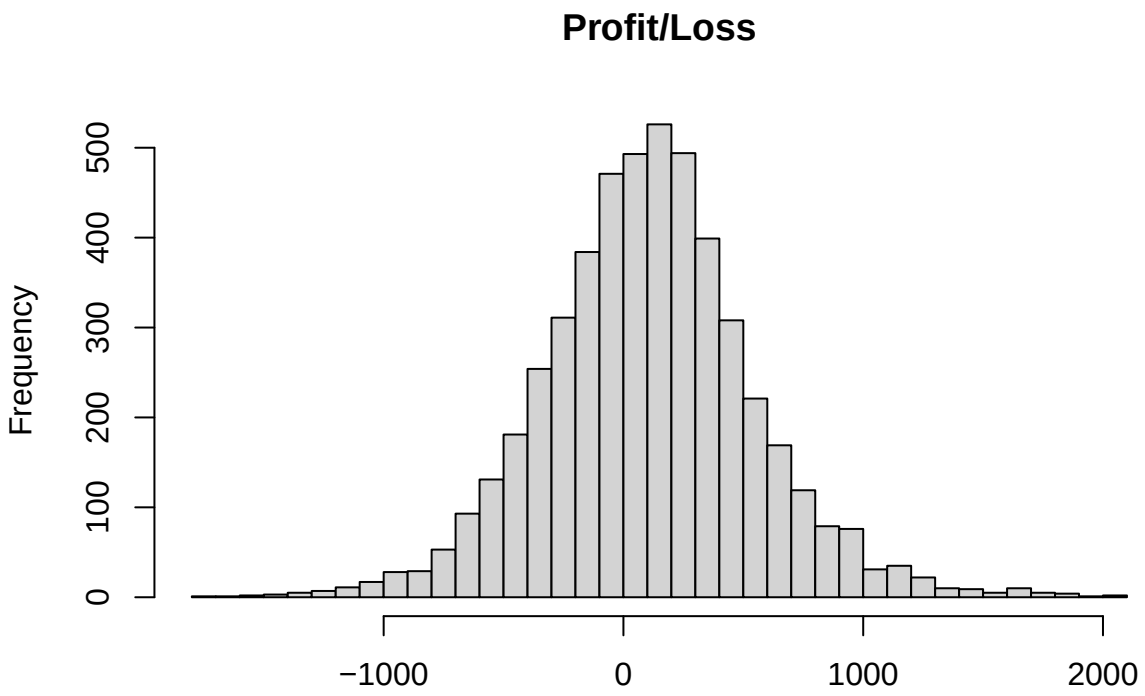
Below is a pairings plot of the selected ETFs:



We are running 5000, 20 trading day models. One of the models looks follows the graph seen below:



A summary of these 5000 models can be seen here:



The mean profit/loss was:

```
## [1] 105.7687
```

The 5% VaR was:

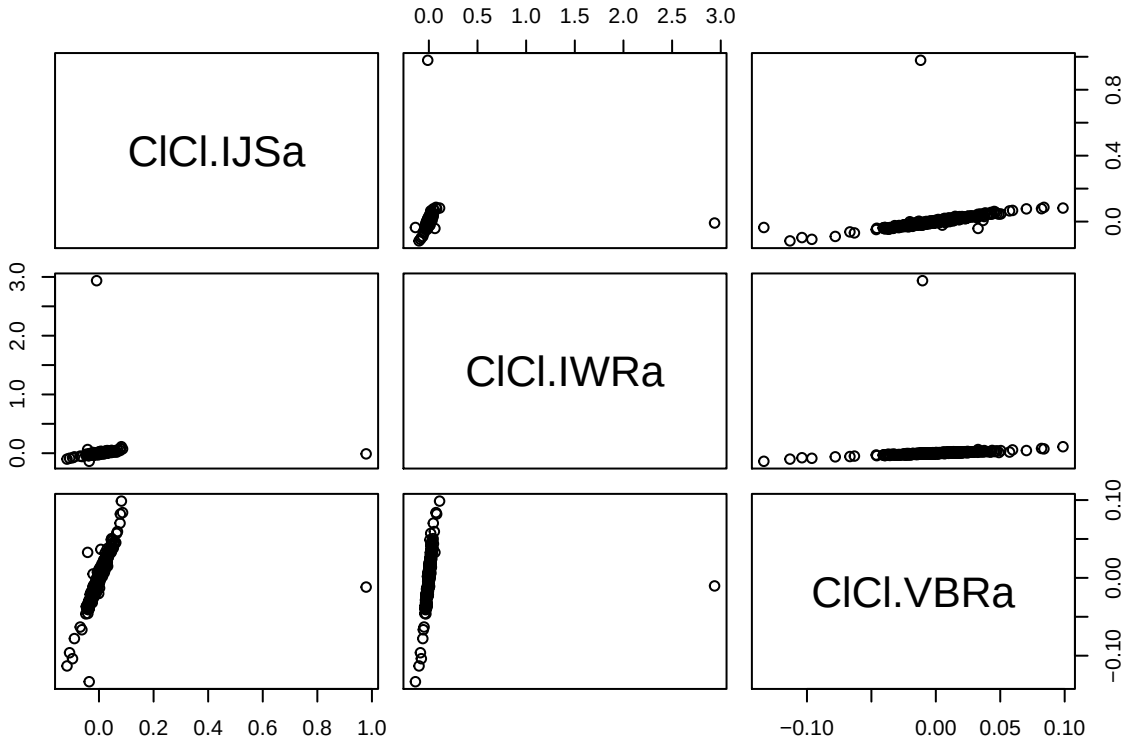
```
##        5%
## -597.3558
```

Portfolio 2 (High Risk):

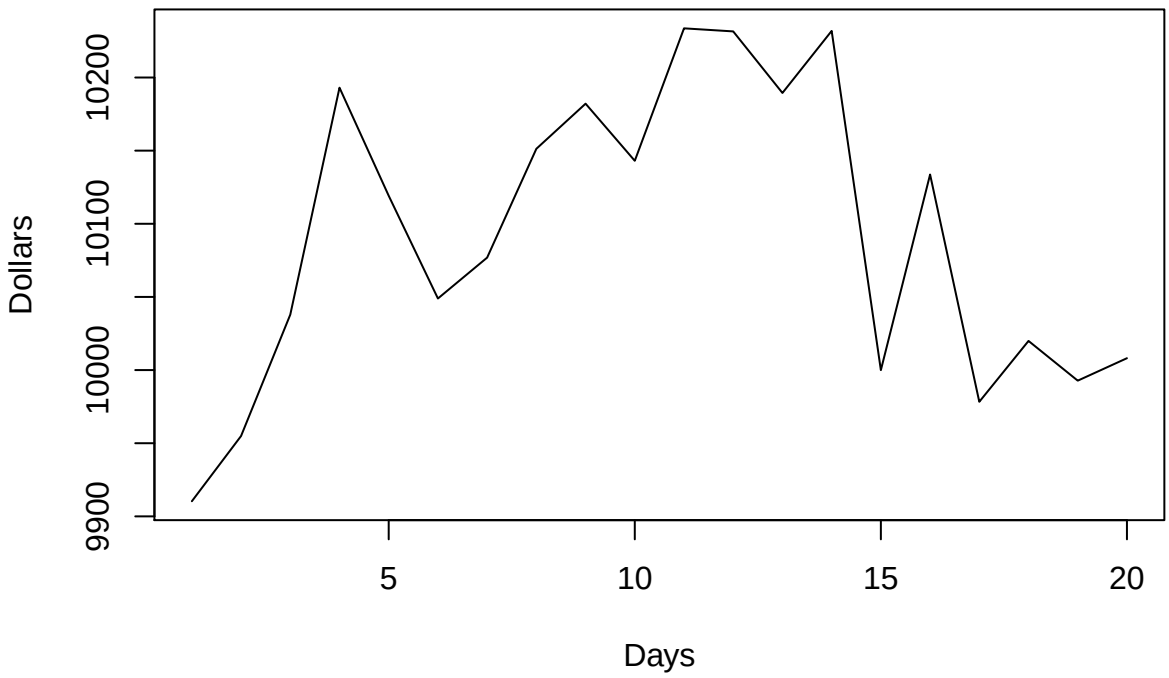
This portfolio is made up of small cap and mid cap index ETFs. It is meant to represent a high risk investment strategy.

- IJS - iShares S&P Small-Cap 600 Value ETF (40%)
- IWR - iShares Russell Mid-Cap ETF (30%)
- VBR - Vanguard Small-Cap Value Index Fund ETF (30%)

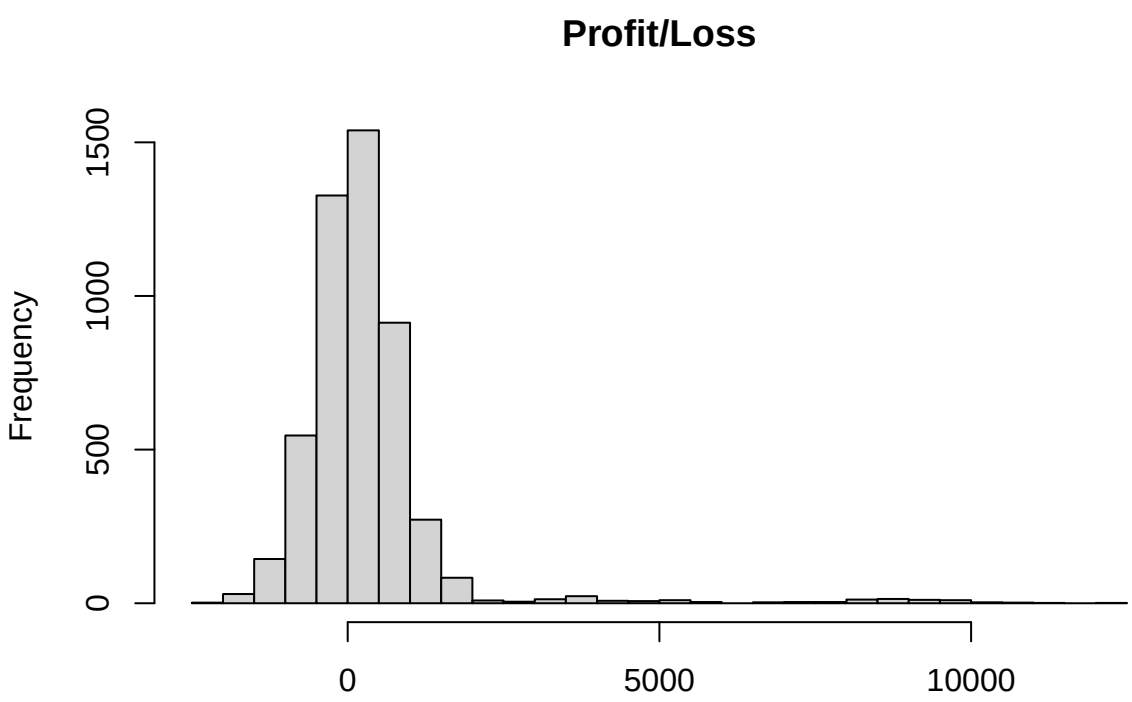
Below is a pairings plot of the selected ETFs:



We are running 5000, 20 trading day models. One of the models looks follows the graph seen below:



A summary of these 5000 models can be seen here:



The mean profit/loss was:

```
## [1] 288.5052
```

The 5% VaR was:

```
##        5%
## -893.4378
```

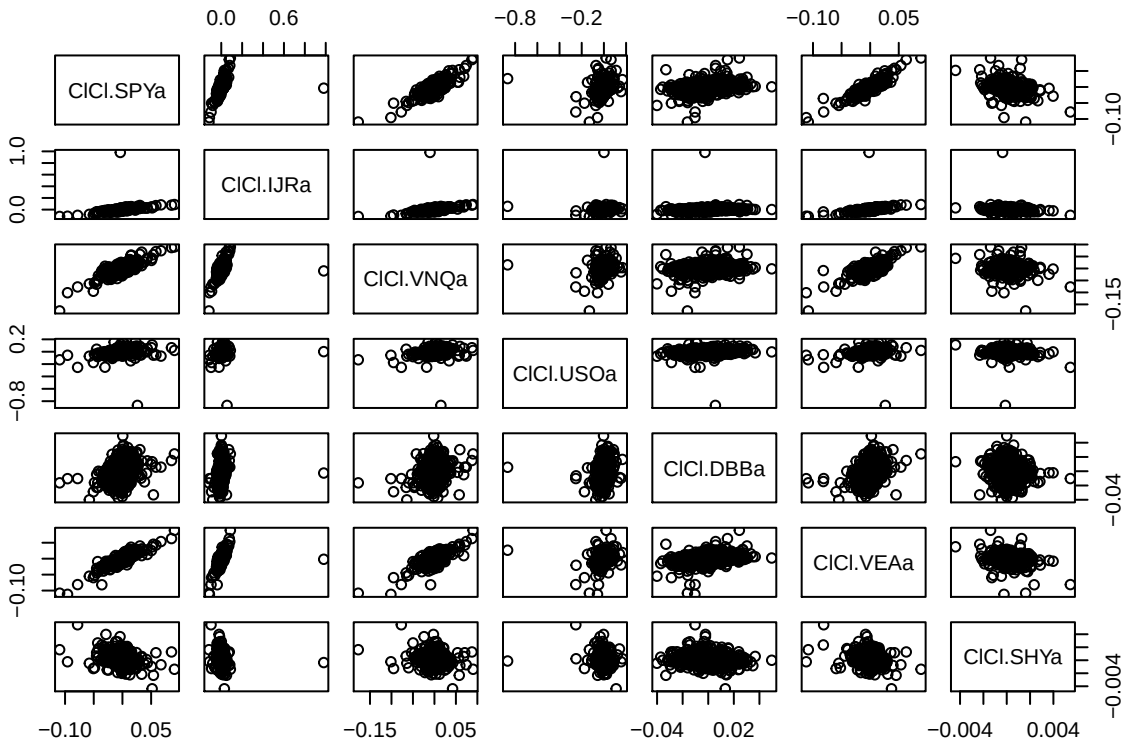
Portfolio 3 (Diverse):

This portfolio is made up of a diverse range of ETFs. It is meant to represent a highly diversified investment strategy.

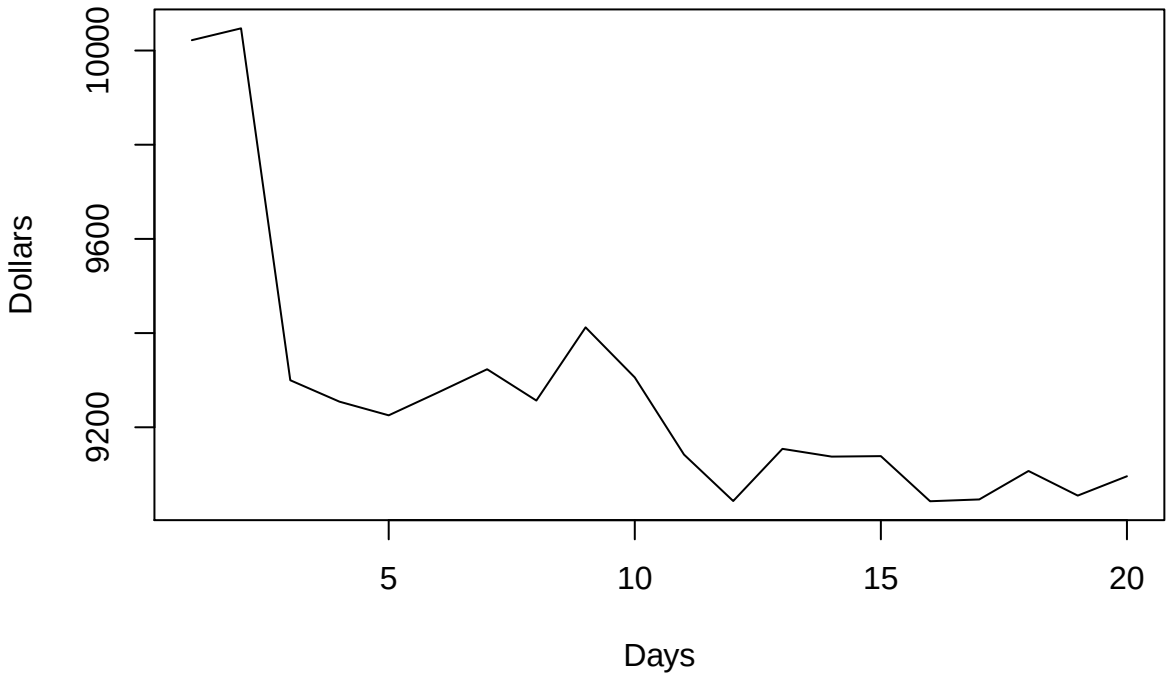
- SPY - SPDR S&P500 ETF (20%)
- IJR - iShares Core S&P Small-Cap ETF (20%)
- VNQ - Vanguard Real Estate Index Fund ETF (10%)
- USO - United State Oil Fund (10%)
- DBB - Investco DB Base Metals (10%)
- VEA - Vanguard Developed Markets Index Fund ETF (10%)
- SHY - iShares 1-3 Year Treasury Bond ETF (10%)

Below is a pairings plot of the selected ETFs:

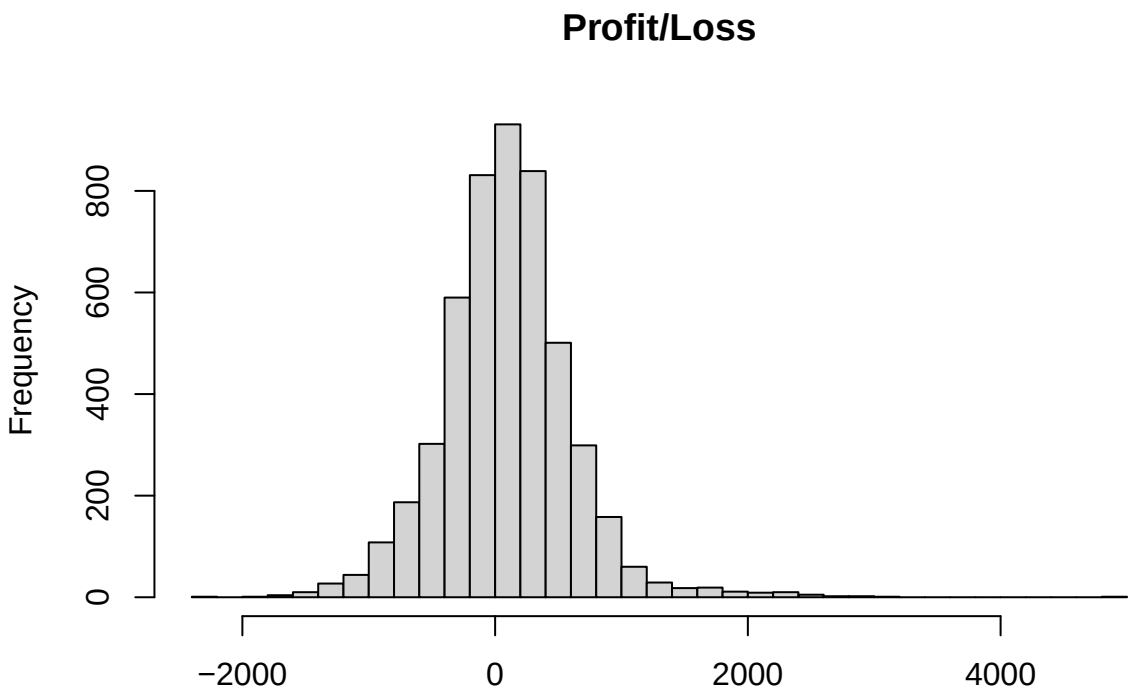
```
## pausing 1 second between requests for more than 5 symbols
## pausing 1 second between requests for more than 5 symbols
## pausing 1 second between requests for more than 5 symbols
```



We are running 5000, 20 trading day models. One of the models looks follows the graph seen below:



A summary of these 5000 models can be seen here:



The mean profit/loss was:

```
## [1] 90.87263
```

The 5% VaR was:

```
##        5%
## -728.5544
```

Question 4: Market Segmentation

One of the best ways to divide a market into different segments is through K-Means Clustering. This method allows us to split the data into like minded groups based on the information we are given.

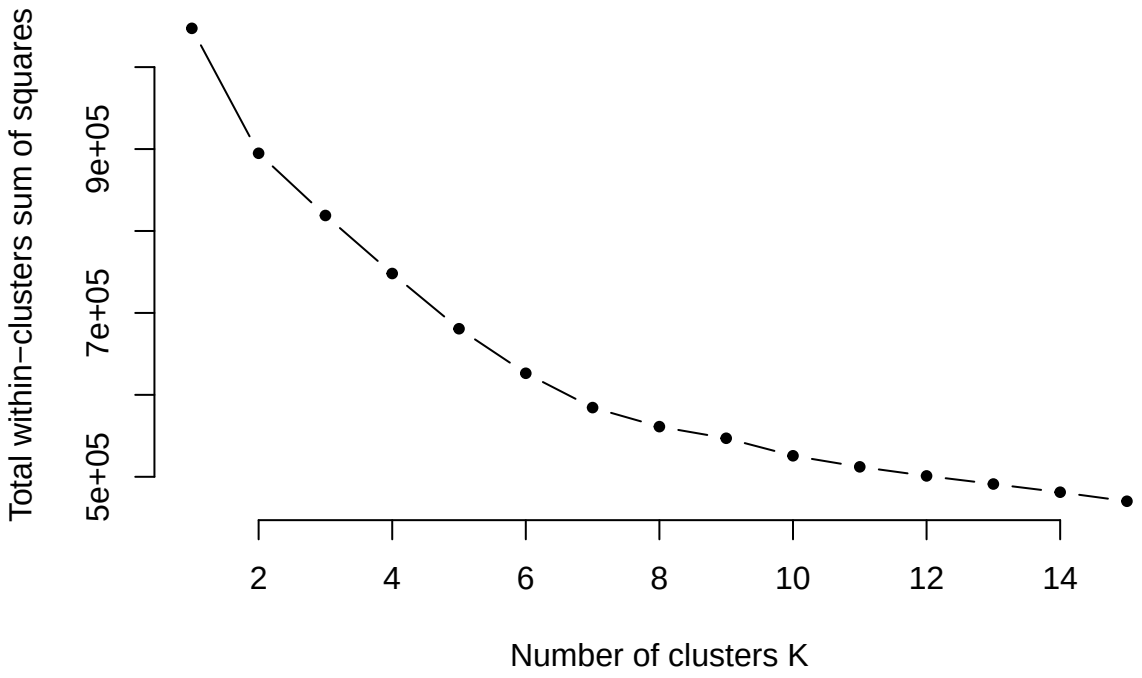
The first step is to look at the correlation matrix of the different variables in order to get a better understanding of possible patterns. We have included a sample of the correlation matrix below:

```
##                    chatter current_events      travel uncategorized    tv_film
## chatter        1.00000000     0.15621116  0.01466681  0.0669934396 0.01289863
## current_events 0.15621116     1.00000000  0.05094616  0.0296742259 0.07767346
## travel         0.01466681     0.05094616  1.00000000  0.0308655618 0.09698995
## uncategorized  0.06699344     0.02967423  0.03086556  1.0000000000 0.16327899
## tv_film        0.01289863     0.07767346  0.09698995  0.1632789878 1.00000000
## sports_fandom  0.01439263     0.06178037 -0.00870919 -0.0005287314 0.03075880
```

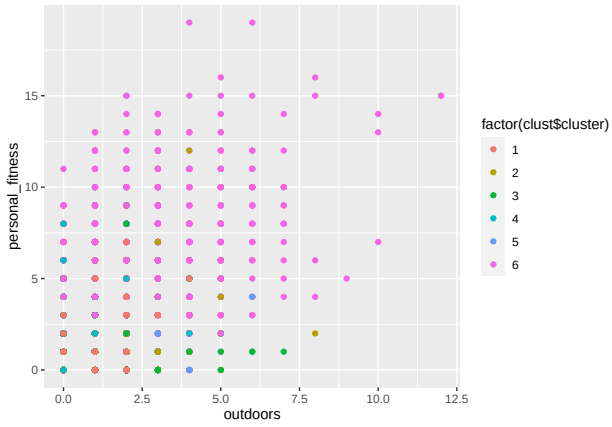
```
##                sports_fandom     politics        food       family
## chatter         0.0143926306  0.051448378 -0.00470408  0.079170381
## current_events  0.0617803698  0.068282733  0.05976753  0.063367006
## travel         -0.0087091896  0.660210000  0.07514222  0.017534763
## uncategorized  -0.0005287314 -0.001143143  0.03534767 -0.004628184
## tv_film         0.0307587998  0.032254541  0.08068332  0.021776392
## sports_fandom   1.0000000000  0.067097905  0.53263837  0.437810382
##                home_and_garden      music         news online_gaming   shopping
## chatter             0.07015760 0.09131777 -0.001163561   0.004784789 0.58337322
## current_events      0.05351146 0.07236614  0.060284226  -0.001645689 0.15014362
## travel              0.04093187 0.03864032  0.250616947   0.013222873 0.01990778
## uncategorized       0.07425914 0.14390770  0.003740334   0.023601681 0.05531255
## tv_film             0.10659105 0.27483223  0.067440558   0.035331802 0.04162464
## sports_fandom       0.08482199 0.05453751  0.200289676   0.024760877 0.02626127
##                health_nutrition college_uni sports_playing      cooking
## chatter             0.003727738  0.03399144     0.06107395 0.0020816423
## current_events      0.019863015  0.03095266     0.03072078 0.0467164788
## travel             -0.011922499  0.05383514     0.05496195 0.0175974884
## uncategorized       0.079855972  0.09471863     0.08383012 0.1612570088
## tv_film            -0.001790684  0.20421826     0.10326318 0.0006017955
## sports_fandom      -0.011229255  0.02645641     0.07104991 0.0076921174
##                      eco    computers   business     outdoors     crafts
## chatter        0.15511774  0.072489571 0.16610690 -0.006597319 0.11174145
## current_events 0.07690347  0.054824859 0.07458087  0.017898948 0.07372363
## travel         0.06143429  0.602934879 0.16184567  0.027133231 0.08695379
## uncategorized  0.04723203  0.026639216 0.06596542  0.093902277 0.08919657
## tv_film        0.06282547 -0.005485945 0.10114738  0.028928541 0.18468945
## sports_fandom  0.08585083  0.050633295 0.06813277  0.062217647 0.20118860
##                 automotive          art    religion     beauty    parenting
## chatter         0.13376735 -0.005076772 -0.02311170 0.02636423  0.020742516
## current_events  0.07244481  0.053477246  0.06769451 0.07011221  0.050237162
## travel         -0.00278411  0.086394257  0.06393306 0.01256552  0.042341130
## uncategorized   0.01391642  0.106447512  0.01703849 0.13737053  0.006268064
## tv_film         0.02050753  0.498771827  0.04502756 0.01678311 -0.001788090
## sports_fandom   0.23964647  0.022319534  0.63797484 0.12286324  0.607718120
##                     dating     school personal_fitness    fashion        adult
## chatter        0.184452453 0.11667862     0.0327477067 0.06331429  0.015372712
## current_events 0.031045863 0.06804713     0.0383886537 0.05590622  0.016764279
## travel         0.086756030 0.02219954    -0.0052992453 0.02601565  0.020244743
## uncategorized  0.127023808 0.05825994     0.0847295855 0.14140872  0.045196686
## tv_film        0.004170894 0.02502227    -0.0004447435 0.01760887 -0.021700249
## sports_fandom  0.016925530 0.49310619     0.0142720768 0.03076729  0.007986164
```

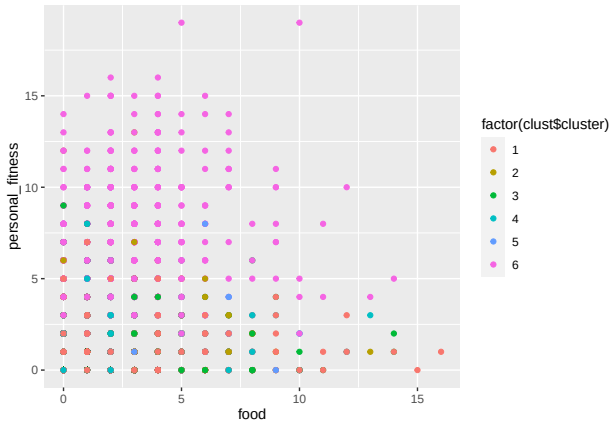
From there we can being to create the model. From the graph below we can see that a good value for K would be around 6. This allows the model to accuratley seperate the data without being too complex.

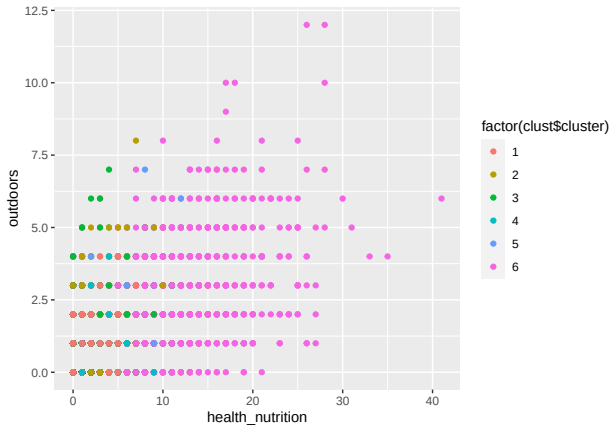
```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 394100)
```



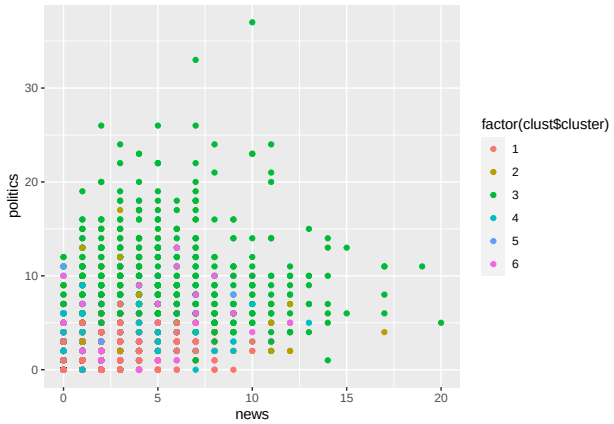
From the graphs below we can visualize our first cluster. This cluster contains variables such as Outdoors, Perosnal Fitness, and Health and Nutrition. This cluster is likely made up of health conscious individuals who enjoy the outdoors.

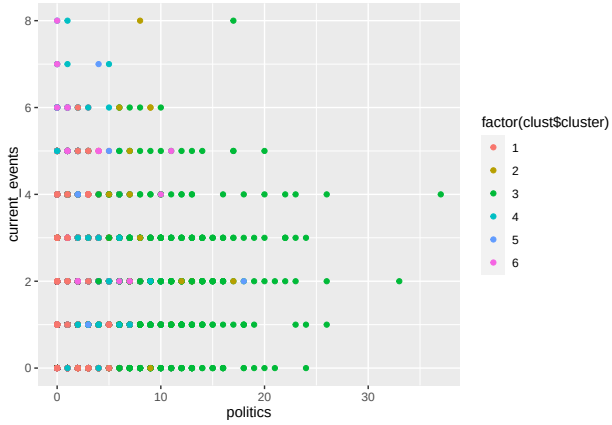


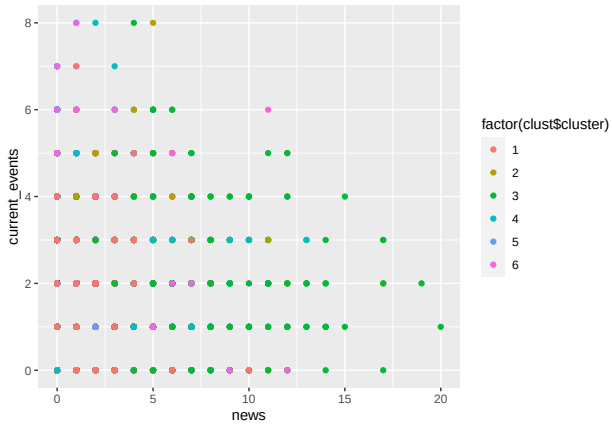




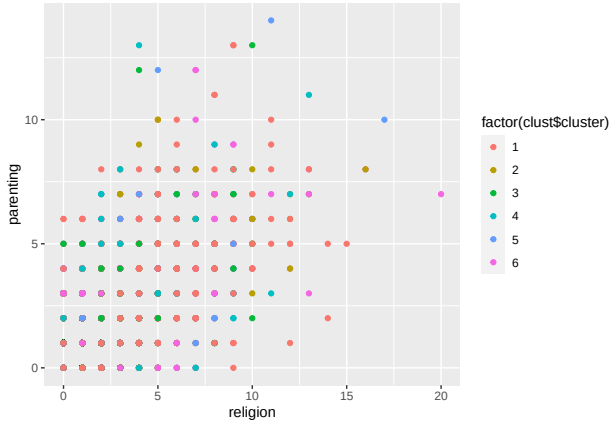
Another distinct group can be seen here and includes News, Politics, and Current Events. This cluster contains users who keep up with what is going on around the world and activley participate in discussion on these topics.

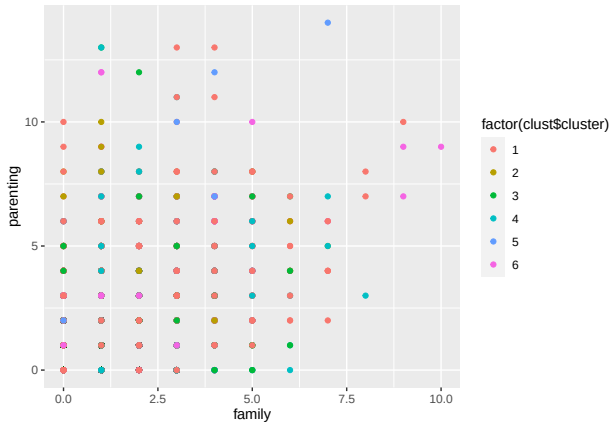


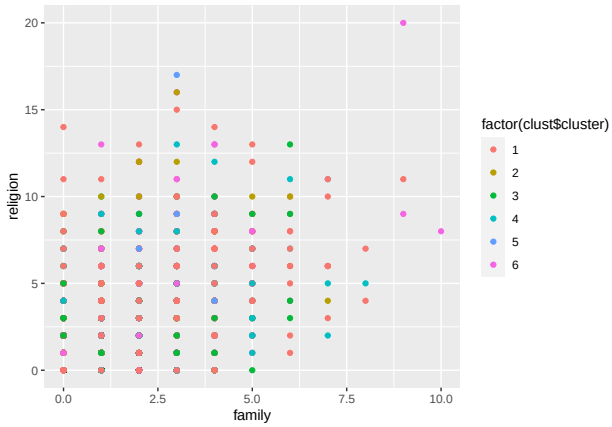




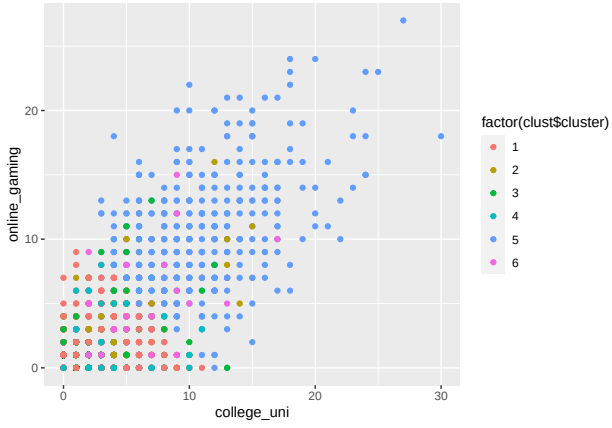
Middle aged users make up another group. This users engage in topics such as Relgion, Parenting, and Family.

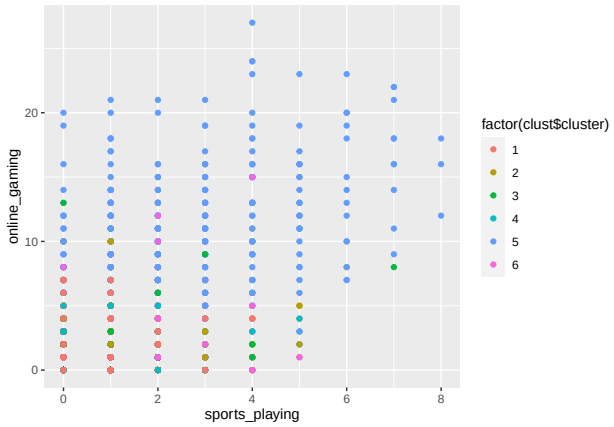


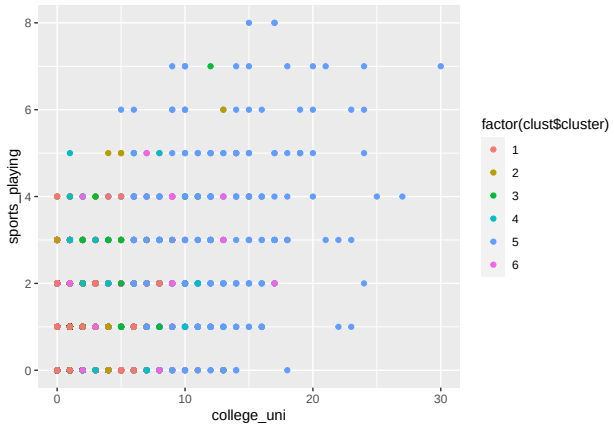




The final distict visual group is college aged users. These users engage in discussing colleges, sports, and online gaming.







When we compare the groups made above with the groups found through Hierarchical Clustering, we see some different results. There are many variables which are closely related and become one large group in the Hierarchical model which the K Means model was much more distinct.

```
##          chatter   current_events           travel    uncategorized
##                1                2                3                2
##         tv_film    sports_fandom         politics             food
##                2                2                3                2
##           family  home_and_garden            music             news
##                2                2                2                2
##    online_gaming         shopping health_nutrition      college_uni
##                4                2                5                4
##   sports_playing          cooking              eco        computers
##                2                6                2                2
##         business         outdoors           crafts       automotive
##                2                2                2                2
##              art         religion           beauty        parenting
##                2                2                2                2
##           dating           school personal_fitness          fashion
##                2                2                2                2
##            adult
##                2
```

Question 5: Author attribution

In this question, we need to build the best performing model to predict the author of an article on the basis of that article's textual content. We used Random Forests and Naive Base

We begin by loading in to the data using a for loop.

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

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

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

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

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

Second, we took some pre-processing/tokenization steps, including: 1) make everything lowercase 2) remove numbers 3) remove punctuation 4) remove excess white-space

```
## <<DocumentTermMatrix (documents: 2500, terms: 660)>>
## Non-/sparse entries: 224397/1425603
## Sparsity           : 86%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

Next we created DTM(doc-term-matrix) for the train data set. We have 2500 documents with 660 terms and a sparcity of 86%

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

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

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

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

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

```
## <<DocumentTermMatrix (documents: 2500, terms: 660)>>
## Non-/sparse entries: 225031/1424969
## Sparsity           : 86%
## Maximal term length: 18
## Weighting          : term frequency (tf)
```

```
## [1] 0.2624
```

```
## [1] 0.5976
```

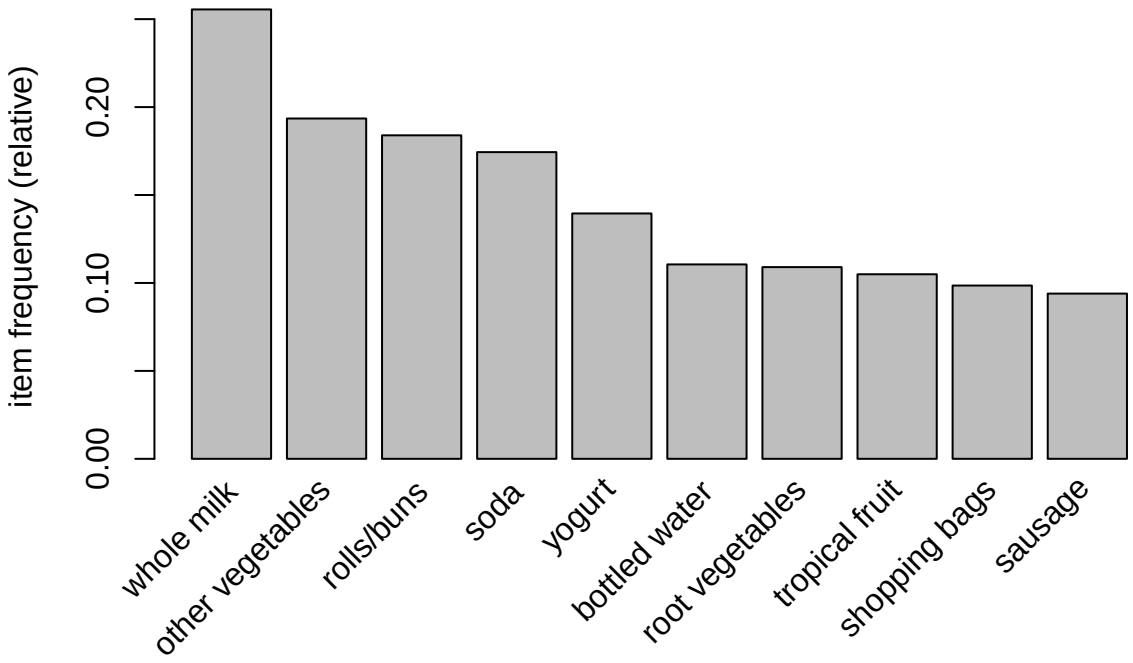
Finally, we pull the naive bayes analysis and get a 26% accuracy. This is not ideal compared to random forest. Random forest gets us an accuracy of 59% with 25 trees. This was our best model. Increasing or decreasing the number of trees only decreased accuracy. Based on this we think random forests is the best method for predicting the author of an article based on the articles textual content.

Question 6: Association Rule Mining

To start, we can take a small look into the dataset.

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

Here we can see the most frequent items bought, as well as the size of all the baskets in the dataset. Whole milk clearly dominates while Other Vegetables comes in a close 2nd.



Here we can see the frequency of the top 10 items found in carts. Whole milk comes up nearly 25% of the time in baskets with Other Vegetables coming in around 19%.

Moving forward to try and find association, we have a 125 rule set created by limiting support to 0.01 and confidence to 0.30.

```
## set of 125 rules
```

```
## set of 125 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 69 56
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   2.000   2.448   3.000   3.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01007   Min.   :0.3079   Min.   :0.01729   Min.   :1.205
##  1st Qu.:0.01149   1st Qu.:0.3454   1st Qu.:0.02888   1st Qu.:1.608
##  Median :0.01454   Median :0.3978   Median :0.03711   Median :1.789
##  Mean   :0.01859   Mean   :0.4058   Mean   :0.04783   Mean   :1.906
##  3rd Qu.:0.02217   3rd Qu.:0.4496   3rd Qu.:0.05663   3rd Qu.:2.155
##  Max.   :0.07483   Max.   :0.5862   Max.   :0.19349   Max.   :3.295
##      count
##  Min.   : 99.0
```

```
##  1st Qu.:113.0
##  Median :143.0
##  Mean   :182.8
##  3rd Qu.:218.0
##  Max.   :736.0
##
## mining info:
##  data ntransactions support confidence
##   df5          9835    0.01        0.3
```

Now we can see we have 69 times their is a correlation of one item to another, and 56 times we have 2 items that correlate to getting a 3rd item.

```
##      lhs                                rhs                support
## [1]  {citrus fruit,other vegetables}   => {root vegetables}  0.01037112
## [2]  {other vegetables,tropical fruit} => {root vegetables}  0.01230300
## [3]  {beef}                            => {root vegetables}  0.01738688
## [4]  {citrus fruit,root vegetables}    => {other vegetables} 0.01037112
## [5]  {root vegetables,tropical fruit}  => {other vegetables} 0.01230300
## [6]  {other vegetables,whole milk}     => {root vegetables}  0.02318251
## [7]  {curd,whole milk}                 => {yogurt}           0.01006609
## [8]  {rolls/buns,root vegetables}      => {other vegetables} 0.01220132
## [9]  {root vegetables,yogurt}          => {other vegetables} 0.01291307
## [10] {tropical fruit,whole milk}       => {yogurt}           0.01514997
##      confidence coverage   lift     count
## [1]  0.3591549  0.02887646 3.295045 102
## [2]  0.3427762  0.03589222 3.144780 121
## [3]  0.3313953  0.05246568 3.040367 171
## [4]  0.5862069  0.01769192 3.029608 102
## [5]  0.5845411  0.02104728 3.020999 121
## [6]  0.3097826  0.07483477 2.842082 228
## [7]  0.3852140  0.02613116 2.761356  99
## [8]  0.5020921  0.02430097 2.594890 120
## [9]  0.5000000  0.02582613 2.584078 127
## [10] 0.3581731  0.04229792 2.567516 149
```

So with this, I separated the ruleset by making the top rules those with higher lift. Lift being the likelyhood of someone buying the item because of the items they already have compared to the base rate. Those who buy citrus fruit and other vegetables are 3 times more liikely to buy root vegetables. What is interesting here is the 3rd rule. All of the rules of the top 10 by lift are rules that require at least 2 items before considering the 3rd, however the 3rd highest rule is only a one for one correlation. These shoppers are 3 times more likely to buy root vegetable when the person also buys beef. This makes sense if you think about how meat is commonly prepared, but the fact it beats out other teams of 2 is interesting. Another interesting element to this data is the infrequency of whole milk here. Whole milk was found to be the most common item by far, however in this comparison it only shows up in 3 combos. I had assumed milk would be all over the place, but in reality when it comes to association, people just seem to buy milk no matter what. Whereas the second highest item in terms of raw frequency is actually much more common to find in associations.