Predictive Modeling Exercises

Samuel Wu

Exercise 1

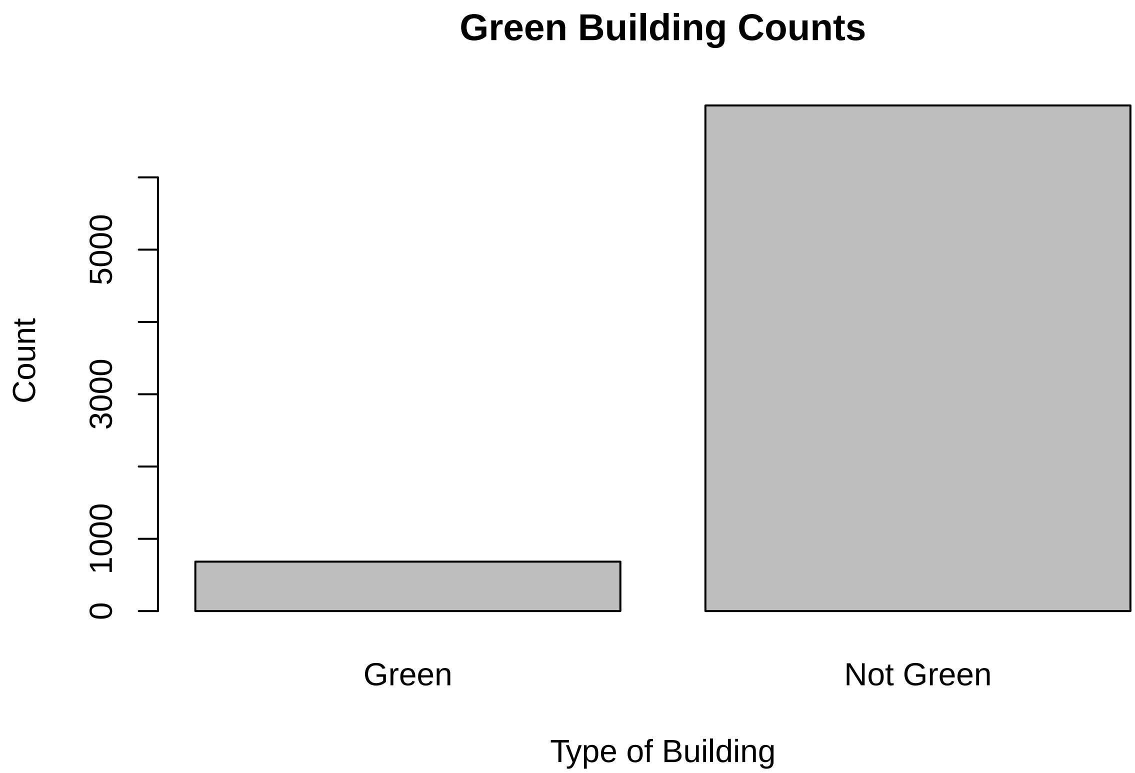
(Maybe we can look at the data removed to see why leasing was so low.) We aim to recreate the process done by the "guru" to confirm the validity of their analysis.

We first read in our data.

```
## -- Attaching packages --------------------------------------- tidyverse 1.3.0 --

## v ggplot2 3.3.2     v purrr   0.3.4
## v tibble  3.0.1     v dplyr   1.0.0
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Parsed with column specification:
## cols(
##   .default = col_double()
## )

## See spec(...) for full column specifications.
```

Now we clean the dataset to remove those with leasing rates lower than 10%. Afterward, we separate the data into green and non-green buildings.

We can visualize how many of each building type there are with a bar plot.



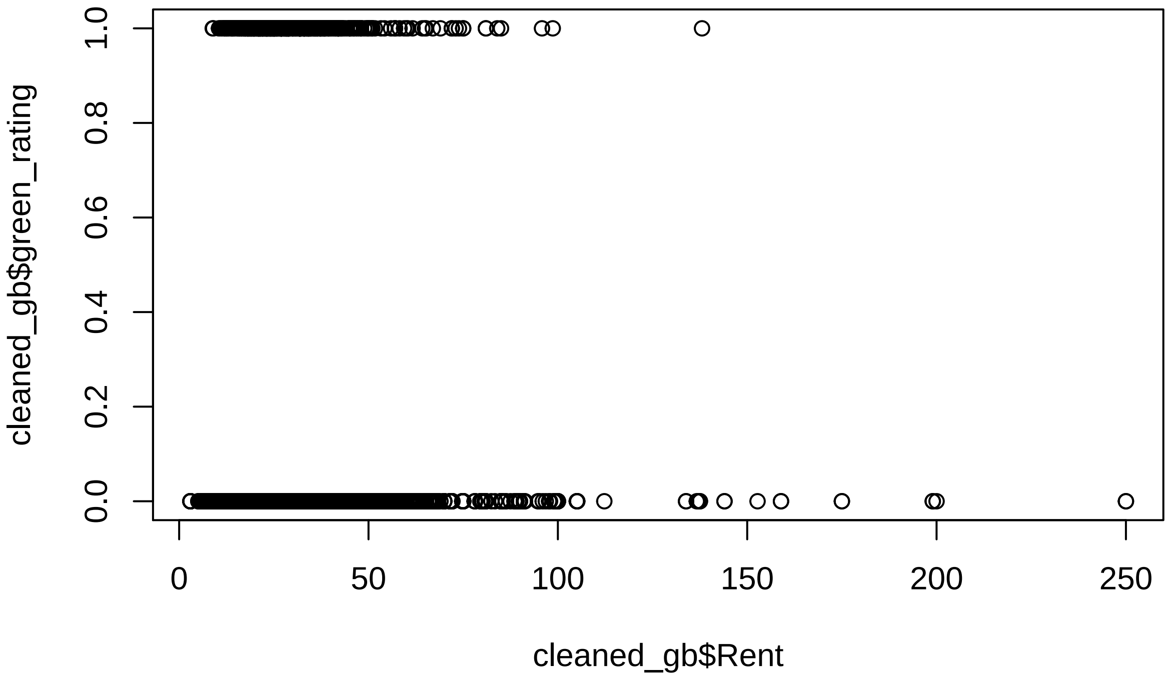
Now we'll check the median rent values of the two building types.

```
## [1] 27.6
```

```
## [1] 25.03
```

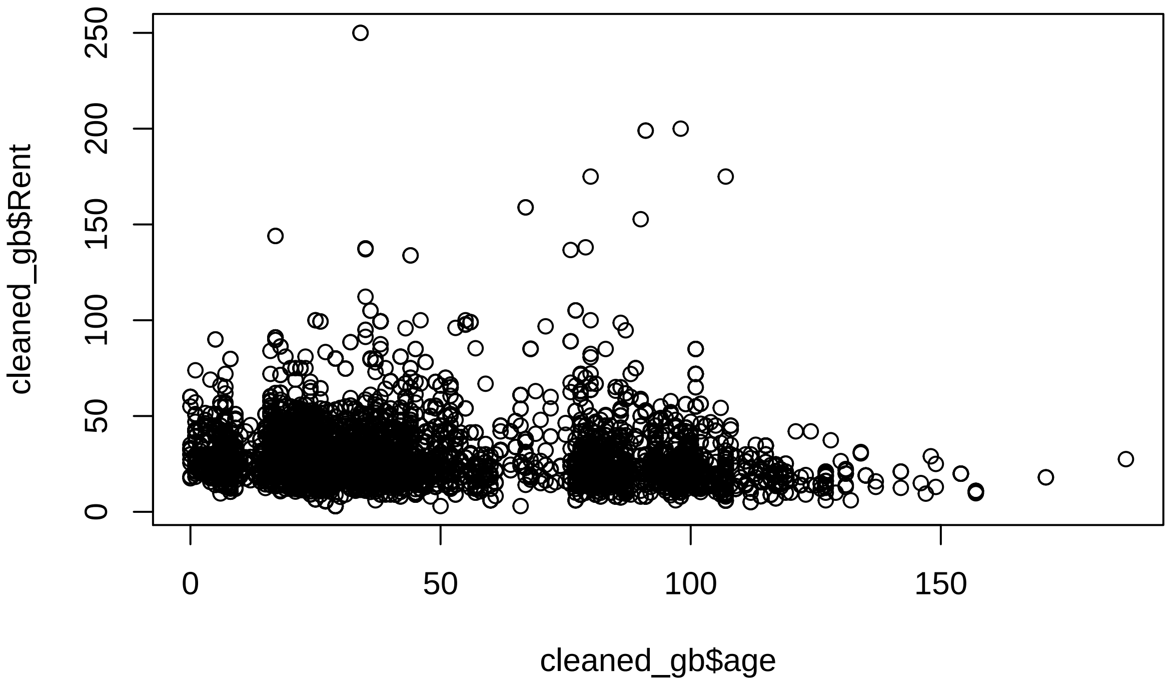
Here, we see the numbers align with what was reported before, with the exceptions of the green building's median rent being $25.03. However, this number is very close to what was reported, so the calculations that were initially reported are still a good representation of our future revenue.

We would also like to see if there are confounding variables in the relationship between Rent and green_status. To get a sense of what's going on, we first plot these two.

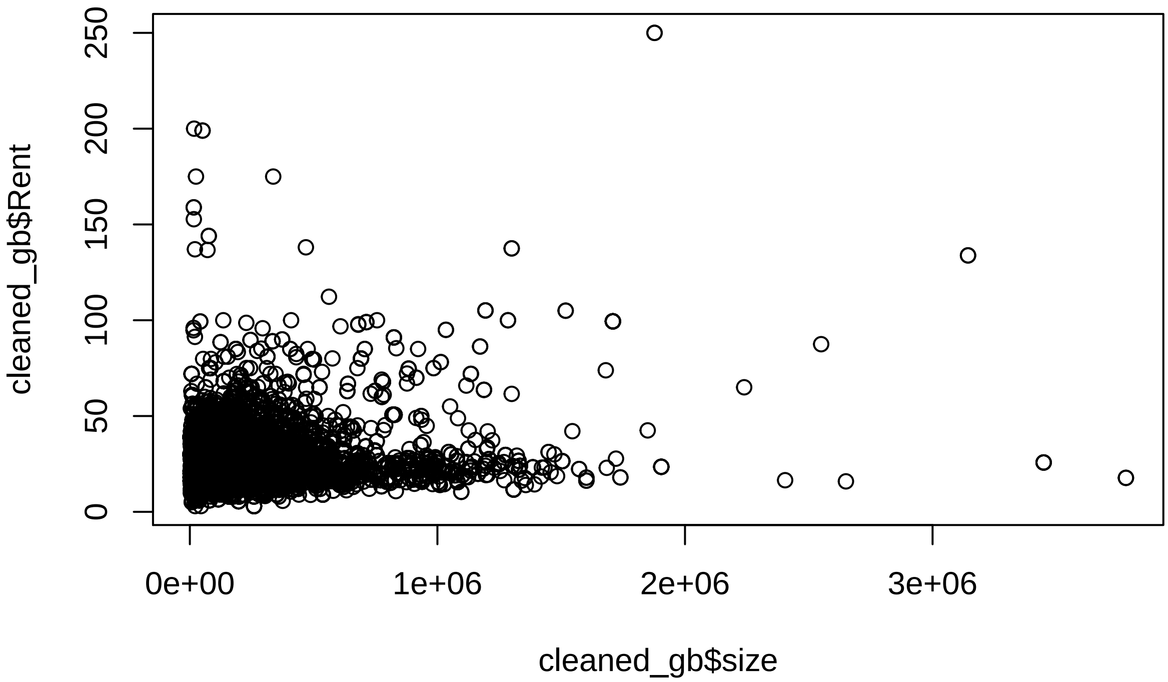


We notice that most of the green buildings have lower rents. So this could indicate a relationship to factors like building size and age.

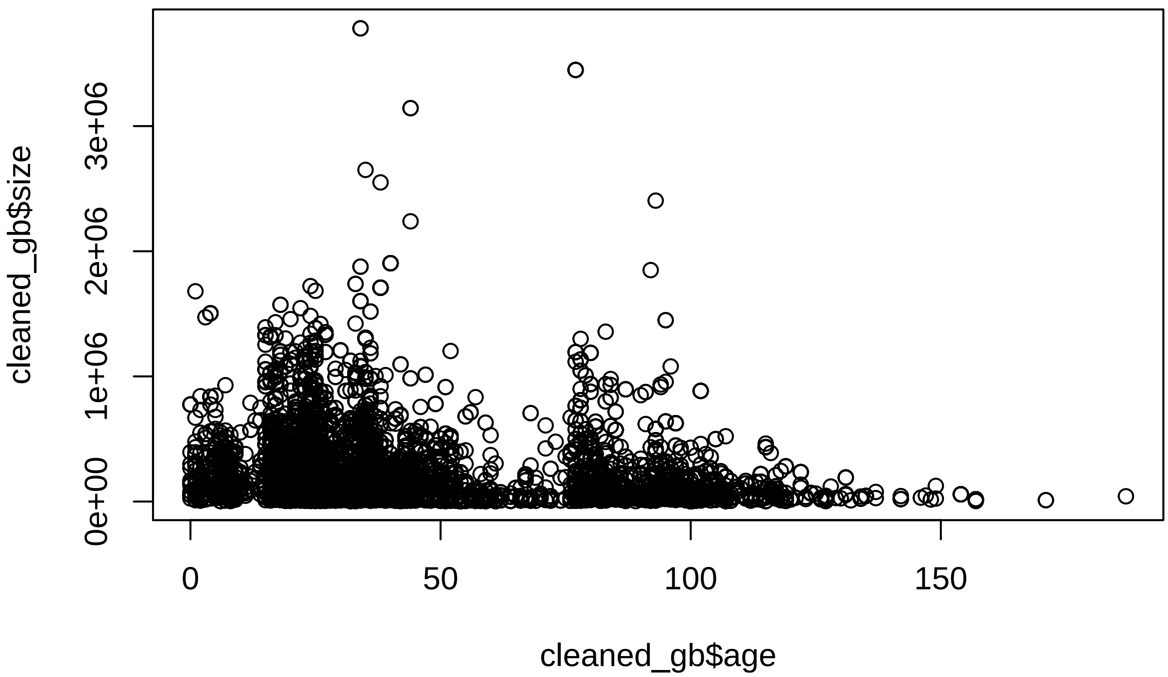
```
plot(cleaned_gb$age, cleaned_gb$Rent)
```




```
plot(cleaned_gb$size, cleaned_gb$Rent)
```




```
plot(cleaned_gb$age, cleaned_gb$size)
```



Summarizing the results above, we see that smaller apartments tend to lead toward cheaper rent, which makes sense since you'd pay less for less space. There doesn't seem to be a trend between rent and age. The older apartments are slightly smaller than their younger counterparts. The graph between age and size is interesting though because it seems like younger apartments have generally bigger sizes. This potentially shows a relationship between confounding variables. If we wanted to reduce the dimensionality of our problem, we could combine data like age and size into 1 variable or only use 1 in our analysis since the information from one column tells us something about the other. In turn, this could allow us to adjust for these confounders in our problem.

Exercise 2

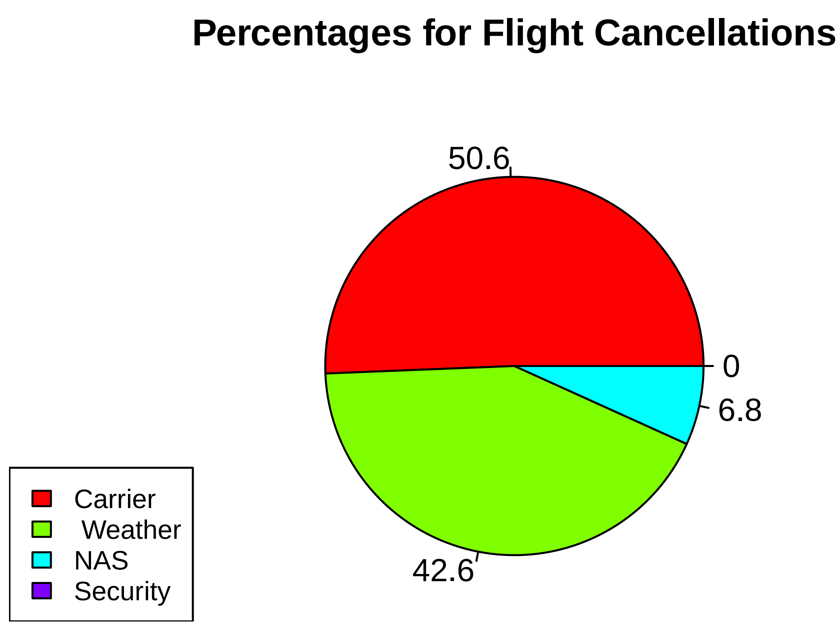
We aim to look at the relationships connected to cancelled flights, the reasons for them, and the days of the week they occur. First, we read in the data necessary.

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   UniqueCarrier = col_character(),
##   TailNum = col_character(),
##   Origin = col_character(),
##   Dest = col_character(),
##   CancellationCode = col_character()
## )
```

```
## See spec(...) for full column specifications.
```

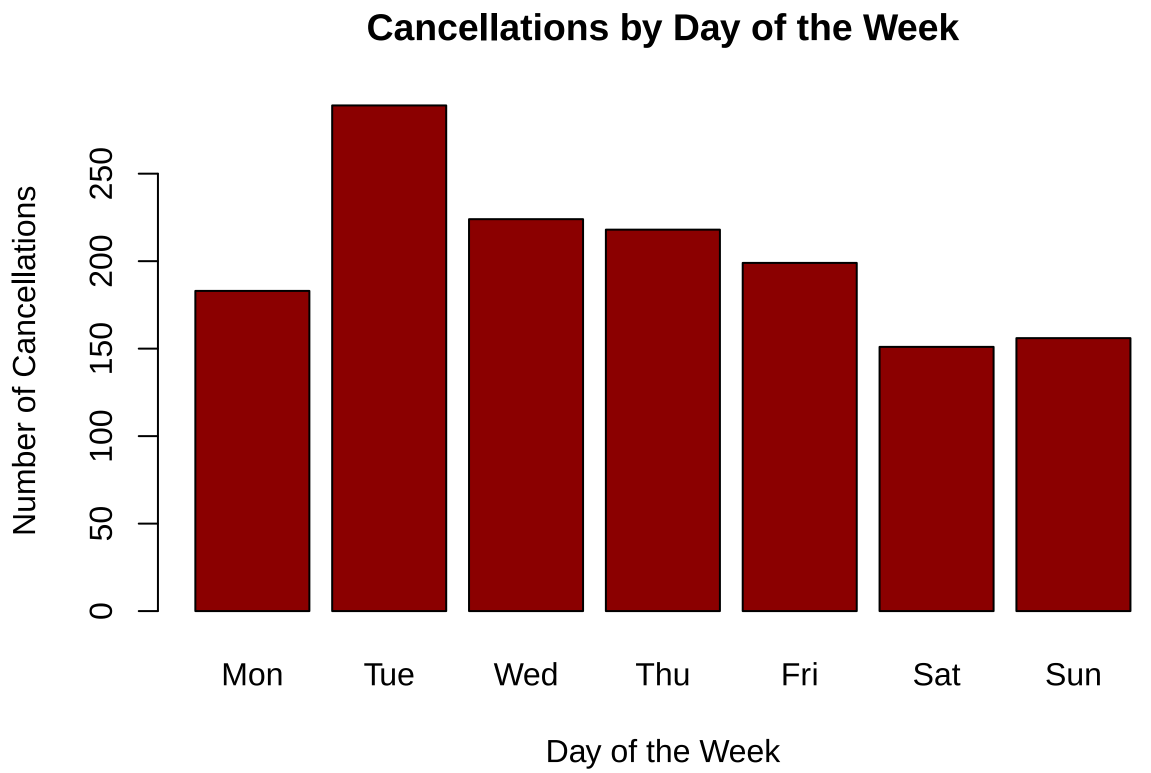
Out of the 99260 rows, we only have data on 1420 cancelled flights, but this can give us some insights still.

We first make a pie chart to see what percentage of these cancellations are due to the carrier, weather, NAS, or security.



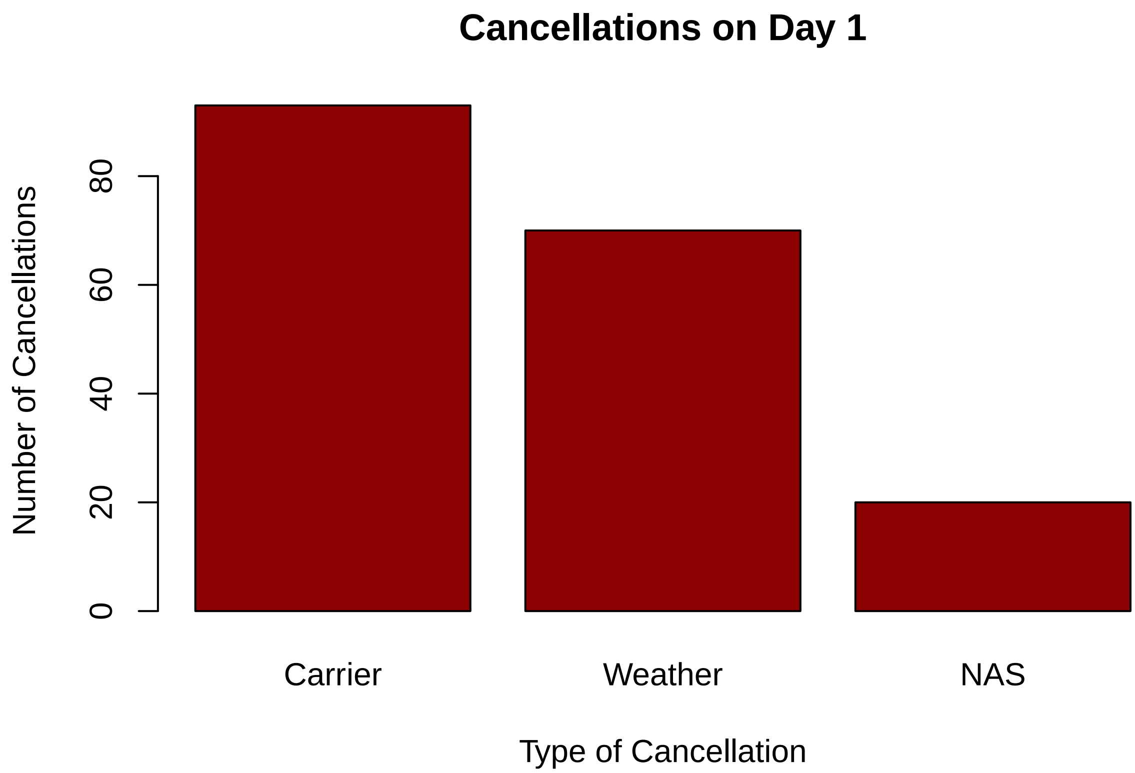
From our first pie chart, we see that 50.6% the cancellations at ABIA are due to carrier issues. This could be overbooked flights or other internal issues. This is followed by a 42.6% cancellation rate due to weather. NAS is a small issues compared to the others, and there were actually no security cancellations. Maybe this suggests Austin's airport is safe for fliers.

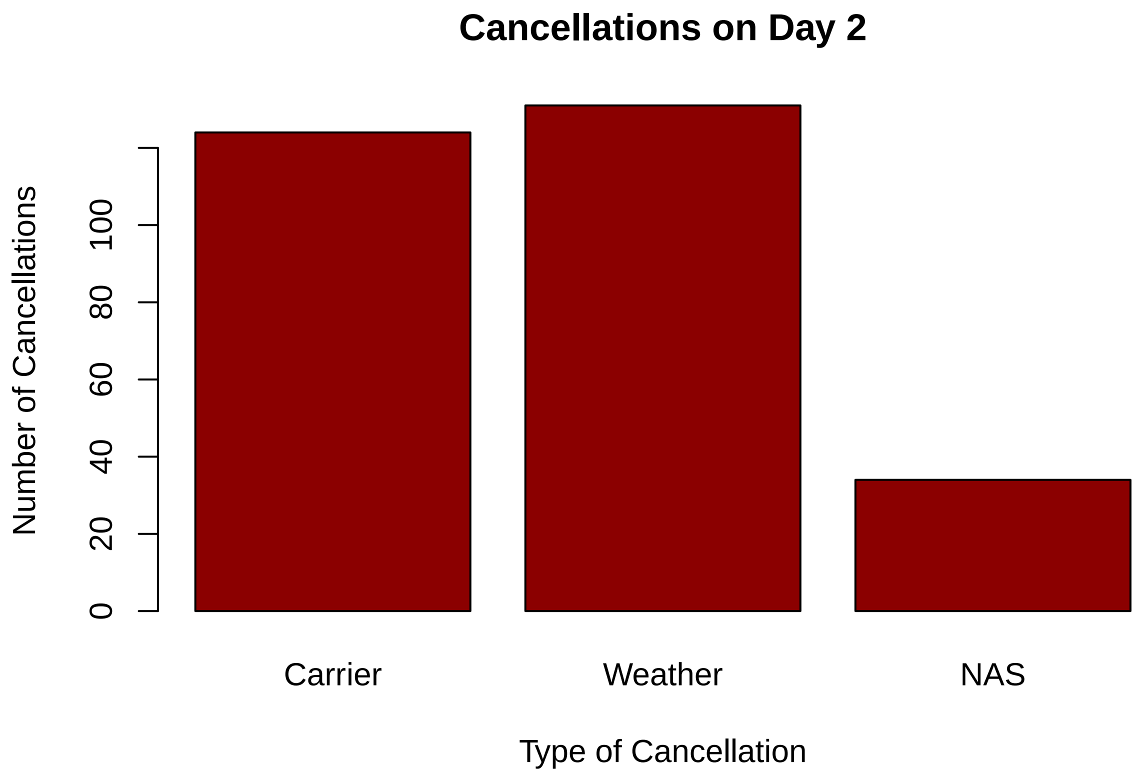
Our next goal is to see if there's a relation between the day of the week and these cancellations. We use a bar graph to illustrate.

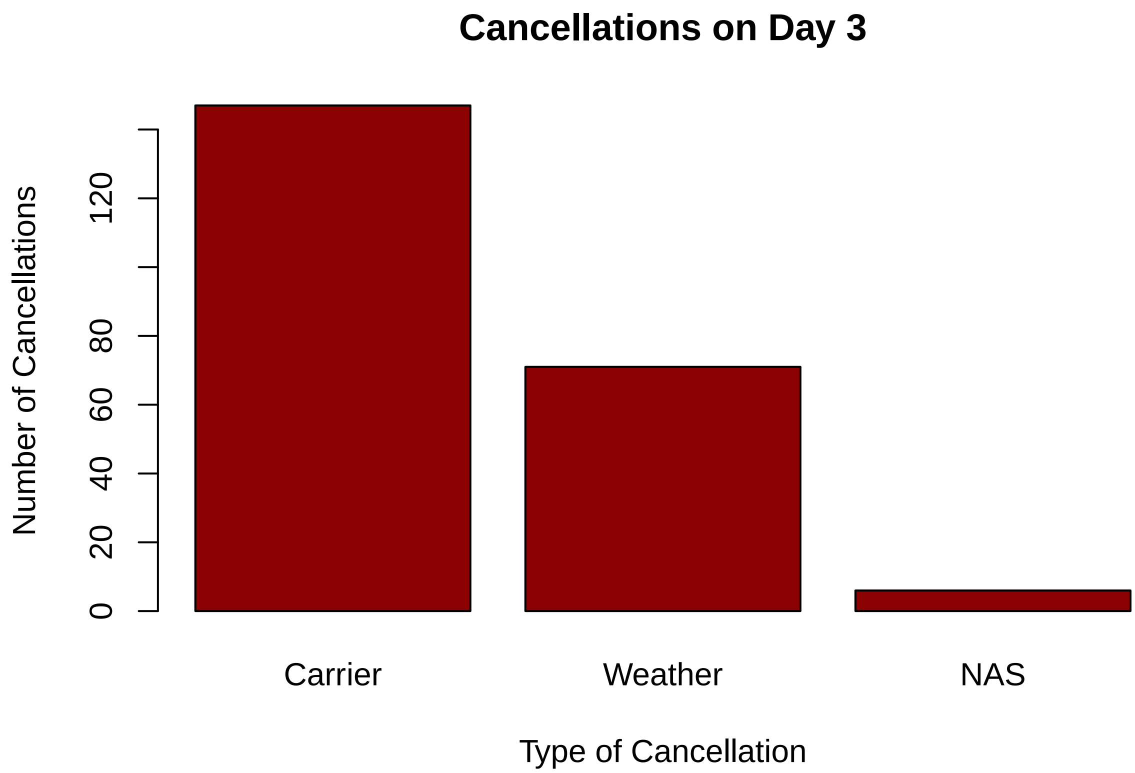


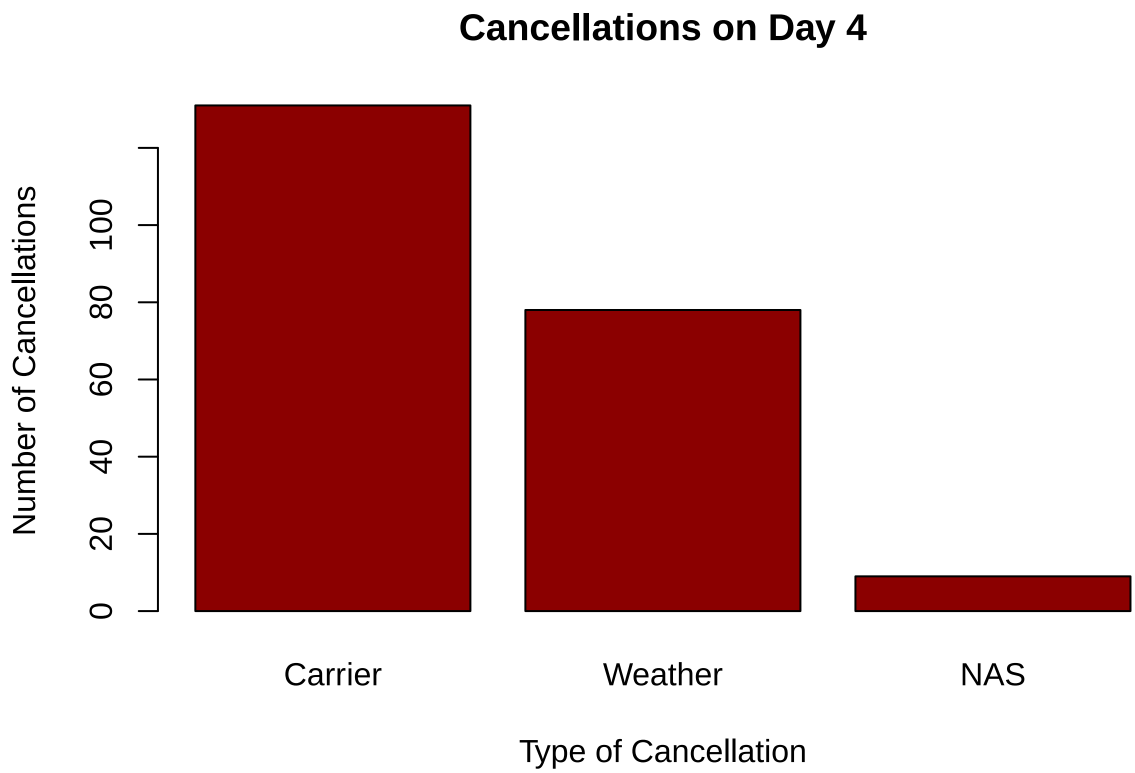
Our results show that Tuesday seems to have the most cancellations compared to the other days. It hsa 289 cancellations. The weekends (Saturday and Sunday) seem to have much lower values compared to the rest (151 and 156 respectively).

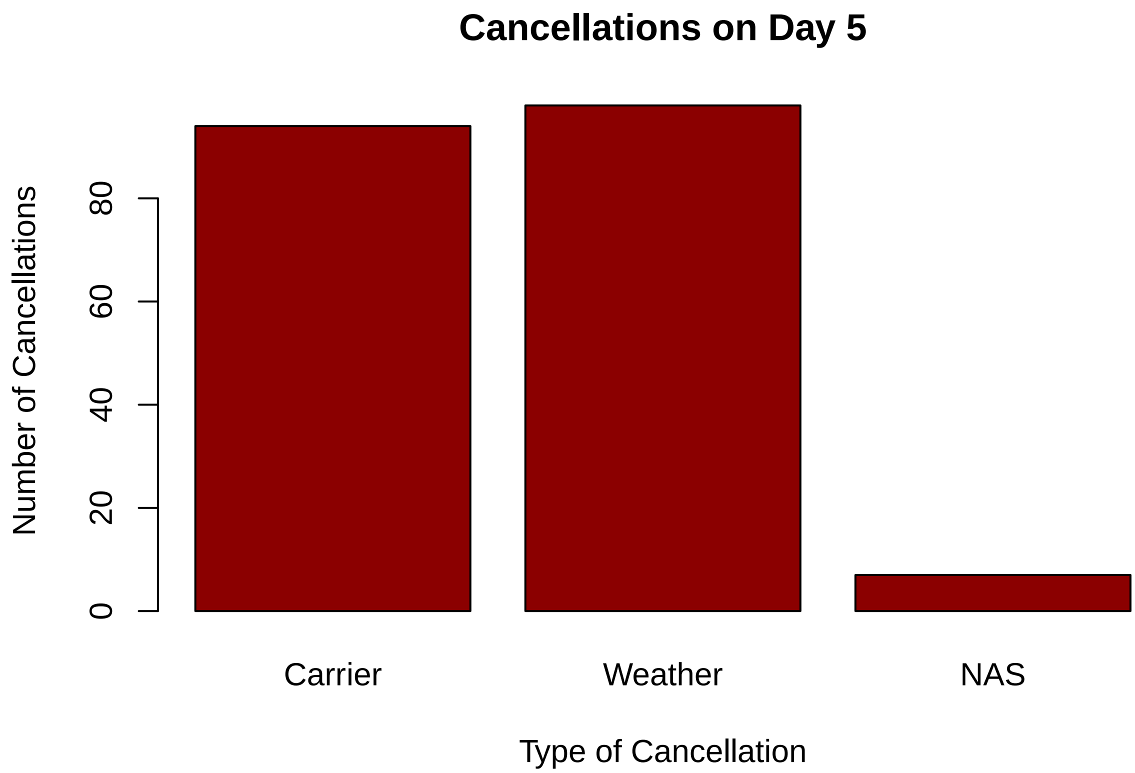
Now we aim to see if there's a relation between the day of the week and the cancellation type. (I think you just loop using the code aboce, just add a parameter to check for abcd, and let i be day of the week)

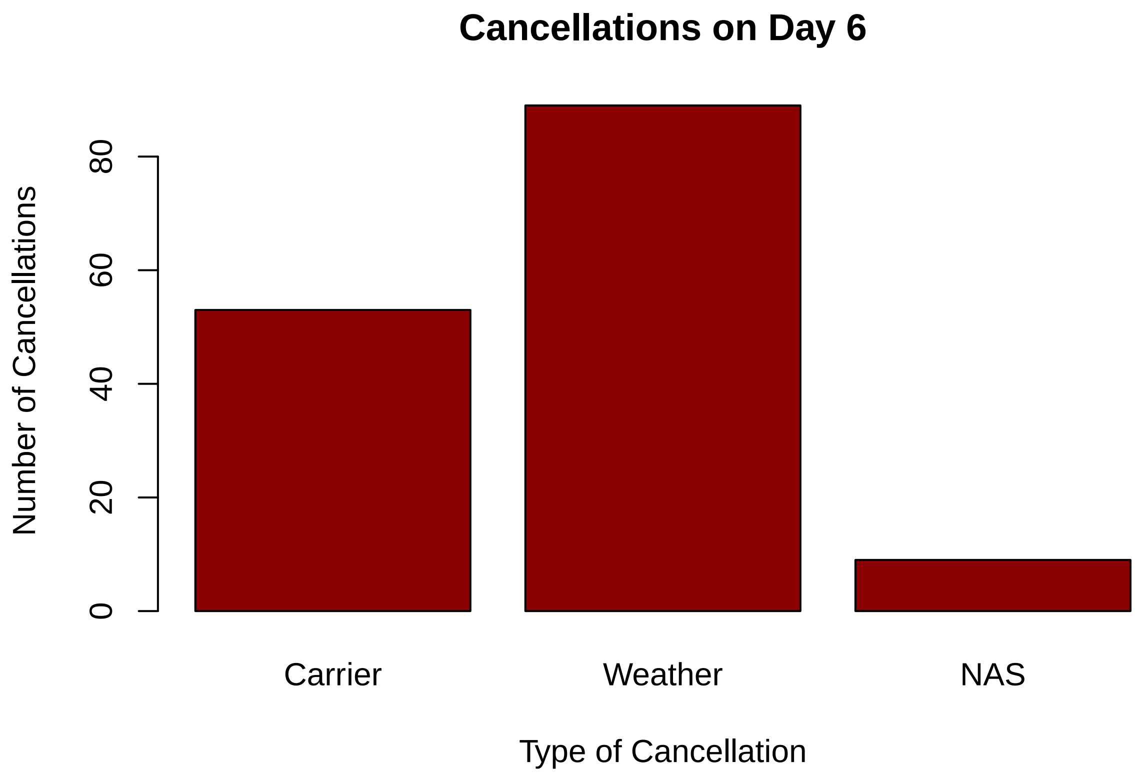

Cancellations on Day 1
Number of Cancellations
0
20
40
60
80
Carrier
Weather
NAS
Type of Cancellation

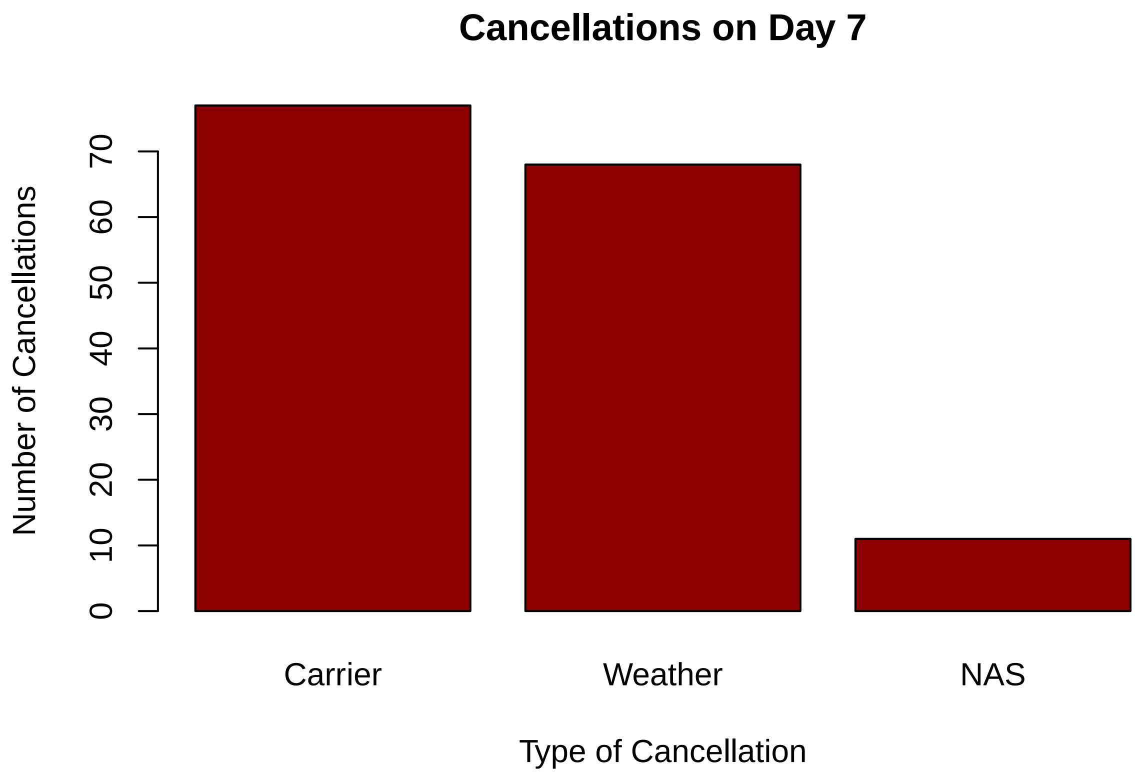

Cancellations on Day 2
Number of Cancellations
0
20
40
60
80
100
Carrier
Weather
NAS
Type of Cancellation


Cancellations on Day 3
Number of Cancellations
0
20
40
60
80
120
Carrier
Weather
NAS
Type of Cancellation


Cancellations on Day 4
Number of Cancellations
0
20
40
60
80
100
Carrier
Weather
NAS
Type of Cancellation


Cancellations on Day 5
Number of Cancellations
0
20
40
60
80
Carrier
Weather
NAS
Type of Cancellation


Cancellations on Day 6
Number of Cancellations
0
20
40
60
80
Carrier
Weather
NAS
Type of Cancellation



In these graphs, the days 1 through 7 correspond to Monday through Sunday in that order. Analyzing the results, we see that on most weekdays, cancellations are due to carrier issues. The only exception to this is on Tuesday where there are slightly more weather based cancellations. Saturdays seem to favor carrier cancellations while Sundays favor those of weather.

To conclude our results, it seems like Tuesdays are some of the worst days to travel from the Austin airport because of a mix of weather and carrier issues. If one wants to reduce the chances of a cancelled flight, choosing a clear day on the weekend seems to suggest the best travel conditions.

Exercise 3

We first start off with a simple portfolio of corporate bonds each invested in evenly from companies like iShares, Fidelity, and Vanguard. These seem like relatively safe bonds to invest into so we aim to see the risk behind this portfolio.

```
## Loading required package: lattice

## Loading required package: ggformula

## Loading required package: ggstance

##
## Attaching package: 'ggstance'

## The following objects are masked from 'package:ggplot2':
##
##     geom_errorbarh, GeomErrorbarh

##
## New to ggformula?  Try the tutorials:
```

```
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## 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.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.
##
## Have you tried the ggformula package for your plots?

##
## Attaching package: 'mosaic'

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

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

## The following object is masked from 'package:purrr':
##
##     cross

## 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

## Loading required package: xts

## Loading required package: zoo

##
## 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

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

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

##
## Attaching package: 'foreach'

## The following objects are masked from 'package:purrr':
##
##     accumulate, when

## 'getSymbols' currently uses auto.assign=TRUE by default, but will
## use auto.assign=FALSE in 0.5-0. You will still be able to use
## 'loadSymbols' to automatically load data. getOption("getSymbols.env")
## and getOption("getSymbols.auto.assign") will still be checked for
## alternate defaults.
##
## This message is shown once per session and may be disabled by setting
## options("getSymbols.warning4.0"=FALSE). See ?getSymbols for details.

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/LQD?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/LQD?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/FCOR?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/FCOR?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VCIT?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
```

```
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VCIT?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
##                ClCl.LQDa ClCl.FCORa ClCl.VCITa
## 2007-01-03            NA         NA         NA
## 2007-01-04  0.0075152938         NA         NA
## 2007-01-05 -0.0006526807         NA         NA
## 2007-01-08 -0.0002798843         NA         NA
## 2007-01-09  0.0001866169         NA         NA
## 2007-01-10 -0.0013063264         NA         NA
```

```
##        5%
## -2549.087
```

Our first portfolio was the smallest, and its value at risk was a loss of $2660.78. So we ultimately lost money by investing in these funds. However, there was not much volatility in this portfolio, so we compare this to our second one which is larger, and much more diverse. We aim to see if we can earn a profit from this kind of investment.

This second portfolio includes 7 funds, and are from varied sources like Japan Equities, All Cap Equities, Corporate Bonds, and more. These are also equally invested into.

```
## 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
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/LQD?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/LQD?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/DXJ?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/DXJ?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
```

```
## on 'https://query2.finance.yahoo.com/v7/finance/download/SDY?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SDY?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/XLK?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/XLK?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

##                ClCl.LQDa    ClCl.WMTa    ClCl.JNJa     ClCl.SPYa    ClCl.DXJa
## 2007-01-03            NA           NA           NA            NA           NA
## 2007-01-04  0.0075152938  0.004837014  0.012500015  0.0021221123  0.012120152
## 2007-01-05 -0.0006526807 -0.008162411 -0.009073330 -0.0079763183 -0.020911564
## 2007-01-08 -0.0002798843 -0.008229563 -0.001651171  0.0046250821  0.001460424
## 2007-01-09  0.0001866169  0.008297851 -0.003758833 -0.0008498831  0.007108986
## 2007-01-10 -0.0013063264 -0.002321165 -0.001660127  0.0033315799 -0.017918588
##                ClCl.SDYa    ClCl.XLKa
## 2007-01-03            NA           NA
## 2007-01-04  0.0014502256  0.015430819
## 2007-01-05 -0.0109412869 -0.008020304
## 2007-01-08 -0.0001627135  0.002978723
## 2007-01-09  0.0017897983  0.001272804
## 2007-01-10  0.0012993666  0.005084703

##       5%
## -4612.09
```

Interestingly, this portfolio also produced a value at risk in the negatives, that being $4835.53. While in general, it's good to diversify, in this case we predicted losses again. This could have been because the funds chosen simply did poorly since most of the choices were selected somewhat randomly.

In order to try to turn a profit, we use our third portfolio to aggressively hone in on funds that do well, instead of dividing our investment equally. We'll invest in the 4 technology equity ETFs that hold the most assets.

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/VGT?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VGT?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/XLK?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/XLK?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/IYW?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/IYW?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/IGV?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/IGV?
## period1=-2208988800&period2=1597622400&interval=1d&events=split&crumb=kyf3WpoMlDG'

##               ClCl.VGTa    ClCl.XLKa    ClCl.IYWa    ClCl.IGVa
## 2007-01-03           NA           NA           NA           NA
## 2007-01-04  0.018240528  0.015430819  0.016462448  0.018377410
## 2007-01-05 -0.007650681 -0.008020304 -0.008097913 -0.009903103
## 2007-01-08  0.001316284  0.002978723  0.003628465  0.004667681
## 2007-01-09  0.005446028  0.001272804  0.005965329  0.000000000
## 2007-01-10  0.006350392  0.005084703  0.008984726  0.003539823

##        5%
## -6431.504
```

The funds "VGT", "XLK", "IYW", and "IGV" were invested into with 50%, 25%, 12.5%, and 12.5% of our 100,000 in that order. Sadly, we find that this portfolio did the worst out of the three with a value at risk of $6858.35 (loss).

Ultimately, our portfolios all produced a loss. However, there are some conclusions to be drawn. It appears that corporate bonds led to the least loss of investment. This is an interesting result because it was one of our smaller portfolios, and it generally seems that diversifying leads to betters results. We can see this in our second portfolio that still operated at a loss but not as much as our third one. The third one suggests that heavily investing in funds of larger companies (in terms of assets) can still do poorly.

Exercise 4

Let's load in the data first.

```
## Warning: Missing column names filled in: 'X1' [1]

## Parsed with column specification:
## cols(
##   .default = col_double(),
##   X1 = col_character()
## )

## See spec(...) for full column specifications.
```

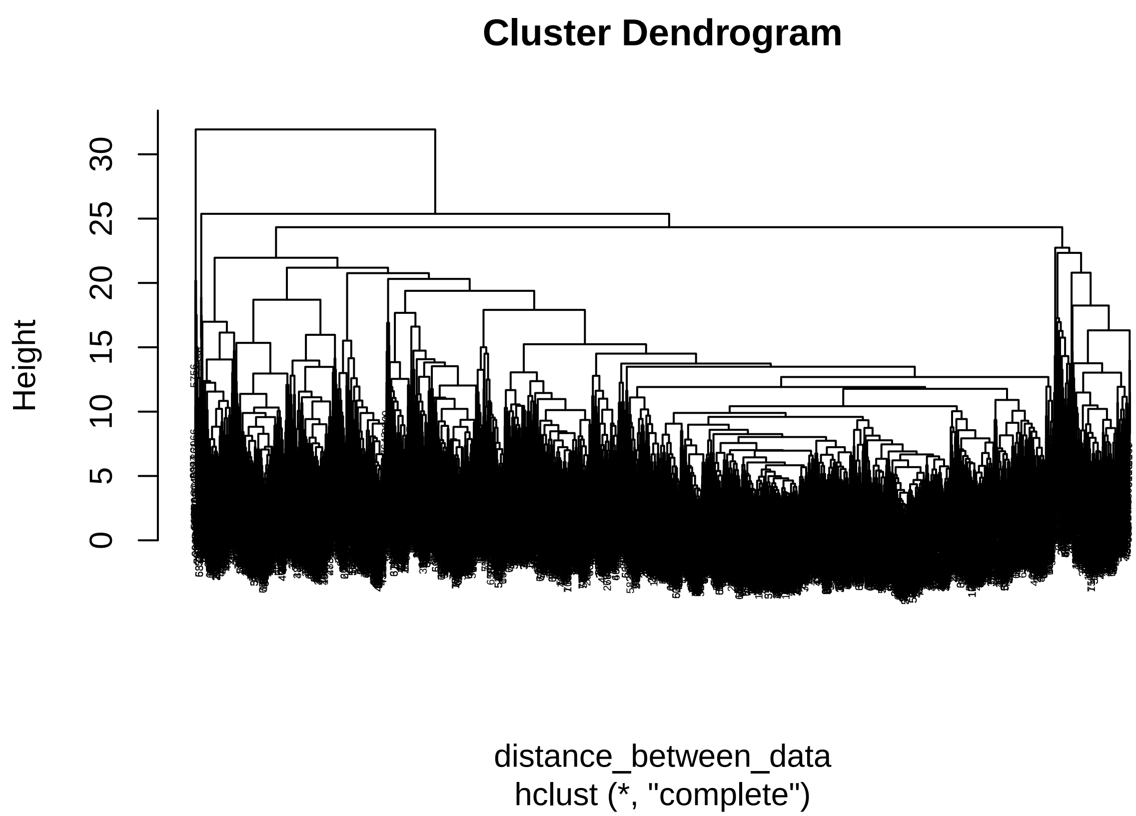
Now we run k-means++ to identify clusters within our data

```
##        X1               chatter       current_events      travel      
##  Length:7882        Min.   : 0.000   Min.   :0.000   Min.   : 0.000  
##  Class :character   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 0.000  
##  Mode  :character   Median : 3.000   Median :1.000   Median : 1.000  
##                     Mean   : 4.399   Mean   :1.526   Mean   : 1.585  
##                     3rd Qu.: 6.000   3rd Qu.:2.000   3rd Qu.: 2.000  
##                     Max.   :26.000   Max.   :8.000   Max.   :26.000  
##  photo_sharing    uncategorized      tv_film      sports_fandom   
##  Min.   : 0.000   Min.   :0.000   Min.   : 0.00   Min.   : 0.000  
##  1st Qu.: 1.000   1st Qu.:0.000   1st Qu.: 0.00   1st Qu.: 0.000  
##  Median : 2.000   Median :1.000   Median : 1.00   Median : 1.000  
##  Mean   : 2.697   Mean   :0.813   Mean   : 1.07   Mean   : 1.594  
##  3rd Qu.: 4.000   3rd Qu.:1.000   3rd Qu.: 1.00   3rd Qu.: 2.000  
##  Max.   :21.000   Max.   :9.000   Max.   :17.00   Max.   :20.000  
##     politics           food            family        home_and_garden 
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000  
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000  
##  Median : 1.000   Median : 1.000   Median : 1.0000   Median :0.0000  
##  Mean   : 1.789   Mean   : 1.397   Mean   : 0.8639   Mean   :0.5207  
##  3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000  
##  Max.   :37.000   Max.   :16.000   Max.   :10.0000   Max.   :5.0000  
##      music              news        online_gaming       shopping     
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000  
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000  
##  Median : 0.0000   Median : 0.000   Median : 0.000   Median : 1.000  
##  Mean   : 0.6793   Mean   : 1.206   Mean   : 1.209   Mean   : 1.389  
##  3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.000   3rd Qu.: 2.000  
##  Max.   :13.0000   Max.   :20.000   Max.   :27.000   Max.   :12.000  
##  health_nutrition  college_uni     sports_playing      cooking      
##  Min.   : 0.000   Min.   : 0.000   Min.   :0.0000   Min.   : 0.000  
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.000  
##  Median : 1.000   Median : 1.000   Median :0.0000   Median : 1.000  
##  Mean   : 2.567   Mean   : 1.549   Mean   :0.6392   Mean   : 1.998  
##  3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.: 2.000  
##  Max.   :41.000   Max.   :30.000   Max.   :8.0000   Max.   :33.000  
##       eco           computers          business         outdoors      
##  Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000  
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000  
##  Median :0.0000   Median : 0.0000   Median :0.0000   Median : 0.0000  
##  Mean   :0.5123   Mean   : 0.6491   Mean   :0.4232   Mean   : 0.7827  
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000  
##  Max.   :6.0000   Max.   :16.0000   Max.   :6.0000   Max.   :12.0000  
##      crafts         automotive           art              religion     
##  Min.   :0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000  
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000  
##  Median :0.0000   Median : 0.0000   Median : 0.0000   Median : 0.000  
##  Mean   :0.5159   Mean   : 0.8299   Mean   : 0.7248   Mean   : 1.095  
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.000  
##  Max.   :7.0000   Max.   :13.0000   Max.   :18.0000   Max.   :20.000  
##      beauty          parenting           dating             school      
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000  
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000  
```

```
##  Median : 0.0000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 0.7052   Mean   : 0.9213   Mean   : 0.7109   Mean   : 0.7677
##  3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000
##  Max.   :14.0000   Max.   :14.0000   Max.   :24.0000   Max.   :11.0000
##  personal_fitness    fashion        small_business        spam
##  Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Min.   :0.00000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.00000
##  Median : 0.000   Median : 0.0000   Median :0.0000   Median :0.00000
##  Mean   : 1.462   Mean   : 0.9966   Mean   :0.3363   Mean   :0.00647
##  3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000
##  Max.   :19.000   Max.   :18.0000   Max.   :6.0000   Max.   :2.00000
##      adult
##  Min.   : 0.0000
##  1st Qu.: 0.0000
##  Median : 0.0000
##  Mean   : 0.4033
##  3rd Qu.: 0.0000
##  Max.   :26.0000

##          chatter   current_events           travel    photo_sharing
##      4.482517483      1.487179487      1.573426573      2.818181818
##    uncategorized          tv_film    sports_fandom         politics
##      0.913752914      1.699300699      1.335664336      1.307692308
##             food           family  home_and_garden            music
##      1.247086247      1.079254079      0.613053613      0.955710956
##             news    online_gaming         shopping health_nutrition
##      0.797202797      9.694638695      1.365967366      1.783216783
##      college_uni   sports_playing          cooking              eco
##     10.564102564      2.613053613      1.482517483      0.489510490
##        computers         business         outdoors           crafts
##      0.585081585      0.417249417      0.659673660      0.603729604
##       automotive              art         religion           beauty
##      0.909090909      1.233100233      0.811188811      0.442890443
##        parenting           dating           school personal_fitness
##      0.675990676      0.748251748      0.512820513      1.025641026
##          fashion   small_business             spam            adult
##      0.899766900      0.461538462      0.009324009      0.445221445

##          chatter   current_events           travel    photo_sharing
##      4.328492849      1.444664466      1.099229923      2.296149615
##    uncategorized          tv_film    sports_fandom         politics
##      0.728272827      1.003080308      0.970517052      1.010341034
##             food           family  home_and_garden            music
##      0.769416942      0.573157316      0.440044004      0.562596260
##             news    online_gaming         shopping health_nutrition
##      0.692409241      0.588778878      1.278987899      1.091529153
##      college_uni   sports_playing          cooking              eco
##      0.908910891      0.421122112      0.862926293      0.389658966
##        computers         business         outdoors           crafts
##      0.373817382      0.339053905      0.401760176      0.363256326
##       automotive              art         religion           beauty
##      0.580858086      0.622002200      0.526732673      0.354015402
##        parenting           dating           school personal_fitness
##      0.458525853      0.543234323      0.477227723      0.659845985
##          fashion   small_business             spam            adult
```

```
##      0.514851485      0.277667767      0.006820682      0.416501650
##          chatter   current_events           travel    photo_sharing
##      4.548387097      1.667155425      5.612903226      2.541055718
##    uncategorized          tv_film    sports_fandom         politics
##      0.775659824      1.199413490      2.014662757      8.960410557
##             food           family  home_and_garden            music
##      1.441348974      0.913489736      0.611436950      0.640762463
##             news    online_gaming         shopping health_nutrition
##      5.318181818      0.828445748      1.379765396      1.639296188
##      college_uni   sports_playing          cooking              eco
##      1.318181818      0.629032258      1.259530792      0.593841642
##        computers         business         outdoors           crafts
##      2.473607038      0.670087977      0.916422287      0.640762463
##       automotive              art         religion           beauty
##      2.347507331      0.718475073      1.030791789      0.473607038
##        parenting           dating           school personal_fitness
##      0.947214076      1.068914956      0.725806452      1.000000000
##          fashion   small_business             spam            adult
##      0.668621701      0.483870968      0.005865103      0.236070381
```

```
##    1    2    3    4    5    6    7    8    9   10
##  487 5628  284  859  130   16  410   49    9   10
```



The K-means++ algorithm gives us some good insights of potential clusters for this data. In our first cluster, we see a high amount of interest in travel, photo sharing, politics, news, computers, and automotives. This market segment seems to be older people who are travellers that are very invested in current events and like to share their experiences online, potentially through social media.

The second cluster we came across has a high interest in online gaming, college_uni, sports_playing, and photo_sharing. This suggests these are young adults who have an interest in video games and competition in general due to the interest in sports as well. This could represent a younger group compared to our initial cluster.

The third cluster we investigated had a high interest in sports_fandom, food, family, religion, parenting, and school. This cluster suggests a group of parents who may potentially be looking into the futures of their children.

While these results are interesting, we'd like to try a similar method with fewer features to see if anything changes.

```r
library(ggplot2)
library(LICORS)  # for kmeans++
library(foreach)
library(mosaic)

#summary(social)

# Center and scale the data
X = social[,(3:35)]
X = scale(X, center=TRUE, scale=TRUE)

# Extract the centers and scales from the rescaled data (which are named attributes)
mu = attr(X,"scaled:center")
sigma = attr(X,"scaled:scale")

# Run k-means with 6 clusters and 25 starts
clust1 = kmeans(X, 6, nstart=25)

# What are the clusters?
#clust1$center  # not super helpful
#clust1$center[1,]*sigma + mu
#clust1$center[2,]*sigma + mu
#clust1$center[4,]*sigma + mu

# A few plots with cluster membership shown
# qplot is in the ggplot2 library
#qplot(current_events, chatter, data=social, color=factor(clust1$cluster))
#qplot(Horsepower, CityMPG, data=social, color=factor(clust1$cluster))


# Using kmeans++ initialization
clust2 = kmeanspp(X, k=6, nstart=25)

clust2$center[1,]*sigma + mu
```

```
##   current_events          travel   photo_sharing   uncategorized
##        1.5563063       1.2387387       2.6632883       0.9650901
##          tv_film   sports_fandom        politics            food
##        0.9909910       1.1677928       1.2590090       2.1227477
##           family home_and_garden           music            news
##        0.7747748       0.6385135       0.7432432       1.1058559
##    online_gaming        shopping health_nutrition     college_uni
##        0.8468468       1.4740991      11.9977477       0.9335586
##   sports_playing         cooking             eco       computers
```

```
##       0.6036036        3.2691441        0.9268018        0.5574324
##        business         outdoors           crafts       automotive
##       0.4763514        2.7308559        0.5889640        0.6655405
##             art         religion           beauty        parenting
##       0.7398649        0.7646396        0.4222973        0.7612613
##          dating           school personal_fitness          fashion
##       1.0337838        0.5968468        6.4335586        0.7894144
##  small_business
##       0.2939189
```

```
clust2$center[2,]*sigma + mu
```

```
##  current_events           travel    photo_sharing    uncategorized
##       1.6809896        1.3463542        2.6380208        0.7591146
##         tv_film    sports_fandom         politics             food
##       1.0885417        5.8697917        1.1614583        4.5520833
##          family  home_and_garden            music             news
##       2.4895833        0.6458333        0.7473958        1.0364583
##   online_gaming         shopping health_nutrition      college_uni
##       1.0078125        1.4804688        1.8541667        1.2356771
##  sports_playing          cooking              eco        computers
##       0.7447917        1.5976563        0.6601562        0.7317708
##        business         outdoors           crafts       automotive
##       0.5013021        0.6888021        1.0807292        1.0455729
##             art         religion           beauty        parenting
##       0.8723958        5.2382812        1.0937500        4.0442708
##          dating           school personal_fitness          fashion
##       0.8164062        2.7018229        1.1927083        1.0156250
##  small_business
##       0.4023438
```

```
clust2$center[4,]*sigma + mu
```

```
##  current_events           travel    photo_sharing    uncategorized
##       1.7750439        1.5026362        6.1282953        1.2934974
##         tv_film    sports_fandom         politics             food
##       1.0615114        1.1616872        1.4428822        1.0790861
##          family  home_and_garden            music             news
##       0.9068541        0.6326889        1.2724077        1.0597540
##   online_gaming         shopping health_nutrition      college_uni
##       1.0632689        2.0404218        2.3005272        1.5307557
##  sports_playing          cooking              eco        computers
##       0.8154657       10.8963093        0.5694200        0.7346221
##        business         outdoors           crafts       automotive
##       0.6080844        0.8365554        0.6344464        0.9015817
##             art         religion           beauty        parenting
##       0.9121265        0.8611599        3.8980668        0.8014060
##          dating           school personal_fitness          fashion
##       0.9666081        0.9912127        1.3620387        5.6010545
##  small_business
##       0.4991213
```

From our first cluster, we see photo_sharing is a big component again with similar results as before. However, shopping becomes a new topic that comes up. This does align with our previous segment that we predicted since they seem to have a lot of online activity and online shopping can contribute to that.

The second cluster we look at again is involved with photo_sharing but this time, it's grouped with beauty, fashion, and cooking. This seems like it could be correlated with the previous group we just talked about.

From the third cluster we look at, we get a similar result to the gaming group mentioned in the first attempt we made. From these results, it seems like NutrientH20's primary demographic is adult parents who have an interest in travelling, health, and their family. There seems to be a potentially younger group of college students who also has an interest in them.

Exercise 5

We first find a way to read in all the training data.

```
library(tm)
```

```
## 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
```

```
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)), id=fname, language='en')
}
#get all authors for the documents
authors <- rep("", 50)
i <- 1
for (f in Sys.glob('./ReutersC50/C50train/*')){
  authors[i] <- tail(strsplit(f, "/")[[1]], 1)
  i <- i+1
}
#since each author has 50 documents, we need to replicate each author 50 times for the dataframe
authors <- rep(authors, each=50)
#instantiate dataframe with first column being authors
train_df <- data.frame(author=authors, txt=rep("", 2500), stringsAsFactors = FALSE)
#add in text for each document after concatening all lines with a space
file_list_train = Sys.glob('./ReutersC50/C50train/*/*.txt')
trainfiles = lapply(file_list_train, readerPlain)
for(i in 1:length(trainfiles)){
  contentvec <- trainfiles[[i]]$content
  train_df[i,2] <- paste(contentvec, collapse = " ")
}
```

Now we aim to find the term frequency used by each author. This will become the metric used to predict later on.

```
library(dplyr)
library(tidytext)
```

```r
train_text <- train_df %>%
  unnest_tokens(word, txt) %>%
  count(author, word, sort = TRUE)
```

```
## Warning: `count_()` is deprecated as of dplyr 0.7.0.
## Please use `count()` instead.
## See vignette('programming') for more help
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_warnings()` to see where this warning was generated.
```

```r
#grab all unique words so we can make them columns in a new dataframe
unique_words = unique(train_text["word"])

# new dataframe to hold word counts
word_freq <- data.frame(authorName=unique(authors), stringsAsFactors = FALSE)

# populate columns with words
for(word in unique_words) {
  word_freq[,word] <- 0
}

# add back in author names
for (i in 1:50)(
  word_freq[i,"author"] <- unique(authors)[i]
)

# now we populate the cell values with counts from train_text

for (i in 1:172933){

  try(word_freq[match(train_text["author"][i,],word_freq$authorName),train_text["word"][i,]] <- train_t
}
```

We now have a dataframe with authors and the number of times certain words appear in their works. We now apply a decision tree model to predict authors from this data.

```r
library(tree)
```

```
## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli
```

```r
train <- word_freq[,1:10000]
csTree <- tree(authorName~., data = data.frame(word_freq[,1:10000]))
```

```
## Warning in tree(authorName ~ ., data = data.frame(word_freq[, 1:10000])): NAs
## introduced by coercion
```

Now we can create our test set and see how our model performs.

```r
library(tm)
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)), id=fname, language='en')
}
#get all authors for the documents
authors <- rep("", 50)
i <- 1
```

```
for (f in Sys.glob('./ReutersC50/C50test/*')){
  authors[i] <- tail(strsplit(f, "/")[[1]], 1)
  i <- i+1
}
#since each author has 50 documents, we need to replicate each author 50 times for the dataframe
authors <- rep(authors, each=50)
#instantiate dataframe with first column being authors
test_df <- data.frame(author=authors, txt=rep("", 2500), stringsAsFactors = FALSE)
#add in text for each document after concatening all lines with a space
file_list_train = Sys.glob('./ReutersC50/C50test/*/*.txt')
testfiles = lapply(file_list_train, readerPlain)
for(i in 1:length(trainfiles)){
  contentvec <- trainfiles[[i]]$content
  test_df[i,2] <- paste(contentvec, collapse = " ")
}
```

Now we create another frequency table like before.

```
library(dplyr)
library(tidytext)

test_text <- test_df %>%
  unnest_tokens(word, txt) %>%
  count(author, word, sort = TRUE)

#grab all unique words so we can make them columns in a new dataframe
unique_words = unique(test_text["word"])

# new dataframe to hold word counts
word_freq2 <- data.frame(authorName=unique(authors), stringsAsFactors = FALSE)

# populate columns with words
for(word in unique_words) {
  word_freq2[,word] <- 0
}

# add back in author names
for (i in 1:50)(
  word_freq2[i,"author"] <- unique(authors)[i]
)

# now we populate the cell values with counts from test_txt

for (i in 1:177674){

  try(word_freq2[match(test_text["author"][i,],word_freq2$authorName),test_text["word"][i,]] <- test_te
}
```

With this data, we can test it on the model we made earlier.

```
treePred <- try(predict(csTree, newdata = data.frame(word_freq2[,1:100])), silent = TRUE)
```

Exercise 6

We first read in the data. This time, we have it on our local system.

```
library(tidyverse)
groceries <- readLines("groceries.txt")       # read all lines
groceries <- strsplit(groceries, ",", fixed=TRUE)     # split each line by commas, returns a list

library(arules)
```

```
##
## 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(arulesViz)
```

```
## Loading required package: grid
## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
groceries <- as(groceries, Class =  "transactions")            # turn into to transaction object
grocery_rules <- apriori(groceries, parameter=list(support=.005, confidence=.1, maxlen=5))   # etc
```

```
## 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 TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
```

```
## writing ... [1582 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
#inspect(grocery_rules)

# plot all the rules in (support, confidence) space
plot(grocery_rules)
```

```
## To reduce overplotting, jitter is added! Use jitter = 0 to prevent jitter.
```


```r
inspect(subset(grocery_rules, subset=lift > 3.8))
```

```
##     lhs                     rhs                     support confidence    coverage     lift count
## [1] {herbs}              => {root vegetables}    0.007015760  0.4312500 0.01626843 3.956477    69
## [2] {ham}                => {white bread}        0.005083884  0.1953125 0.02602949 4.639851    50
## [3] {white bread}        => {ham}                0.005083884  0.1207729 0.04209456 4.639851    50
## [4] {other vegetables,
##      root vegetables}    => {onions}             0.005693950  0.1201717 0.04738180 3.875044    56
## [5] {butter,
##      other vegetables} => {whipped/sour cream} 0.005795628  0.2893401 0.02003050 4.036397    57
## [6] {citrus fruit,
##      pip fruit}          => {tropical fruit}     0.005592272  0.4044118 0.01382816 3.854060    55
## [7] {citrus fruit,
##      other vegetables,
##      whole milk}         => {root vegetables}    0.005795628  0.4453125 0.01301474 4.085493    57
```

From the plot, we let our lift be greater than 3.8 to get a few rules that seem to be strong. Looking at the data above, we see that herbs usually lead to a purchase of other root vegetables. We also see that ham and white bread are typically purchased together. Another significant association rule is that the purchase of butter and other vegetables usually lead to whipped/sour cream as well.

```r
inspect(subset(grocery_rules, subset=confidence > 0.63))
```

```
##     lhs                      rhs                     support confidence    coverage     lift count
## [1] {curd,
##      tropical fruit}     => {whole milk}       0.006507372  0.6336634 0.010269446 2.479936    64
## [2] {butter,
##      whipped/sour cream} => {whole milk}       0.006710727  0.6600000 0.010167768 2.583008    66
## [3] {butter,
##      root vegetables}    => {whole milk}       0.008235892  0.6377953 0.012913066 2.496107    81
## [4] {butter,
##      yogurt}             => {whole milk}       0.009354347  0.6388889 0.014641586 2.500387    92
## [5] {pip fruit,
##      whipped/sour cream} => {whole milk}       0.005998983  0.6483516 0.009252669 2.537421    59
## [6] {other vegetables,
##      pip fruit,
##      root vegetables}    => {whole milk}       0.005490595  0.6750000 0.008134215 2.641713    54
## [7] {citrus fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005795628  0.6333333 0.009150991 3.273165    57
## [8] {root vegetables,
##      tropical fruit,
##      yogurt}             => {whole milk}       0.005693950  0.7000000 0.008134215 2.739554    56
```

Here, we check the rules where the confidence is above 0.63. Here we see many rules that lead to the purchase of whole milk. This usually stems from the purchase of some kind of dairy product like butter, cream, or yogurt. Another interesting rule is that the citrus, root vegetables, and, whole milk can lead to the purchase of other vegetables.

```r
inspect(subset(grocery_rules, subset=lift > 3 & confidence > 0.5))
```

```
##     lhs                      rhs                     support confidence    coverage     lift count
## [1] {onions,
##      root vegetables}    => {other vegetables} 0.005693950  0.6021505 0.009456024 3.112008    56
## [2] {curd,
##      tropical fruit}     => {yogurt}           0.005287239  0.5148515 0.010269446 3.690645    52
## [3] {pip fruit,
##      whipped/sour cream} => {other vegetables} 0.005592272  0.6043956 0.009252669 3.123610    55
## [4] {citrus fruit,
##      root vegetables}    => {other vegetables} 0.010371124  0.5862069 0.017691917 3.029608   102
## [5] {root vegetables,
##      tropical fruit}     => {other vegetables} 0.012302999  0.5845411 0.021047280 3.020999   121
## [6] {pip fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005490595  0.6136364 0.008947636 3.171368    54
## [7] {citrus fruit,
##      root vegetables,
##      whole milk}         => {other vegetables} 0.005795628  0.6333333 0.009150991 3.273165    57
## [8] {root vegetables,
##      tropical fruit,
##      whole milk}         => {other vegetables} 0.007015760  0.5847458 0.011997966 3.022057    69
```

In this last subset, we chose a combination of lift greater than 3 and confidence greater than 0.5 to see a few of the higher end rules. Most of these rules seem to tie root vegetables to the purchase of other vegetables, but another interesting rule is that curd and tropical fruit lead to the purchase of yogurt as well.