Predictive Modeling Exercises

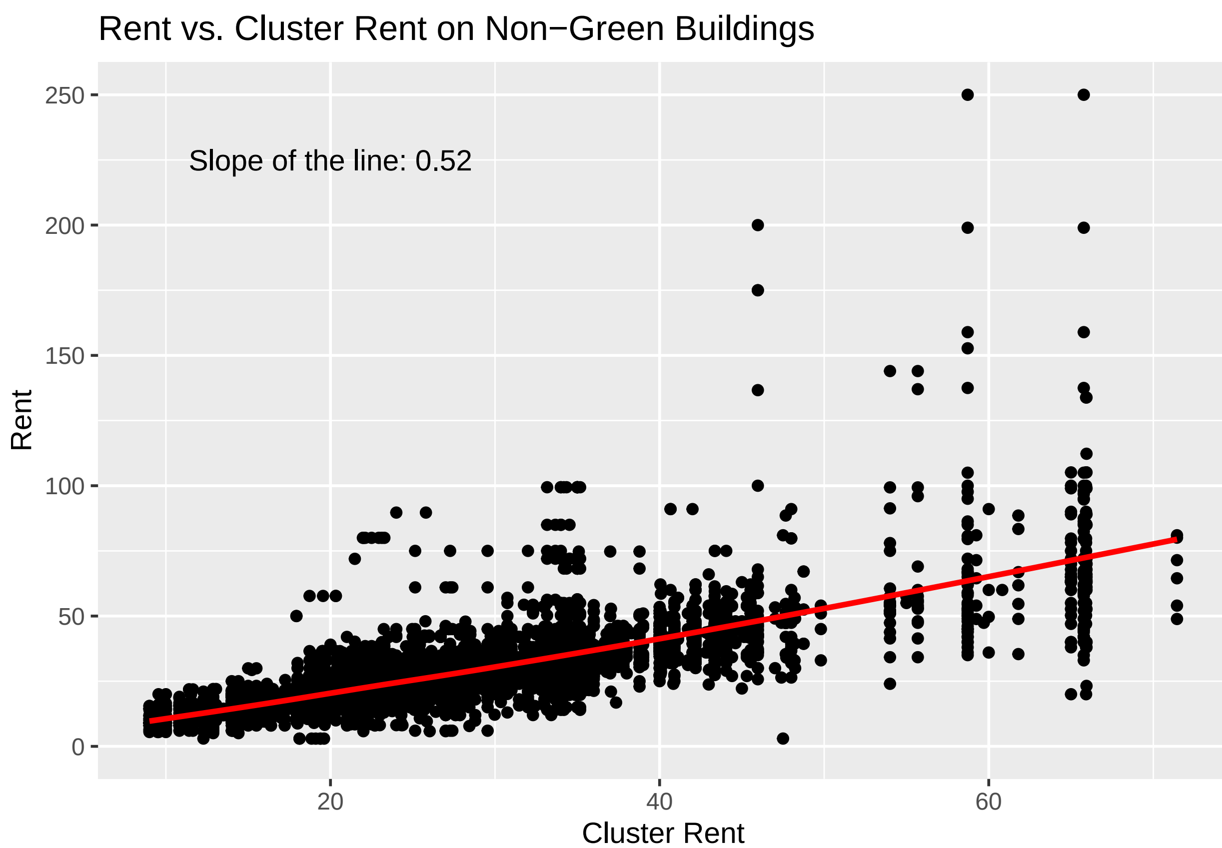
Ram Kapistalam

8/16/2020

Question 1

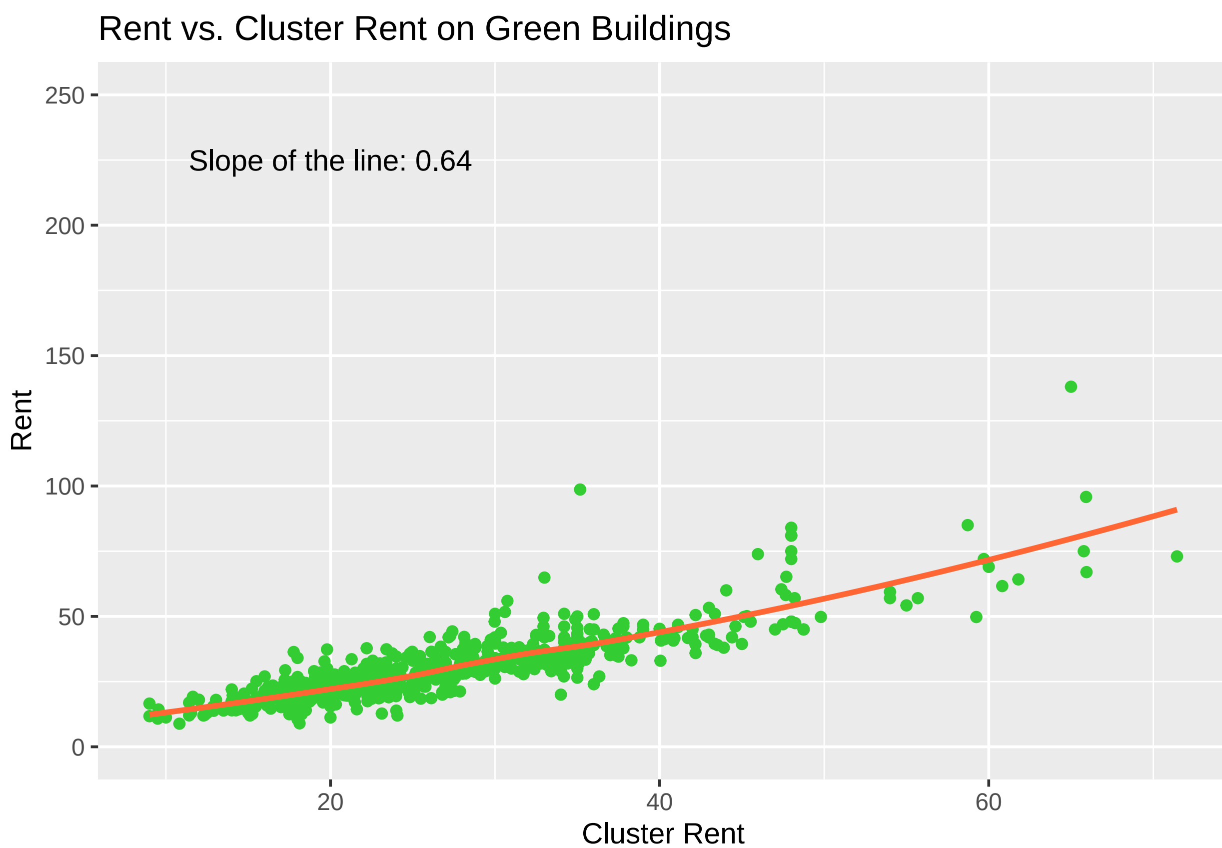
```
non_eco_slope <- summary(lm(cluster_rent ~ Rent, data = non_eco))
non_eco_slope <- non_eco_slope$coefficients[2,1]
geom_non_eco <- ggplot(data = non_eco, aes(x=cluster_rent, y=Rent)) +
  geom_point(color = 'black') +
  geom_smooth(se=FALSE, color = 'red') +
  labs(
    x = 'Cluster Rent',
    y = 'Rent',
    title = 'Rent vs. Cluster Rent on Non-Green Buildings') +
  ylim(0,250)
geom_non_eco + annotate(geom = 'text', x = 20, y=225, label =
                          paste("Slope of the line:", round(non_eco_slope, 2), sep = ' '))
```

```
## `geom_smooth()` using method = 'gam' and formula 'y ~ s(x, bs = "cs")'
```




```r
eco_slope <- summary(lm(cluster_rent ~ Rent, data = eco))
eco_slope <- eco_slope$coefficients[2,1]
geom_eco <- ggplot(data = eco, aes(x=cluster_rent, y=Rent)) +
  geom_point(color = '#33CC33') +
  geom_smooth(se=FALSE, color = '#FF6633') +
  labs(
    x = 'Cluster Rent',
    y = 'Rent',
    title = 'Rent vs. Cluster Rent on Green Buildings') +
  ylim(0,250)
geom_eco + annotate(geom = 'text', x = 20, y=225, label =
                         paste("Slope of the line:", round(eco_slope, 2), sep = ' '))
```

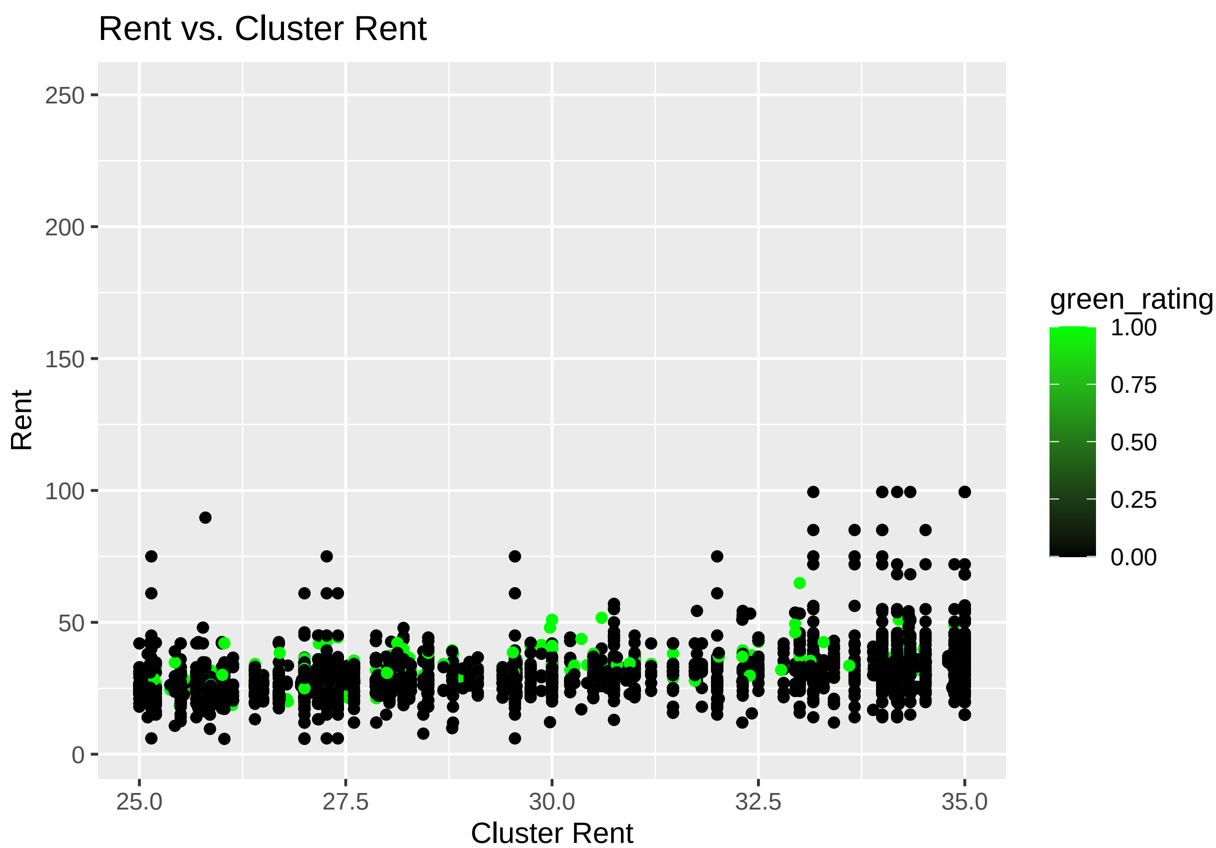
```
## `geom_smooth()` using method = 'loess' and formula 'y ~ x'
```



In the two plots above, we can see two important details. One being the average rent per sqaure foot of a green building is $1.59 more. The second aspect is that the the slope of the line comparing with green buildings is slighly steeper than the line on the non-green buildings plot. This demonstrates that the developer can charge more from green buildings compared to the average rent of the surrounding area than without a green building.

```
subsetted_df <- subset(greenbuildings, leasing_rate > 10)
ggplot(data = subsetted_df, aes(x=cluster_rent, y=Rent, color = green_rating)) +
  geom_point() +
  labs(x = 'Cluster Rent',
    y = 'Rent',
    title = 'Rent vs. Cluster Rent') +
  xlim(25,35) +
  scale_colour_gradient(low = 'black', high = 'green')
```

```
## Warning: Removed 5051 rows containing missing values (geom_point).
```



The plot seen above is a representation of green and non-green buildings’ rent when compared to cluster rent. As seen above, the green buildings tend to be at the top of for each cluster rent price.

```
number_of_non_eco <- length(non_eco[,1])
number_of_eco <- length(eco[,1])
number_of_buildings <- length(subset(greenbuildings, leasing_rate > 10) [,1])
leasing_rate_eco <- length(subset(eco, leasing_rate >= 90)[,1])
leasing_rate_non_eco <- length(subset(non_eco, leasing_rate >= 90)[,1])
eco_pro <- (leasing_rate_eco/number_of_eco)*100
paste(’Proportion of Green Buildings with high leasing rates: ’, sep = ’’, round(eco_pro,2), ’%’)
```

```
## [1] "Proportion of Green Buildings with high leasing rates: 61.4%"
```

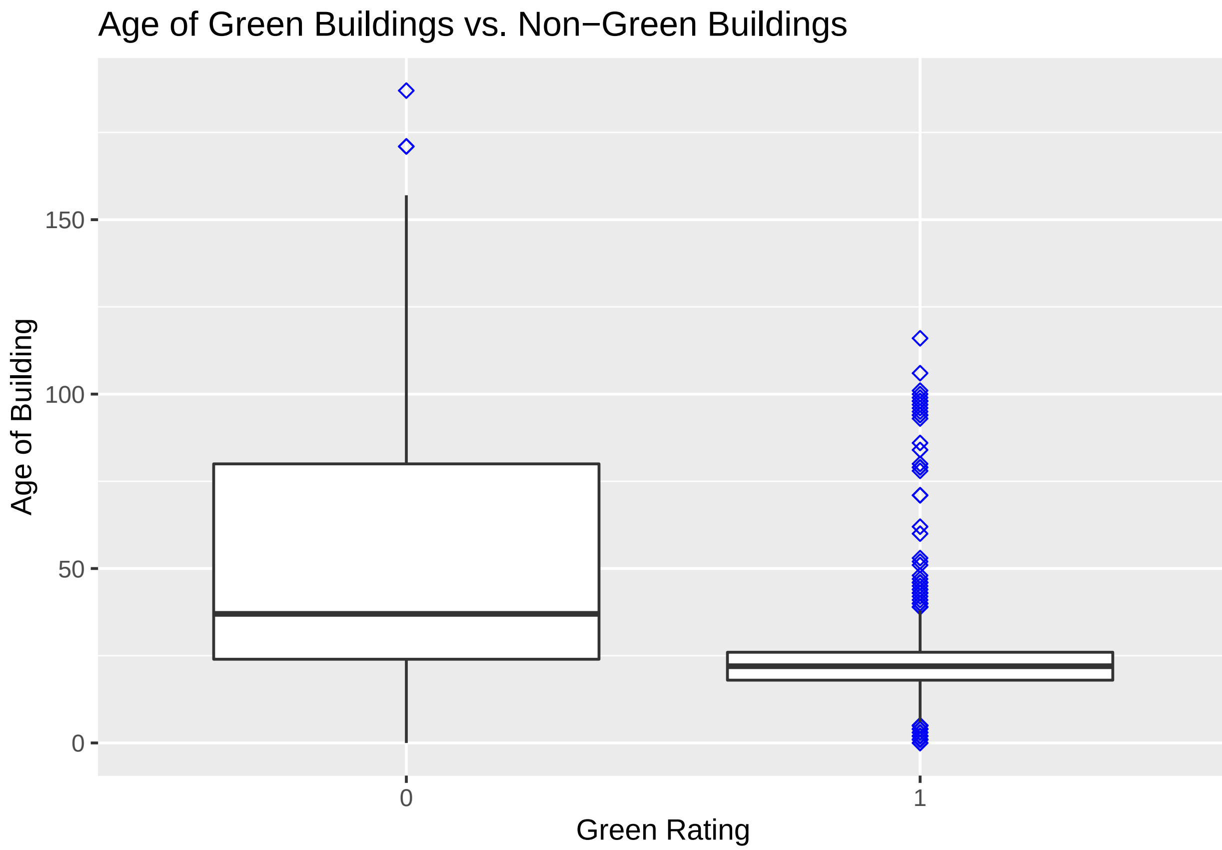
```
non_eco_pro <- (leasing_rate_non_eco/number_of_non_eco)*100
paste(’Proportion of Non-Green Buildings with high leasing rates: ’, sep = ’’, round(non_eco_pro, 2), ’
```

```
## [1] "Proportion of Non-Green Buildings with high leasing rates: 49.36%"
```

An issue that wasn’t discussed in the Stats Gurus analysis was the question about leasing rates. Fortunately, as seen by the numbers above, green buildings also have a larger proportion of higher leasing rates. This provides significant detail about how successful a green building is likely to lease out their space.

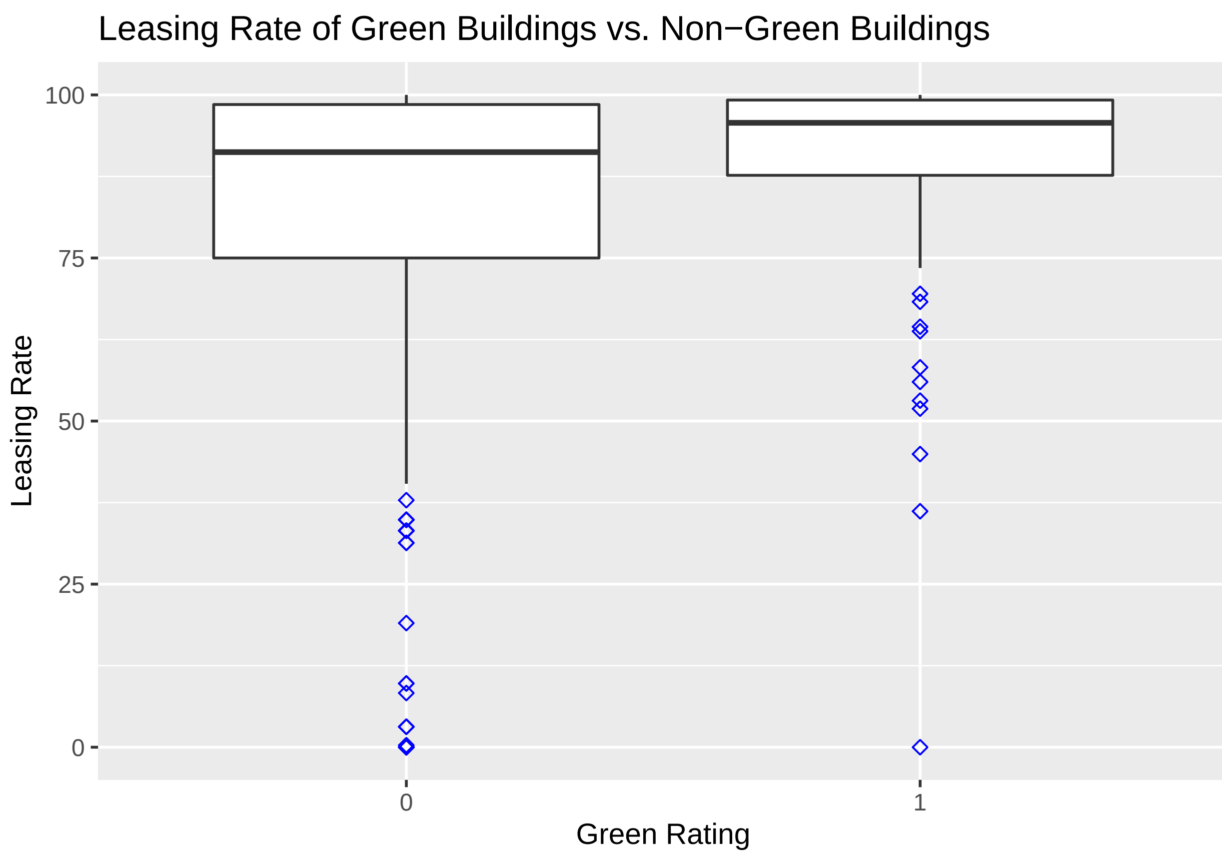
```
ggplot(greenbuildings, aes(x=factor(green_rating), y=age)) +
  geom_boxplot(outlier.colour="blue", outlier.shape=5) +
```

```r
  labs(x = 'Green Rating',
       y = 'Age of Building',
       title = 'Age of Green Buildings vs. Non-Green Buildings'
  )
```



Detailed above is a boxplot, which measures the age of green buildings compared to age of non-green buildings. Age might be a significant compounding variable when it comes to why average rent is higher, average cluster rent is higher, and leasing rates are higher. Putting the age in perspective demonstrates that green buildings tend to be much younger than non-green buildings thus leading to these other factors.

```r
age_15 <- subset(greenbuildings, age < 15)
ggplot(age_15, aes(x=factor(green_rating), y=leasing_rate)) +
  geom_boxplot(outlier.colour="blue", outlier.shape=5) +
  labs(x = 'Green Rating',
       y = 'Leasing Rate',
       title = 'Leasing Rate of Green Buildings vs. Non-Green Buildings')
```



Finally, after subsetting the data to buildings that are less than 15 years old, we find that green buildings continue to have a higher and more concise range of leasing rates. This is essential in understanding the success of a building in terms of a landlord. Yes, you can always charge more for rent, but even when matched up by age with non-green buildings, green buildings continue to perform beter with leasing rates.

Question 2

```
abia <- read.csv('ABIA.csv')
time1 = paste(abia$Year, sep = "-", abia$Month, abia$DayofMonth)
time1 = as.Date(time1)
abia$date = time1
```

```
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

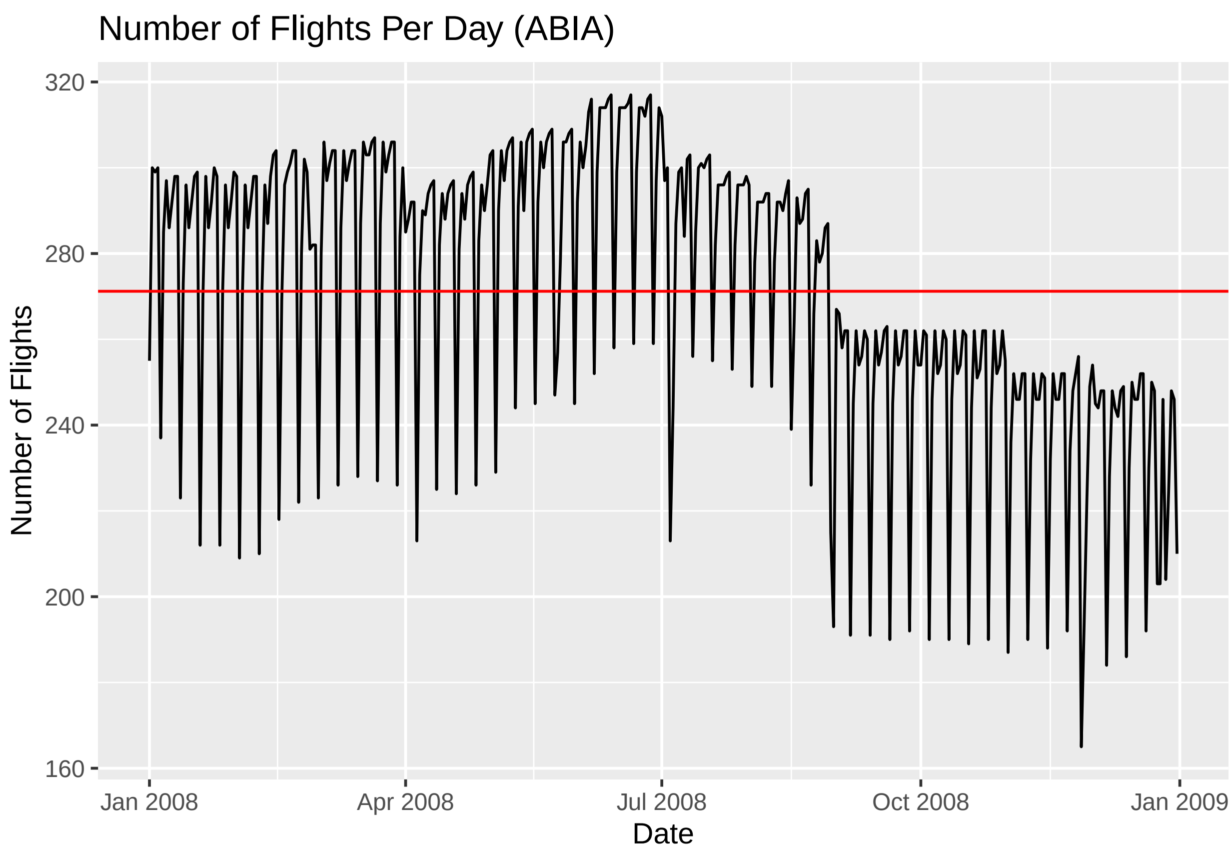
```r
library(plotly)
```

```
##
## Attaching package: 'plotly'

## 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
```

```r
df1 = abia %>%
  group_by(date) %>%
  summarize(count=n())
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```r
names(df1)[names(df1) == "date"] <- "Date"
names(df1)[names(df1) == "count"] <- "Count"
line_graph <- ggplot(df1, aes(x = Date, y = Count, group = 1)) +
  geom_line() +
  labs(
    x='Date',
    y='Number of Flights',
    title='Number of Flights Per Day (ABIA)'
  ) +
  geom_hline(yintercept = mean(df1$Count), color = 'red')
line_graph
```



This plot is a visual representation of the amount of flights in and out of ABIA per day over the course of 2008.

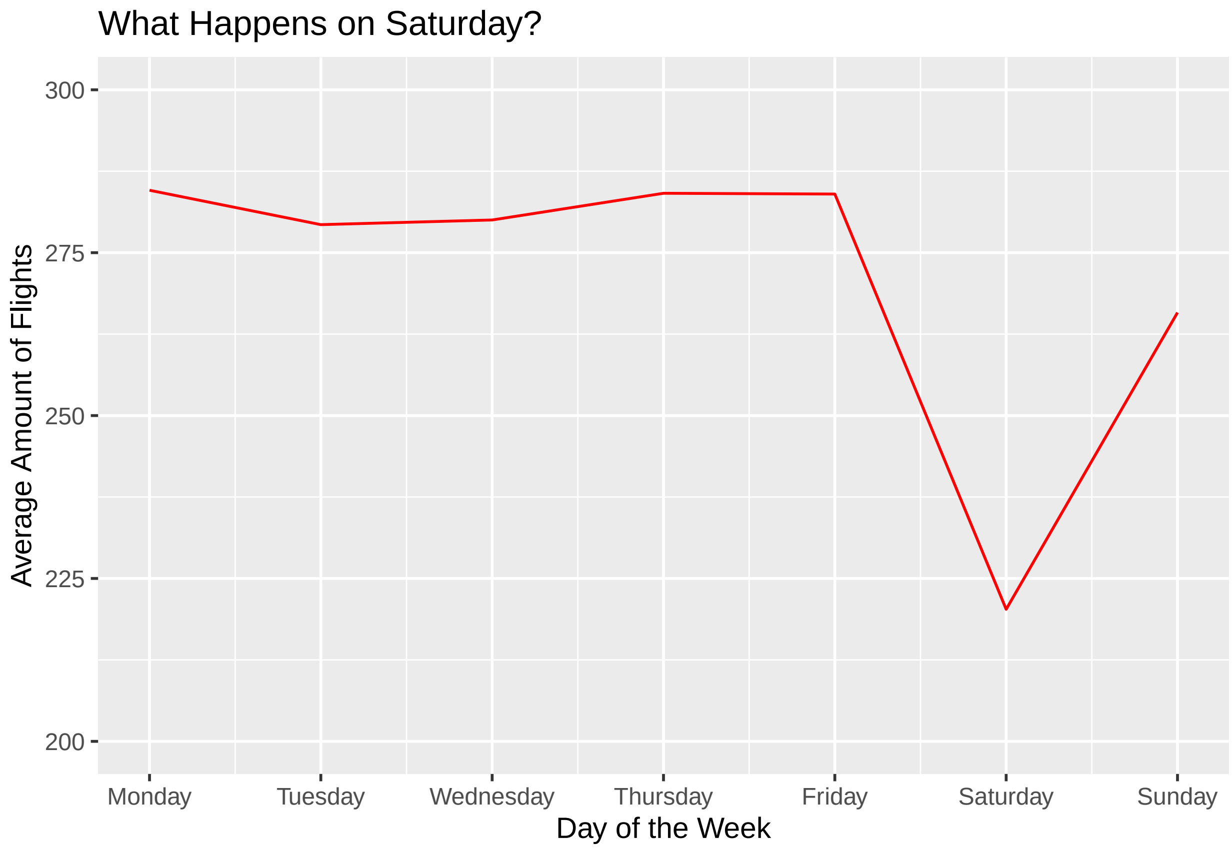
```
df2 = abia %>%
  group_by(date, DayOfWeek) %>%
  summarize(count=n())
```

```
## `summarise()` regrouping output by 'date' (override with `.groups` argument)
```

```
df3 = df2 %>%
  group_by(DayOfWeek) %>%
  summarize(avg = mean(count))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(df3, aes(x=DayOfWeek, y=avg)) +
  geom_line(color = 'red') +
  labs(
    x='Day of the Week',
    y='Average Amount of Flights',
    title='What Happens on Saturday?'
  ) +
  ylim(200,300) +
  scale_x_continuous(breaks=c(1,2,3,4,5,6,7), labels=c('Monday','Tuesday','Wednesday', 'Thursday', 'Fri
```

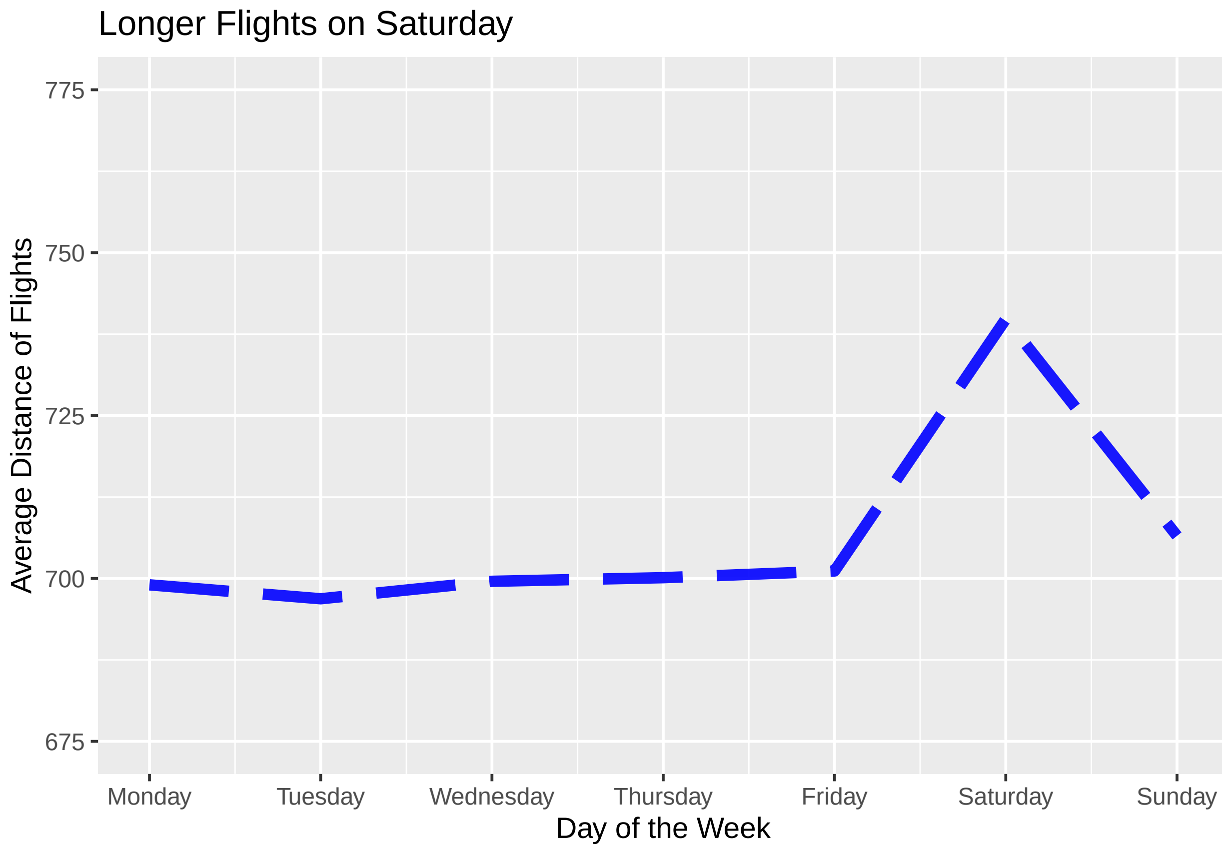


The plot above reveals a significant drop in flights on Saturdays when compared to all other days of the week.

```
df4 = abia %>%
  group_by(DayOfWeek) %>%
  summarize(avg = mean(Distance))
```

```
## `summarise()` ungrouping output (override with `.groups` argument)
```

```
ggplot(df4, aes(x=DayOfWeek, y=avg)) +
  geom_line(color = 'blue', size = 2, alpha = 0.9, linetype =  5) +
  ylim(675, 775) +
  labs(
    x='Day of the Week',
    y='Average Distance of Flights',
    title='Longer Flights on Saturday'
  ) +
  scale_x_continuous(breaks=c(1,2,3,4,5,6,7), labels=c('Monday','Tuesday','Wednesday', 'Thursday','Frida
```



The line graph above demonstrates that flights coming in and out of ABIA have a larger distance on average. Perhaps, ABIA has less flights on Saturday because many of those flights travel further.

Question 3

For the sake of this problem, the three different ETF possibilities we chose were a value, growth, and diversified portfolio. This is a good mix to analyze because typically, there is a tradeoff between value and growth ETF's based on the risk you are willing to take. Growth ETF's such Facebook and Amazon can yield higher returns but could also cause increased volatility. In contrast, value ETF's can provide a certain level of stability that may not be as available with growth. The final portfolio we chose was diversified with multiple different stocks. This was used as initial experimentation to assess whether diversified ETF's could yield better returns. To begin this report, we have illustrated histograms of returns for these ETF groups as well as the mean profit/loss and Value at Risk (VAR)

```
set.seed(12345)
library(mosaic)
```

```
## Loading required package: lattice
```

```
## Loading required package: ggformula
```

```
## Loading required package: ggstance
```

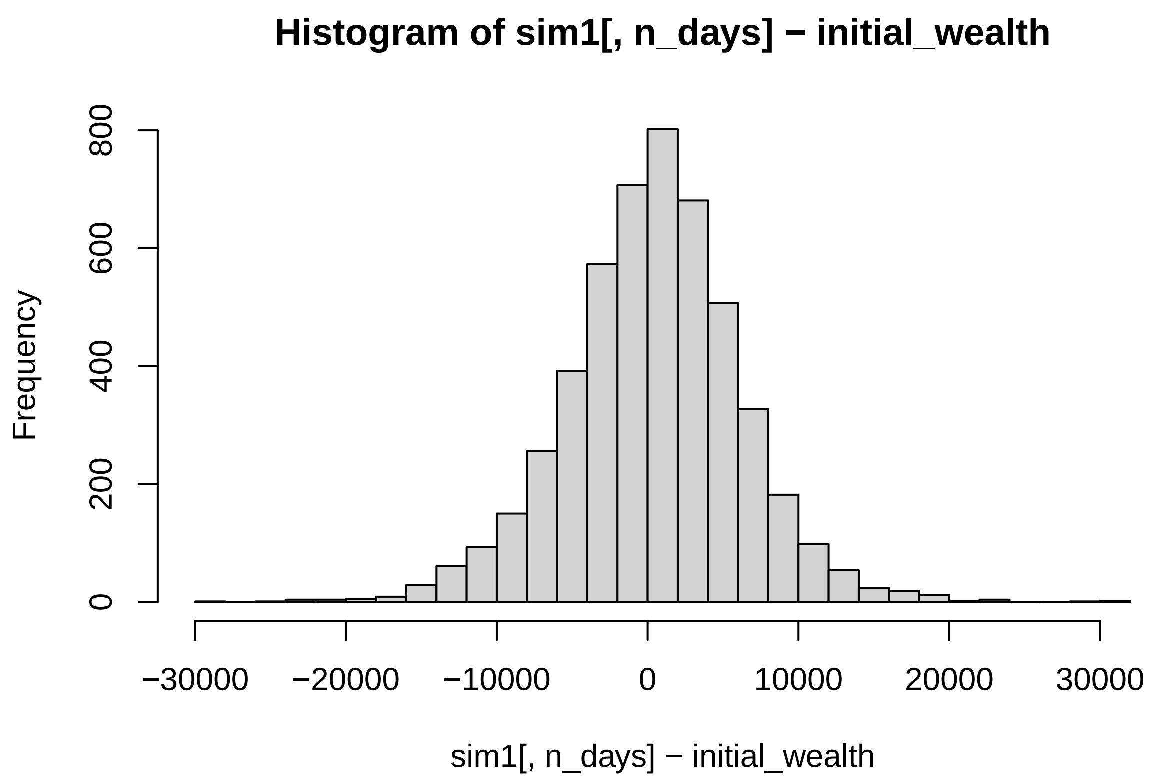
```
##
## Attaching package: 'ggstance'
```

```
## 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

## 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 object is masked from 'package:plotly':
##
##     do

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

## 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
```

```r
library(quantmod)
library(foreach)
all_returns_value = as.matrix(na.omit(all_returns_value))
total_wealth = 100000
weights = c(0.25, 0.25, 0.25, 0.25)
holdings = weights * total_wealth
n_days = 20
wealthtracker_value = rep(0, n_days)
for(today in 1:n_days) {
    return.today = resample(all_returns_value, 1, orig.ids=FALSE)
    holdings = holdings + holdings * return.today
    total_wealth = sum(holdings)
    wealthtracker_value[today] = total_wealth
    holdings = weights * total_wealth
}

initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.25,0.25,0.25,0.25)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker_value = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns_value, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker_value[today] = total_wealth
        holdings = weights * total_wealth
    }
    wealthtracker_value
}

print(mean(sim1[,n_days] - initial_wealth))
```

```
## [1] 390.9841
```

```r
hist(sim1[,n_days]- initial_wealth, breaks=30)
```




```
print(quantile(sim1[,n_days]- initial_wealth, prob=0.05))
```

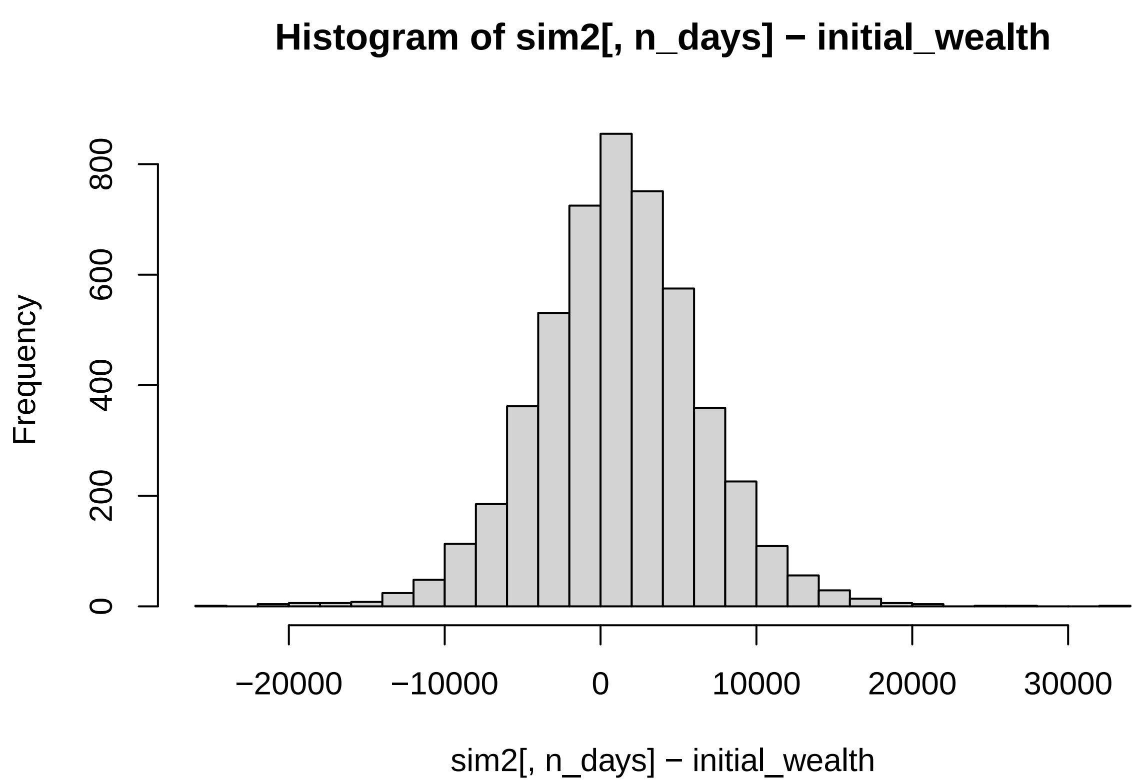
```
##        5%
## -9311.094
```

```
set.seed(12345)
all_returns_growth = as.matrix(na.omit(all_returns_growth))
total_wealth = 100000
weights = c(.142857142857,.142857142857,.142857142857,.142857142857,.142857142857,.142857142857,.142857
holdings = weights * total_wealth
n_days = 20
wealthtracker_growth = rep(0, n_days)
for(today in 1:n_days) {
    return.today = resample(all_returns_growth, 1, orig.ids=FALSE)
    holdings = holdings + holdings * return.today
    total_wealth = sum(holdings)
    wealthtracker_growth[today] = total_wealth
    holdings = weights * total_wealth
}

initial_wealth = 100000
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(.142857142857,.142857142857,.142857142857,.142857142857,.142857142857,.142857142857,.14
    holdings = weights * total_wealth
```

```r
    n_days = 20
    wealthtracker_growth = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns_growth, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker_growth[today] = total_wealth
        holdings = weights * total_wealth
    }
    wealthtracker_growth
}

print(mean(sim2[,n_days] - initial_wealth))
```

```
## [1] 1137.549
```

```r
hist(sim2[,n_days]- initial_wealth, breaks=30)
```



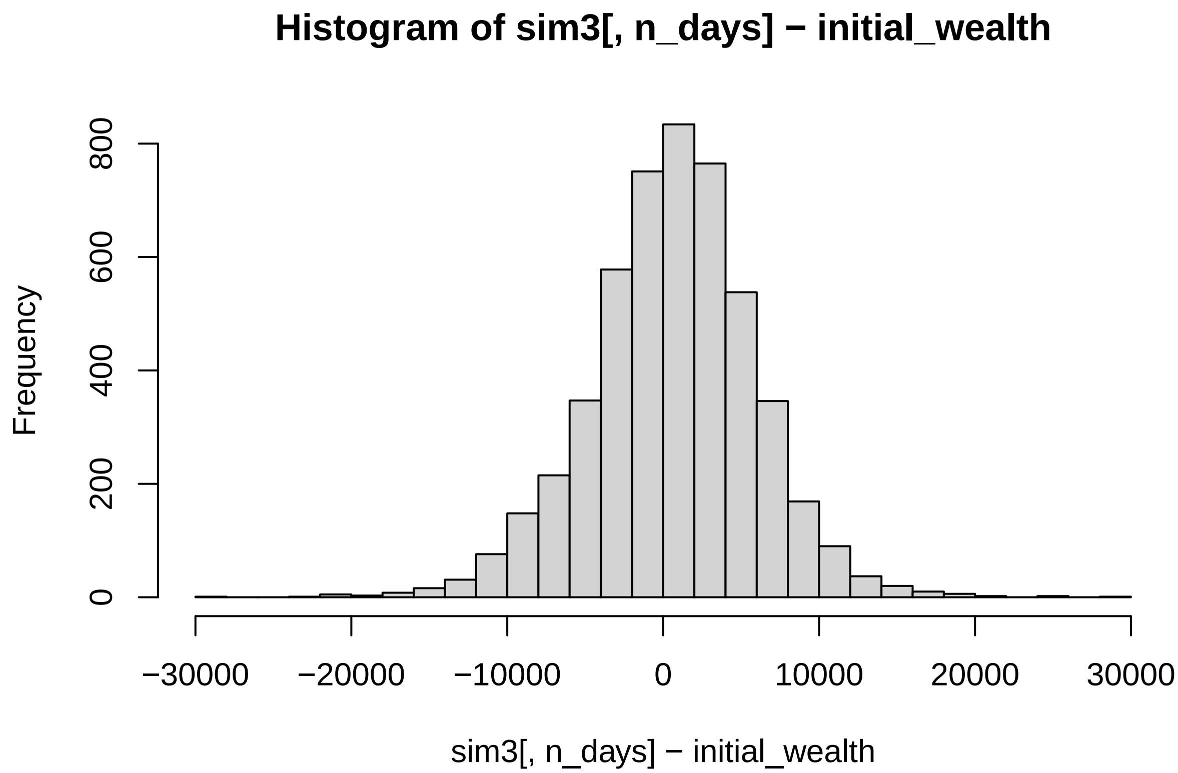

```r
print(quantile(sim2[,n_days]- initial_wealth, prob=0.05))
```

```
##        5%
## -7546.154
```

```r
set.seed(12345)
all_returns_diverse = as.matrix(na.omit(all_returns_diverse))
total_wealth = 100000
weights = c(.1666666667,.1666666667,.1666666667,.1666666667,.1666666667,.1666666667)
holdings = weights * total_wealth
n_days = 20
wealthtracker_diverse = rep(0, n_days)
for(today in 1:n_days) {
    return.today = resample(all_returns_diverse, 1, orig.ids=FALSE)
    holdings = holdings + holdings * return.today
    total_wealth = sum(holdings)
    wealthtracker_diverse[today] = total_wealth
    holdings = weights * total_wealth
}
wealthtracker_diverse <- as.matrix(wealthtracker_diverse)


initial_wealth = 100000
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(.1666666667,.1666666667,.1666666667,.1666666667,.1666666667,.1666666667)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker_diverse = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns_diverse, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker_diverse[today] = total_wealth
        holdings = weights * total_wealth
    }
    wealthtracker_diverse
}

print(mean(sim3[,n_days] - initial_wealth))
```

```
## [1] 601.6955
```

```r
hist(sim3[,n_days]- initial_wealth, breaks=30)
```




```
print(quantile(sim3[,n_days]- initial_wealth, prob=0.05))
```

```
##        5%
## -8435.679
```

In this case, the results point favorably to the Growth portfolio we created. This portfolio had the lowest VAR at 7546.154, meaning that in 5% of scenarios during normal market conditions we might lose that amount of money with the portfolio. This is an interesting outcome of this analysis, especially because Growth stocks are considered more volatile than value stocks. The growth ETF also had a higher average return versus the two other ETF's. Although the growth ETF's were successful based on these metrics, it is worth noting that the diversified ETF had a smaller amount of outcomes below 0 (a loss) based on the histogram. This is valuable information because it could lead to further research assessing whether diversified portfolios have a more favorable distribution of outcomes, even though it's mean outcome is not as good as value portfolios.

Question 4

The first step in assessing this data set was finding out what data would be necessary when segmenting. Although the problem states that filtering for adult and spam content was already done to a certain extent, there were still entries where 2+ annotators labeled tweets in these categories. We first removed all rows in which either of these categories had a value greater than 2, and then created a new data frame as shown below.

```r
socialmarketing = subset(socialmarketing, rowSums(socialmarketing[-1:-36] < 2) > 0)
socialclust = subset(socialmarketing, select = -c(chatter,uncategorized,spam,adult,X) )

summary(socialclust)
```

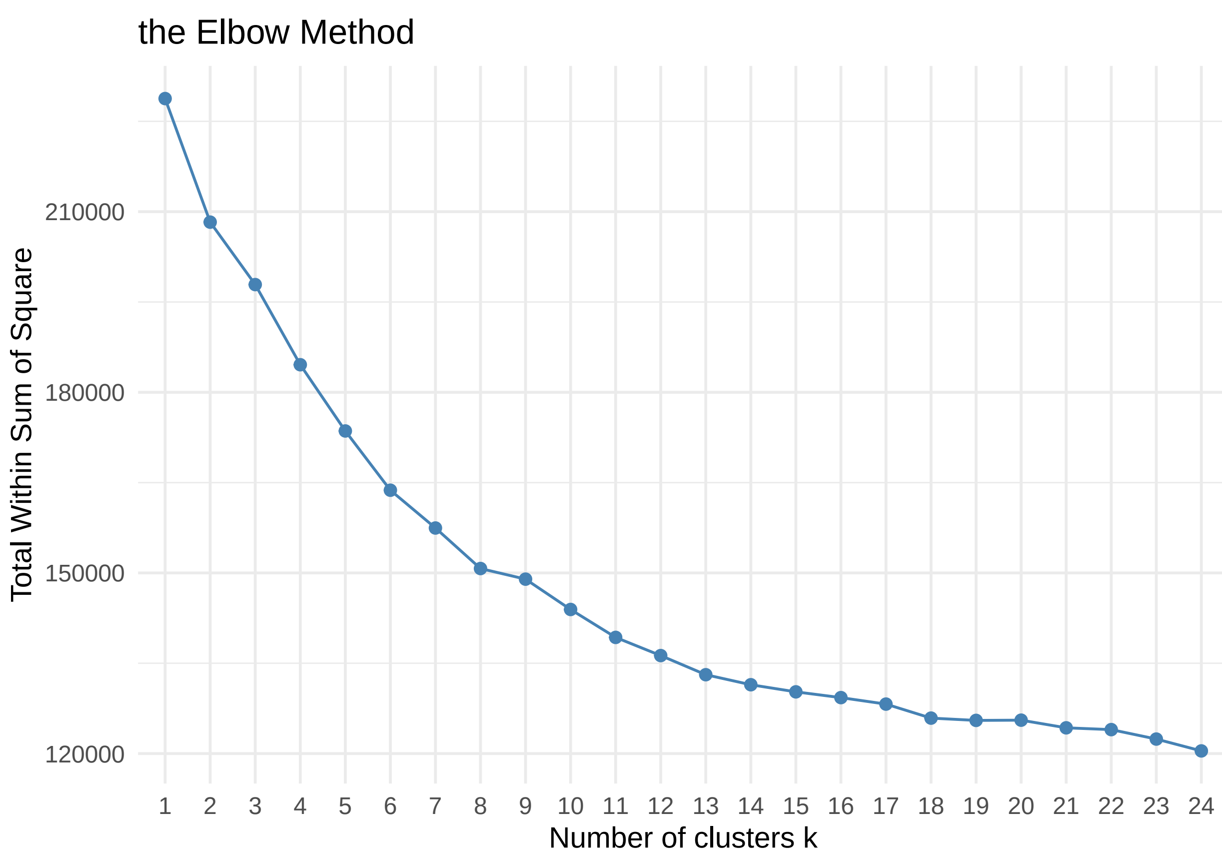
```
##  current_events      travel       photo_sharing       tv_film
##  Min.   :0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:1.000   1st Qu.: 0.000   1st Qu.: 1.000   1st Qu.: 0.000
##  Median :1.000   Median : 1.000   Median : 2.000   Median : 1.000
##  Mean   :1.521   Mean   : 1.573   Mean   : 2.712   Mean   : 1.078
##  3rd Qu.:2.000   3rd Qu.: 2.000   3rd Qu.: 4.000   3rd Qu.: 1.000
##  Max.   :8.000   Max.   :26.000   Max.   :21.000   Max.   :17.000
##  sports_fandom       politics           food           family
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 1.000   Median : 1.000   Median : 1.000   Median : 1.0000
##  Mean   : 1.594   Mean   : 1.807   Mean   : 1.385   Mean   : 0.8544
##  3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.: 2.000   3rd Qu.: 1.0000
##  Max.   :20.000   Max.   :37.000   Max.   :16.000   Max.   :10.0000
##  home_and_garden      music              news         online_gaming
##  Min.   :0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median :0.0000   Median : 0.0000   Median : 0.000   Median : 0.000
##  Mean   :0.5135   Mean   : 0.6843   Mean   : 1.212   Mean   : 1.203
##  3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.000
##  Max.   :5.0000   Max.   :13.0000   Max.   :20.000   Max.   :27.000
##     shopping      health_nutrition  college_uni     sports_playing
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.:0.0000
##  Median : 1.000   Median : 1.000   Median : 1.000   Median :0.0000
##  Mean   : 1.402   Mean   : 2.583   Mean   : 1.562   Mean   :0.6429
##  3rd Qu.: 2.000   3rd Qu.: 3.000   3rd Qu.: 2.000   3rd Qu.:1.0000
##  Max.   :12.000   Max.   :41.000   Max.   :30.000   Max.   :8.0000
##     cooking            eco             computers          business
##  Min.   : 0.000   Min.   :0.0000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 1.000   Median :0.0000   Median : 0.0000   Median :0.0000
##  Mean   : 2.008   Mean   :0.5018   Mean   : 0.6427   Mean   :0.4242
##  3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.:1.0000
##  Max.   :33.000   Max.   :6.0000   Max.   :12.0000   Max.   :6.0000
##     outdoors           crafts          automotive            art
##  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   : 0.7685   Mean   :0.5101   Mean   : 0.8189   Mean   : 0.7128
##  3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000
##  Max.   :12.0000   Max.   :7.0000   Max.   :13.0000   Max.   :18.0000
##     religion          beauty           parenting            dating
##  Min.   : 0.000   Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.0000
##  Median : 0.000   Median : 0.0000   Median : 0.0000   Median : 0.0000
##  Mean   : 1.092   Mean   : 0.7027   Mean   : 0.9087   Mean   : 0.7109
##  3rd Qu.: 1.000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000
```

```
##  Max.   :20.000   Max.   :14.0000   Max.   :14.0000   Max.   :24.0000
##      school          personal_fitness     fashion          small_business
##  Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000
##  Median : 0.0000   Median : 0.000   Median : 0.0000   Median :0.0000
##  Mean   : 0.7586   Mean   : 1.463   Mean   : 0.9986   Mean   :0.3264
##  3rd Qu.: 1.0000   3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000
##  Max.   :10.0000   Max.   :19.000   Max.   :18.0000   Max.   :6.0000
```

Once this initial cleansing was done, we centered and scaled the data.

To identify what would be the ideal amount of clusters to use for our analysis, we used the elbow method. The graph below displays that a cluster around 11-12 seems ideal when considering that is when the curve starts flattening. Thus, we started with a cluster size of 11 for analysis. Although this is generally considered a naive method for estimating cluster size, we thought it would be sufficient when trying to give general segmentation advise in an advertising setting.

```r
fviz_nbclust(X, kmeans, method = "wss", k.max = 24) +
  theme_minimal() + ggtitle("the Elbow Method")
```

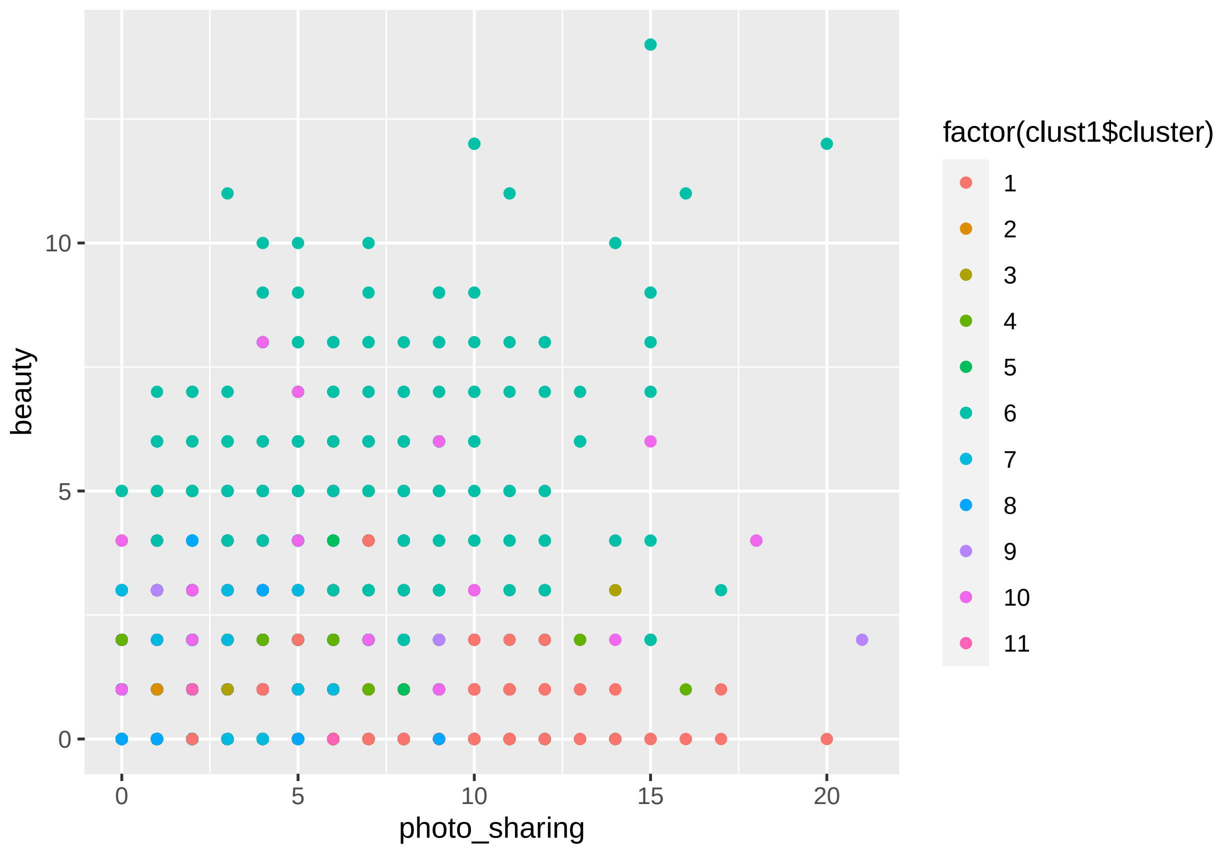


We first looked at the size of each cluster, and printed a few plots showing the relationship between certain variables that seemed comparable.

```r
clust1[["size"]]
```

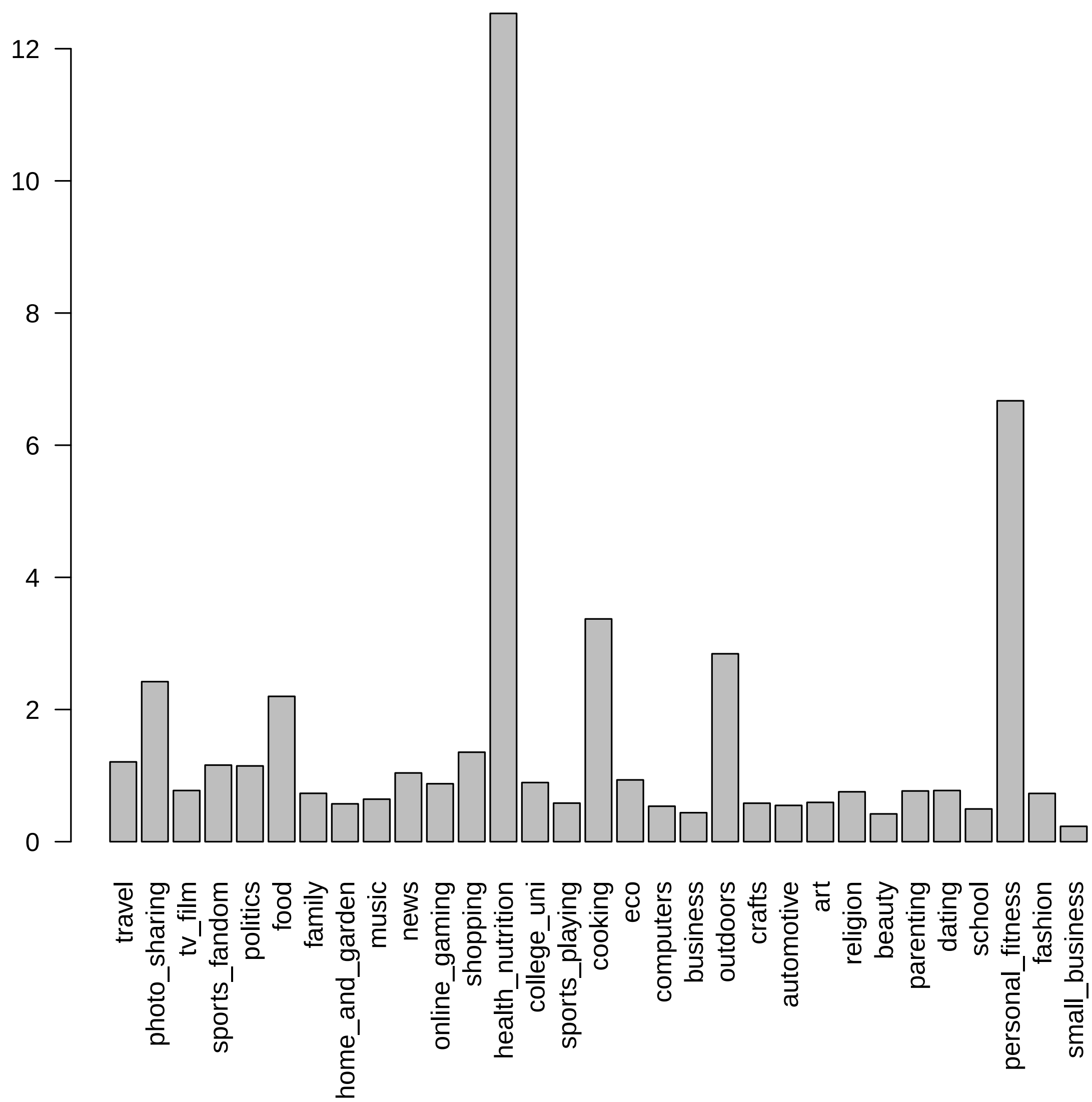
```
##  [1]  825  266  421  324  330  449 2992  723  190  616  245
```

```r
qplot(photo_sharing, beauty, data=socialclust, color=factor(clust1$cluster))
```



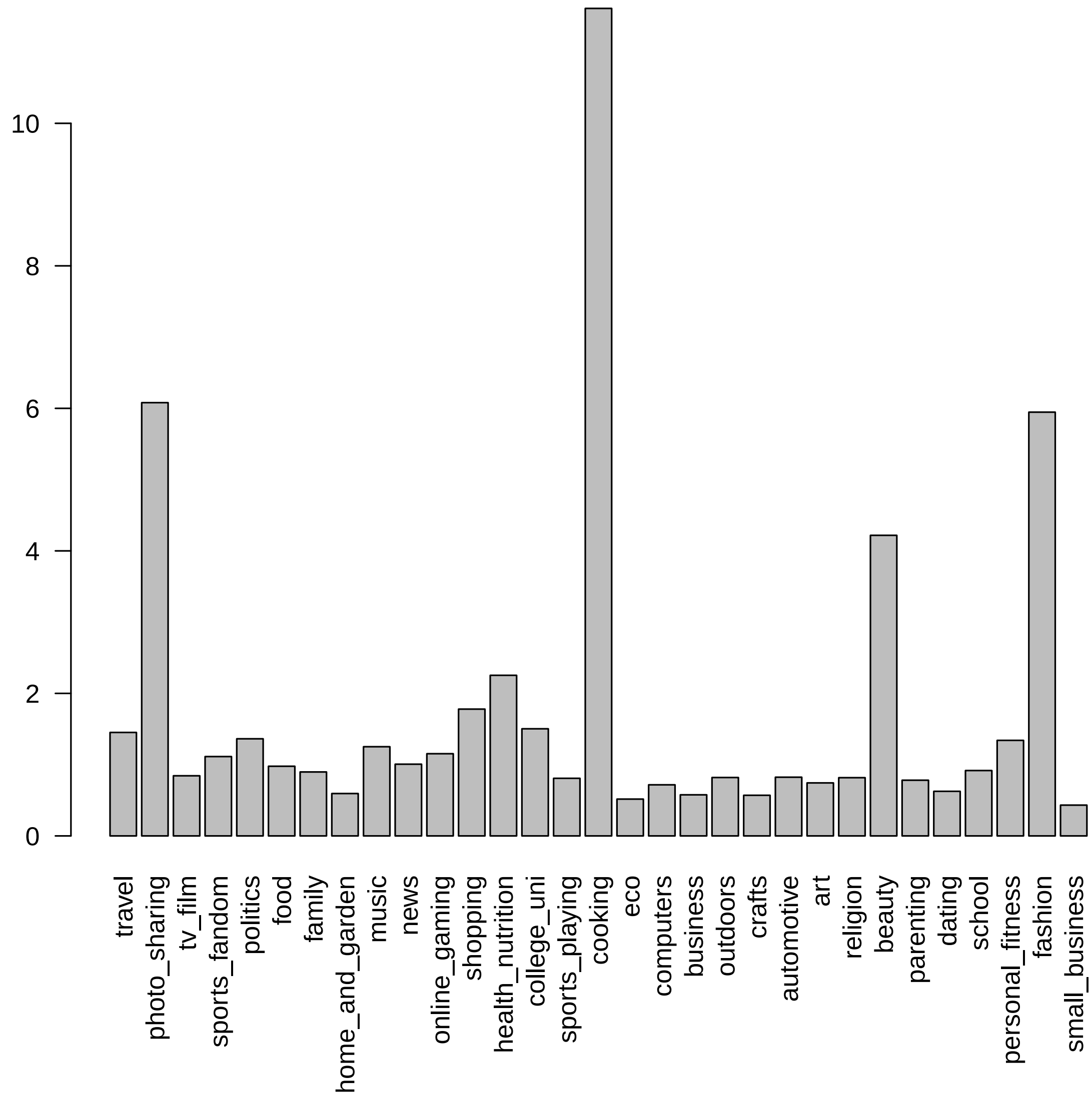
This is one plot showing the relationship between the nutrition feature and the personal fitness feature. We chose to plot these two features based on the assumption that they are in the same family of attributes. It looks like cluster 7 ranks highly in both of these features, so we printed the results of this below.

```r
par(mar = c(9,4,4,2) + 0.1)
cluster8 = (clust1$center[8,]*sigma + mu)
barplot(cluster8, las=2)
```



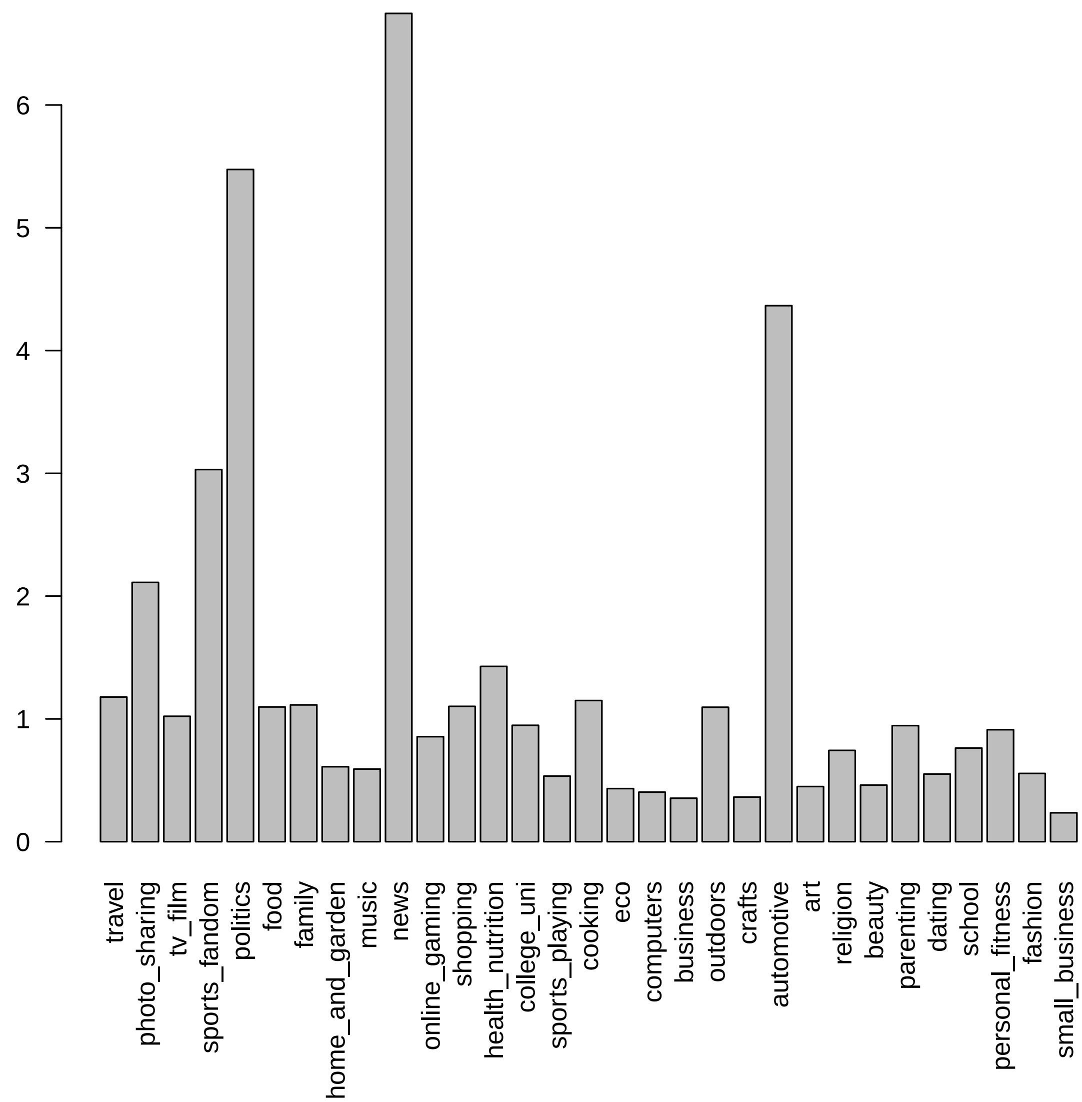
As shown in the bar graph above, this cluster ranks very highly (above the 75%) in health/nutrition, personal fitness, and cooking. This is very valuable information for a consumer brand like ours because these different interests could be packaged into a useful advertising campaign. For example, we could create targeted advertisements that illustrate how our brand can benefit those who are trying to live a health lifestyle. The advertisement could be very fitness oriented and thus entice these kinds of consumers.

```
par(mar = c(9,4,4,2) + 0.1)
cluster6 = (clust1$center[6,]*sigma + mu)
barplot(cluster6, las=2)
```



We then tried looking at other clusters, attempting to understand whether there were any significant relations between variables for advertising. A bar chart of cluster number 6 is shown above. This cluster ranks very highly in cooking, beauty, and photo sharing. Based on these high values, one suggestion we may have in terms of advertising would be to share aesthetically pleasing visuals when trying to target this group since that seems to be among core values of their tweets. Many companies successfully advertise to this market with very visually appealing social media advertising on platforms such as Instagram.

```
par(mar = c(9,4,4,2) + 0.1)
cluster3 = (clust1$center[3,]*sigma + mu)
barplot(cluster3, las=2)
```



One final cluster that is worth mentioning in terms of potential advertising value is displayed above, in which politics and news rank highly. This is more of an unconventional criteria to target in terms of advertising, but it could still be very valuable. For example, many companies have started implementing advertising that ties into the core values of their user base. For example, Nike had a recent advertising campaign supporting professional athletes in the black lives matter movement and kneeling during the national anthem because they understood that a massive segment of their users would likely have political affiliations related to this movement. In the case of our brand, we could have similar campaigns for our segment that have social/political implications. This could help establish an emotional connection to our brand that would not have been apparent otherwise.

Question 5

```r
rm(list=ls())
set.seed(1)
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
```

```r
library(tidyverse)
```

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

## v tibble  3.0.3     v purrr   0.3.4
## v tidyr   1.1.1     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0

## -- Conflicts ---------------------------------------------- tidyverse_conflicts() --
## x purrr::accumulate()       masks foreach::accumulate()
## x NLP::annotate()           masks ggplot2::annotate()
## x mosaic::count()           masks dplyr::count()
## x purrr::cross()            masks mosaic::cross()
## x mosaic::do()              masks plotly::do(), dplyr::do()
## x tidyr::expand()           masks Matrix::expand()
## x plotly::filter()          masks dplyr::filter(), stats::filter()
## x xts::first()              masks dplyr::first()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x dplyr::lag()              masks stats::lag()
## x xts::last()               masks dplyr::last()
## x tidyr::pack()             masks Matrix::pack()
## x mosaic::stat()            masks ggplot2::stat()
## x mosaic::tally()           masks dplyr::tally()
## x tidyr::unpack()           masks Matrix::unpack()
## x purrr::when()             masks foreach::when()
```

```r
library(slam)
library(proxy)
```

```
##
## Attaching package: 'proxy'

## The following object is masked from 'package:Matrix':
##
##     as.matrix

## The following objects are masked from 'package:stats':
##
##     as.dist, dist

## The following object is masked from 'package:base':
##
##     as.matrix
```

```r
library(caret)
```

```
##
## Attaching package: 'caret'

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

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

```r
#the readerplain function
readerPlain = function(fname){
  #convert a txt file to a PlainTextDocument
              readPlain(elem=list(content=readLines(fname)),
                          id=fname, language='en') }
```

#Set up train and test data *To start with, we created a process to transform the raw .txt file folders into TF-IDF matrices which we could run PCA on. First, we used a for loop to put all .txt files into a corpus and their corresponding authors into a separated author list (as outcome). Then we did tokenization on the corpus and obtained the DTM and TF-IDF matrix from there. We applied this process to C50Train and C50Test separately. We ended up having two TF-IDF matrices from the two folders.* ## Train data ### Create the train corpus which contains all the articles from train and a list of their author names

```r
#a list of links to authors' folders(folders each contain 50 articles for its author)
author_folder = Sys.glob('C:/Users/rkapistalam/Downloads/ReutersC50/ReutersC50/C50train/*')
#article file link list and author list
arti_list = c() #a list of all article docs (50 articles x 50 authors)
author_list_tr = c() # a list of author names corresponding every doc in the corpus
for (author in author_folder) {
```

```
  name = substring(author, first = 63) #slice the folder link to obtain the name of the author
  articles = Sys.glob(paste0(author, '/*.txt')) #a list of all 50 articles from one author
  arti_list = append(arti_list, articles)
  author_list_tr = append(author_list_tr, rep(name, times = length(articles)))
}
#use the link list to create a list of readPlaindocs
c50_tr = lapply(arti_list, readerPlain)
#clean the doc names and apply them to c50_tr
mynames = arti_list %>%
    { strsplit(., '/', fixed=TRUE) } %>%
    { lapply(., tail, n=2) } %>%
    { lapply(., paste0, collapse = '') } %>%
    unlist
names(c50_tr) = mynames
#create the train corpus
cps_tr_raw = Corpus(VectorSource(c50_tr))
```

Tokenization on train corpus

```
cps_tr = cps_tr_raw %>%
  tm_map(content_transformer(tolower))  %>%             # make everything lowercase
  tm_map(content_transformer(removeNumbers)) %>%        # remove numbers
  tm_map(content_transformer(removePunctuation)) %>%    # remove punctuation
  tm_map(content_transformer(stripWhitespace)) %>%      # remove excess white-space
  tm_map(content_transformer(removeWords),
        stopwords("en"))                                #remoce stop words
```

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

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

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

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

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

```
cps_tr
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 2500
```

create the DTM and TF-IDF matrix for train corpus

```
#the DTM matrix
DTM_tr = DocumentTermMatrix(cps_tr) %>%
  removeSparseTerms(.99) #remove sparse elements which don't show in 95% of the articles
#the TF-IDF matrix
tfidf_tr = as.matrix(weightTfIdf(DTM_tr))
```

Test data

Create the test corpus which contains all the articles from test and a list of their author names

```
#a list of links to authors' folders(folders each contain 50 articles for its author)
author_folder = Sys.glob('C:/Users/rkapistalam/Downloads/ReutersC50/ReutersC50/C50test/*')
#article file link list and author list
arti_list = c() #a list of all article docs (50 articles x 50 authors)
author_list_te = c() # a list of author names corresponding every doc in the corpus
for (author in author_folder) {
  name = substring(author, first = 63) #slice the folder link to obtain the name of the author
  articles = Sys.glob(paste0(author, '/*.txt')) #a list of all 50 articles from one author
  arti_list = append(arti_list, articles)
  author_list_te = append(author_list_te, rep(name, times = length(articles)))
}
#use the link list to create a list of readPlaindocs
c50_te = lapply(arti_list, readerPlain)
#clean the doc names and apply them to c50_te
mynames = arti_list %>%
    { strsplit(., '/', fixed=TRUE) } %>%
    { lapply(., tail, n=2) } %>%
    { lapply(., paste0, collapse = '') } %>%
    unlist
names(c50_te) = mynames
#create the test corpus
cps_te_raw = Corpus(VectorSource(c50_te))
```

Tokenization on test corpus

```
cps_te = cps_te_raw %>%
  tm_map(content_transformer(tolower))  %>%             # make everything lowercase
  tm_map(content_transformer(removeNumbers)) %>%        # remove numbers
  tm_map(content_transformer(removePunctuation)) %>%    # remove punctuation
  tm_map(content_transformer(stripWhitespace)) %>%      # remove excess white-space
  tm_map(content_transformer(removeWords),
         stopwords("en"))                               #remoce stop words
```

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

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

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

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

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

```
cps_te
```

```
## <<SimpleCorpus>>
## Metadata:  corpus specific: 1, document level (indexed): 0
## Content:  documents: 2500
```

create the DTM and TF-IDF matrix for test corpus

```
#the DTM matrix
DTM_te = DocumentTermMatrix(cps_te, control = list(dictionary = colnames(DTM_tr)))
#This ensure the DTM_te has the same col structure (the terms) as DTM_tr
#the TF-IDF matrix
tfidf_te = as.matrix(weightTfIdf(DTM_te))
```

```
## Warning in weightTfIdf(DTM_te): unreferenced term(s):
## cusersrkapistalamdownloadsreuterscreuterscctrainaaronpressmannewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainalancrosbynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainalexandersmithnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainbenjaminkanglimnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainbernardhickeynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainbraddorfmannewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraindarrenschuettlernewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraindavidlawdernewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainednafernandesnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainericauchardnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainfumikofujisakinewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraingrahamearnshawnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainheatherscoffieldnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjanlopatkanewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjanemacartneynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjimgilchristnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjowinterbottomnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjoeortiznewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjohnmastrininewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainjonathanbirtnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainkarlpenhaulnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainkeithweirnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainkevindrawbaughnewsmltxt
```

```
## cusersrkapistalamdownloadsreuterscreuterscctrainkevinmorrisonnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainkirstinridleynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainkouroshkarimkhanynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainlydiazajcnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainlynneodonnellnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainlynnleybrowningnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmarcelmichelsonnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmarkbendeichnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmartinwolknewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmatthewbuncenewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmichaelconnornewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainmuredickienewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainnicklouthnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainpatriciacomminsnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainpeterhumphreynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainpierretrannewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainrobinsidelnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainrogerfillionnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainsamuelperrynewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainsarahdavisonnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainscotthillisnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainsimoncowellnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraintaneelynnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraintheresepolettinewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraintimfarrandnewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctraintoddnissennewsmltxt
## cusersrkapistalamdownloadsreuterscreuterscctrainwilliamkazernewsmltxt
```

PCA on TF-IDF Matrix

We first did some clean-up on train and test matrices to make sure they don't have empty columns and have the same size. Then we run PCA on the train TF-IDF matrix. ## clean up empty columns in TF-IDF matrices

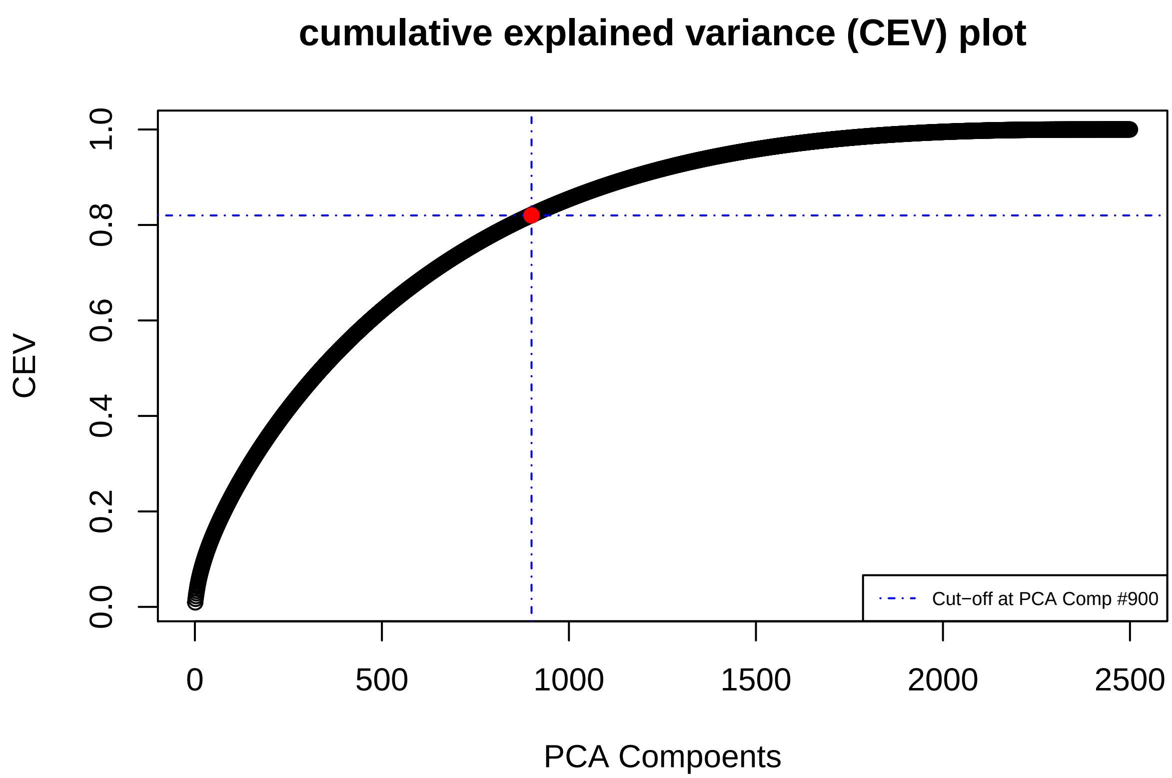
```r
#filter out empty columns
tfidf_tr_1 = tfidf_tr[,which(colSums(tfidf_tr) != 0)]
tfidf_te_1 = tfidf_te[,which(colSums(tfidf_te) != 0)]
#make sure the train and test matrices have the same size
tfidf_te_1 = tfidf_te_1[,intersect(colnames(tfidf_te_1),colnames(tfidf_tr_1))]
tfidf_tr_1 = tfidf_tr_1[,intersect(colnames(tfidf_te_1),colnames(tfidf_tr_1))]
```

Set up PCA for train X

```r
c50_pca = prcomp(tfidf_tr_1, scale = TRUE)
```

Plot the cumulative explained variance(CEV) by PCA components

```
cum_v = cumsum(c50_pca$sdev^2 / sum(c50_pca$sdev^2))
plot(cum_v, xlab = "PCA Compoents",
     ylab = "CEV",
     main = "cumulative explained variance (CEV) plot")
abline(v = 900, col="blue", lty=10)
abline(h = .82, col="blue", lty=10)
points(x = 900, y = .82, col="red", lty=5, pch = 19)
legend("bottomright",
       legend=c("Cut-off at PCA Comp #900"),
       col=c("blue"), lty=10, cex=0.6)
```



We found that CEV reaches somewhere above 80% (which is good enough) at the 900th of the PCA components, thus we will use these 900 PCA comps to train our models. # Model Comparison in Author Attribution *We used Randomforest and KNN models to do author attribution on the test data and compared their performance.* ## Set up train and test data (author ~ pca comps)

```
#using the first 400 pca comps
#train data
c50_tr = data.frame(c50_pca$x[,1:900])
c50_tr['author'] = as.factor(author_list_tr)
#test data
c50_load = c50_pca$rotation[, 1:900]
c50_te_pre = scale(tfidf_te_1) %*% c50_load
c50_te = as.data.frame(c50_te_pre)
c50_te['author'] = as.factor(author_list_tr)
```

RandomForest

We did a 5-fold CV in order to find out the optimal mty for the RF model. Then we did author attribution using the optimal RF model. ## 5-fold CV for optimal mty

```
#this chunk will cost about 10min to run
#Do a 10-fold CV with 3 repeats in each to pick up optimal tree numbers (B values)
library('e1071')
control = trainControl(method="repeatedcv", number=5, repeats=3)
#set up arguments for train()
metric =  "Accuracy"
mtry = sqrt(ncol(c50_tr) - 1)
tunegrid = expand.grid(.mtry=mtry)
#model fit
model_rf = train(author ~ .,
                 data = c50_tr,
                 method="rf",
                 metric = metric,
                 tuneGrid = tunegrid,
                 trControl = control)
#optimal mty = 30
```

Calculate accuracy rate on test data

```
#prediction on test data
c50_rf_te_pred = predict(model_rf,
                         newdata = c50_te)
#confusion matrix
conf_rf = confusionMatrix(c50_rf_te_pred, data = c50_te$author)
c50_rf_accuracy = conf_rf$overall[1]
cat('Random Forest Model gives us an accruacy rate of', c50_rf_accuracy)
```

```
## Random Forest Model gives us an accruacy rate of 0.5792
```

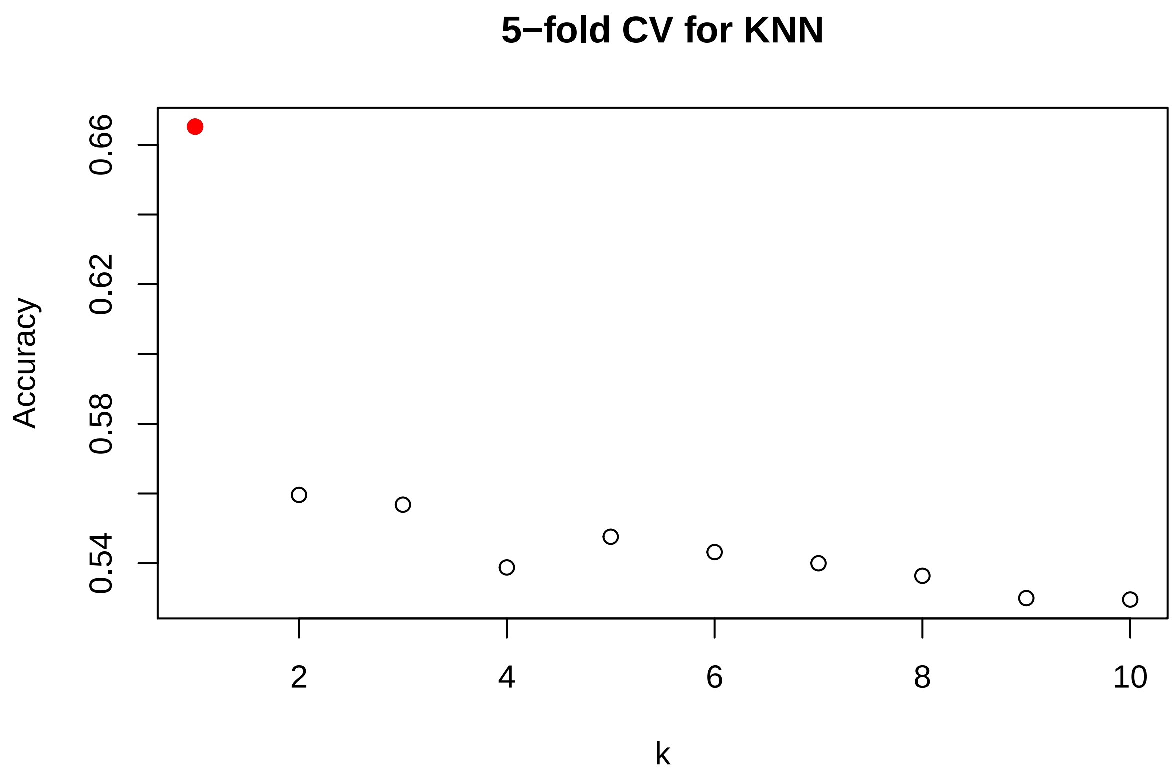
KNN

We did a 5-fold CV in order to find out the optimal k for the KNN model. Then we did author attribution using the optimal KNN model. #5-fold CV to pick up the optimal k

```
#this chunk will cost about 5min to run
#set up a 5-fold cv
tr_control = trainControl(method  = "cv",
                          number  = 5)
#Test knn models with k = 1-10
c50_knn = train(author ~ .,
                method = "knn",
                tuneGrid = expand.grid(k = 1:10),
                trControl = tr_control,
                metric = "Accuracy",
                data = c50_tr)
#optimal k=1
```

Plot accruacy level for k = 1-10

```
k_accuracy = c50_knn$results[, c(1, 2)]
plot(k_accuracy, xlab = "k",
     ylab = "Accuracy",
     main = "5-fold CV for KNN")
points(x = 1, y = k_accuracy[1, 2], col="red", lty=5, pch = 19)
```



Since the KNN with k=1 has the hightest accuracy, we pick optimal k = 1. ### Calculate accuracy rate on test data

```
#prediction on test data
c50_knn_te_pred = predict(c50_knn,
                          newdata = c50_te)
#compare prediction with actual values
compare_knn = as.data.frame(cbind(actual = c50_te$author,
                                  pred = c50_knn_te_pred))
compare_knn['hit'] = ifelse(compare_knn$actual == compare_knn$pred, 1, 0)
c50_knn_accuracy = sum(compare_knn$hit) / nrow(compare_knn)
cat('KNN Model gives us an accruacy rate of', c50_knn_accuracy)
```

```
## KNN Model gives us an accruacy rate of 0.3288
```

Conclusion

Among the two models we used (RandomForest and KNN), RandomForest model has the best accuracy of 57% while KNN gives an accuracy of 32%. There are potential ways of improvement to our models' performance. First we could implement more PCA components when fitting the models in order to capture more data variance. Second in our case we ignored the words in test data which don't show in the train data. We will figure out a way to include these words in the future to improve our model performance. Lastly we will try other classification models like logistic regression in doing author attribution in the future.

Question 6

The first step in this problem was to properly read the grocery data as a transaction so that the Apriori algorithm could be implemented. The following code does this, and then inspects the first few values in the resulting data.

```
grocery <- read.transactions('groceries.txt', sep = ',')
inspect(grocery[1:5])
```

```
##     items
## [1] {citrus fruit,
##      margarine,
##      ready soups,
##      semi-finished bread}
## [2] {coffee,
##      tropical fruit,
##      yogurt}
## [3] {whole milk}
## [4] {cream cheese,
##      meat spreads,
##      pip fruit,
##      yogurt}
## [5] {condensed milk,
##      long life bakery product,
##      other vegetables,
##      whole milk}
```

The following is a summary of the same data.

```
summary(grocery)
```

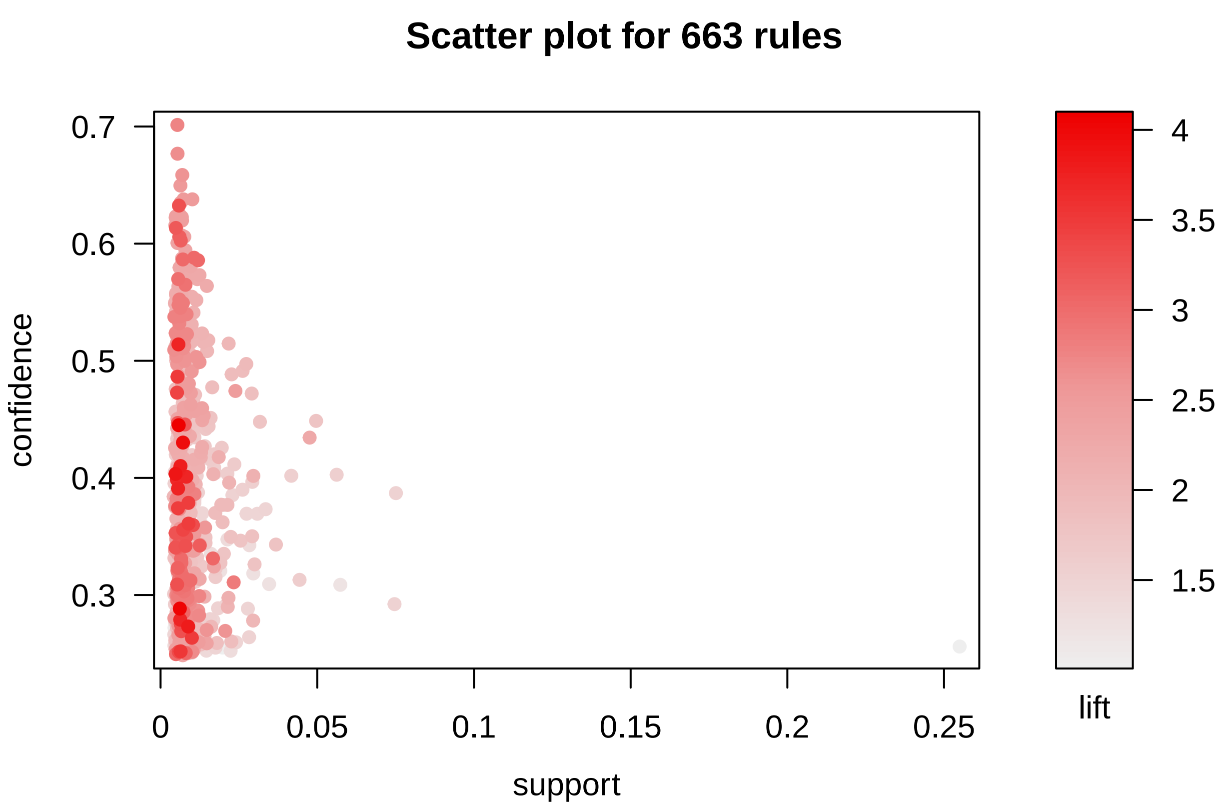
```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             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
```

The next step in our process was to run the Apriori algorithm and conduct exploratory analysis to understand what lift and confidence levels to set, and also understand what specific grocery items would be helpful to analyze. The following are two plots highlighting lift and confidence levels.

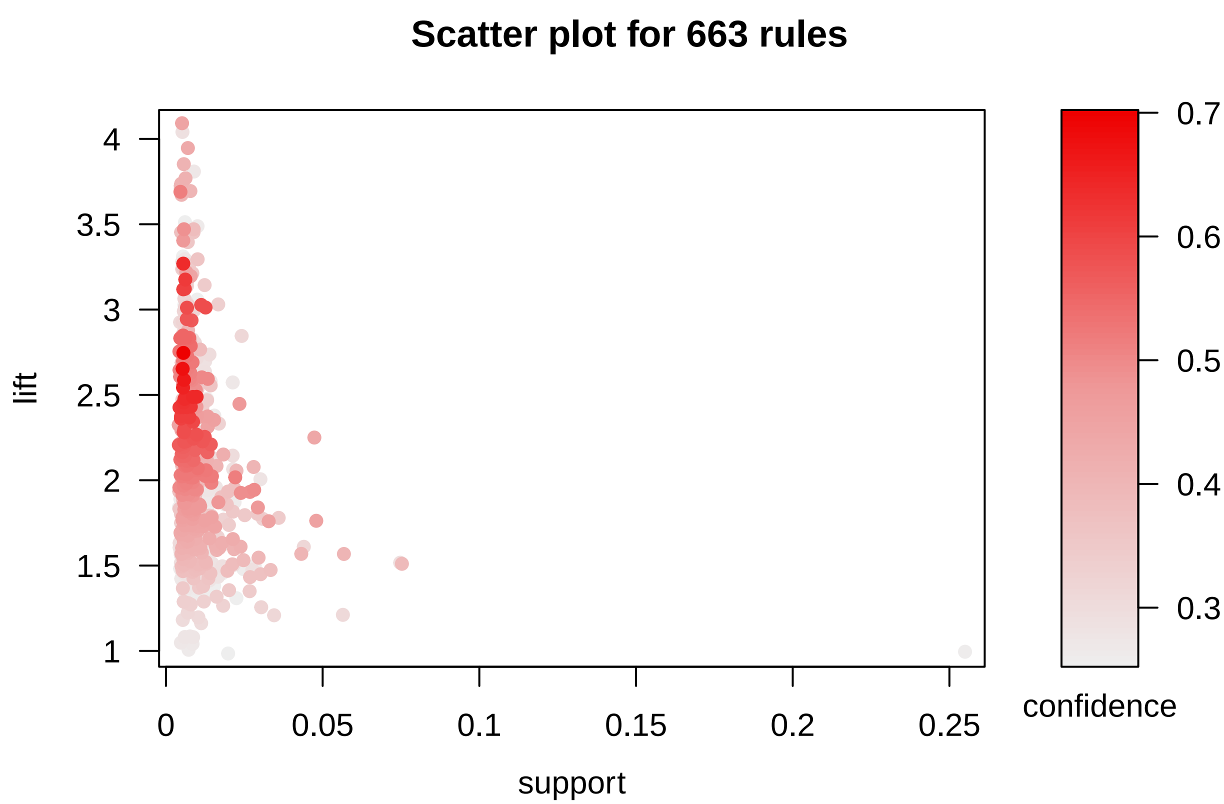
```r
plot(grocrules)
```

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



```r
plot(grocrules, measure = c("support", "lift"), shading = "confidence")
```

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



Based on the plots above, we started by sub-setting the data with a lift over 3 and a confidence level of over .5. We chose these benchmarks because it would help cut down the data to find association rules that could be most beneficial discoveries. The following is a summary of this information.

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

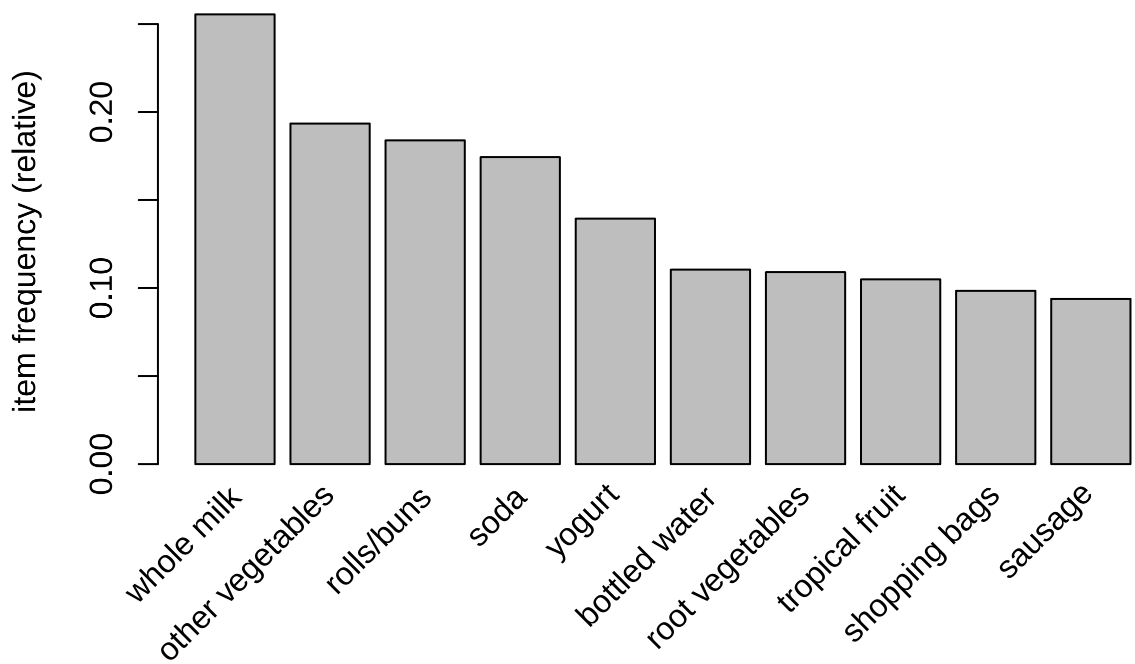
```
##     lhs                       rhs                    support confidence    coverage     lift count
## [1] {onions,
##      root vegetables}      => {other vegetables} 0.005693950  0.6021505 0.009456024 3.112008    56
## [2] {curd,
##      tropical fruit}       => {yogurt}           0.005287239  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
```

The primary interesting observation from the above data is the relationship between Curd, tropical fruit and yogurt in line 2. This seems to be a common combination, in which the confidence is above .5 and the lift is 3.69. If consulting for this grocery store, we might suggest trying to put these items close together to allow customers to easily buy them all.

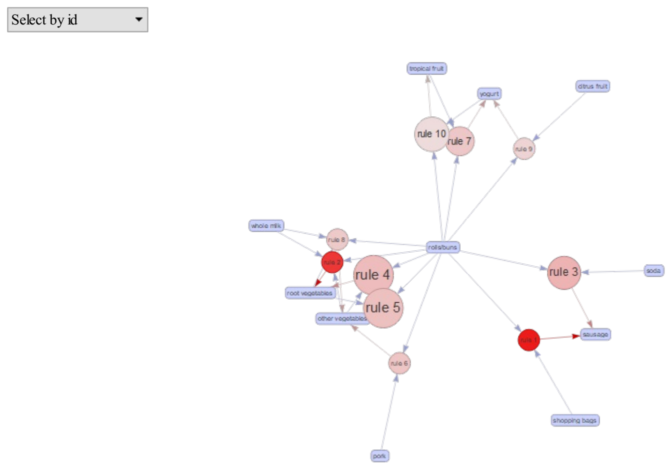
Although this may have been a useful observation, most of the other relationships are not that informative. Most of the relationships are with "Other vegetables" which is not clear or useful. Thus, we then tried identifying the most common grocery items to isolate based on popularity.

```
itemFrequencyPlot(grocery, topN =10)
```



The graph above shows the item frequency of different items. We focused on rolls/buns, soda, and yogurt when visualizing association rules because they appear often in the data.

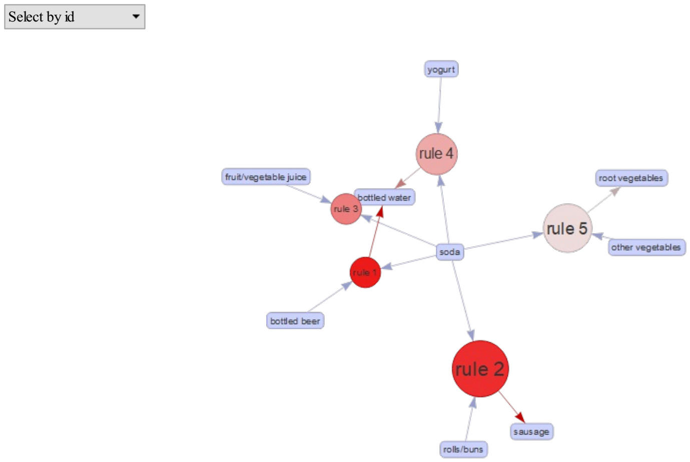
```
bunrules <- subset(grocrules, items %in% 'rolls/buns')
top10bunrules <- head(bunrules, n = 10, by = "lift")
plot(top10bunrules, method = "graph",  engine = "htmlwidget")
```



The illustration above displays the top 10 association rules for buns, ranked by lift. Buns seem to be very commonly attributed with soda, sausage and vegetables. This is a valuable observation because it suggests that buns can be promoted for a variety of different events. For example, buns, soda, and sausage are very commonly associated and it might be because many American events (such as sporting events) have this combination of items as food options. This could be useful information for a grocery store because they could put these items close together on important sports days or holidays such as Independence Day.

The next feature we focused on was soda, as shown below.

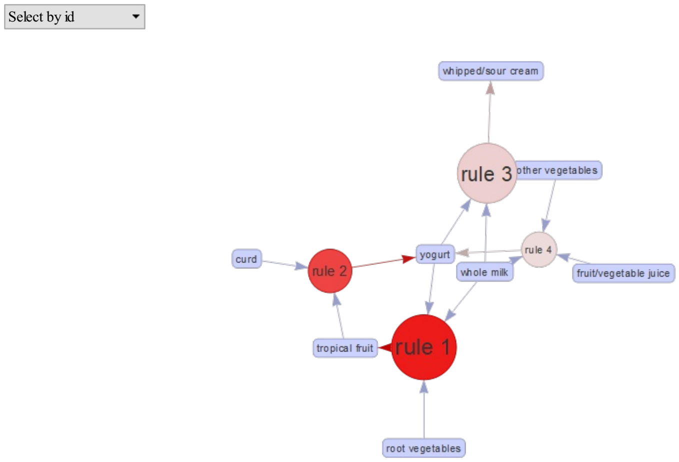
```
sodarules <- subset(grocrules, items %in% 'soda')
top10sodarules <- head(sodarules, n = 5, by = "lift")
plot(top10sodarules, method = "graph",  engine = "htmlwidget")
```



The results of this association illustration make a lot of sense when you think about the typical structure of a U.S grocery store. Drinks are typically put in the same aisle, whether it is soda or juice items. In the above image, soda is most often associated with other drinks such as alcohol, water and fruit juice which aligns with this general idea. As mentioned in the previous analysis, soda is also associated with food items like buns/sausages which makes sense considering typical American food selection at events.

Finally, we illustrated the relationship with yogurt and different grocery items.

```
yogurtrules <- subset(grocrules, items %in% 'yogurt')
top10yogurtrules <- head(yogurtrules, n = 4, by = "lift")
plot(top10yogurtrules, method = "graph",  engine = "htmlwidget")
```



It is interesting that yogurt is associated with different fruits, whipped cream, and curd. This could mean that common uses of yogurt are in dessert or snack combinations with theses items. This could be useful information for a grocery store to potentially put refrigerated yogurt next to the fruit section or dessert section to encourage this group purchase.