Exercises

Bryant Leal, Adam Hoard, Kevin Huan, Suchit Das

08/18/2020

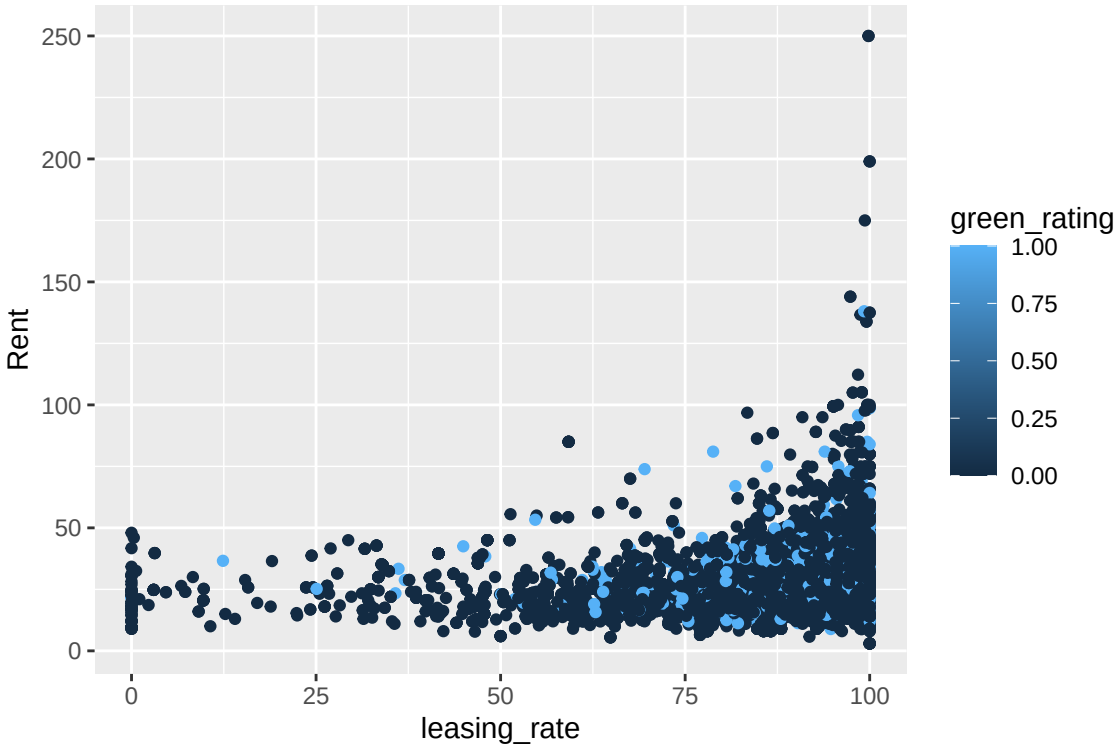
Visual Story Telling Part 1: Green Buildings

Green certification has been very popular in recent times, and it is important that the effects of being green certified are stratified from other effects since the economic implications of this decision can mean 5 million dollars invested or 5 million dollars wasted. To begin, we will be cleaning the data to include those buildings in the top 75% of buildings for size, which starts at 50891 Square Feet.

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##    1624   50891  128838  234638  294212 3781045
```

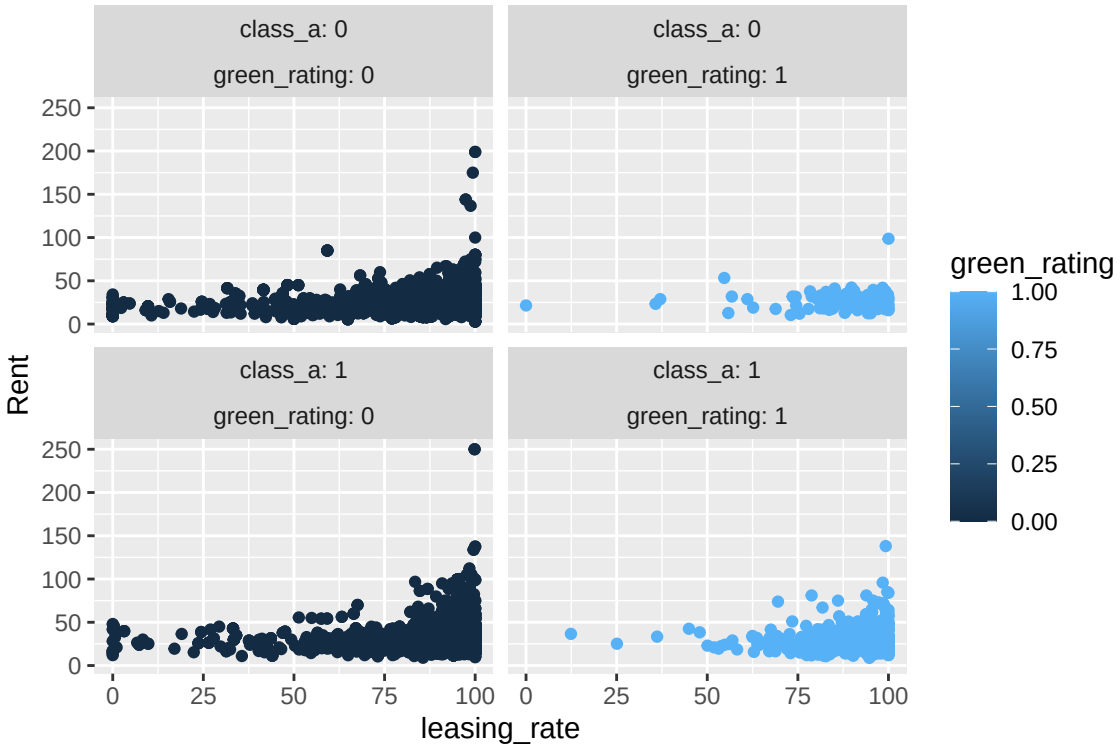
After cleaning the data, the leasing rate of a building (as a proxy for popularity) will be plotted against rent, and a color indicator will be used to get a high level stratification between a green rated building, and a regular building.

```
## Warning: package 'ggplot2' was built under R version 3.6.3
```

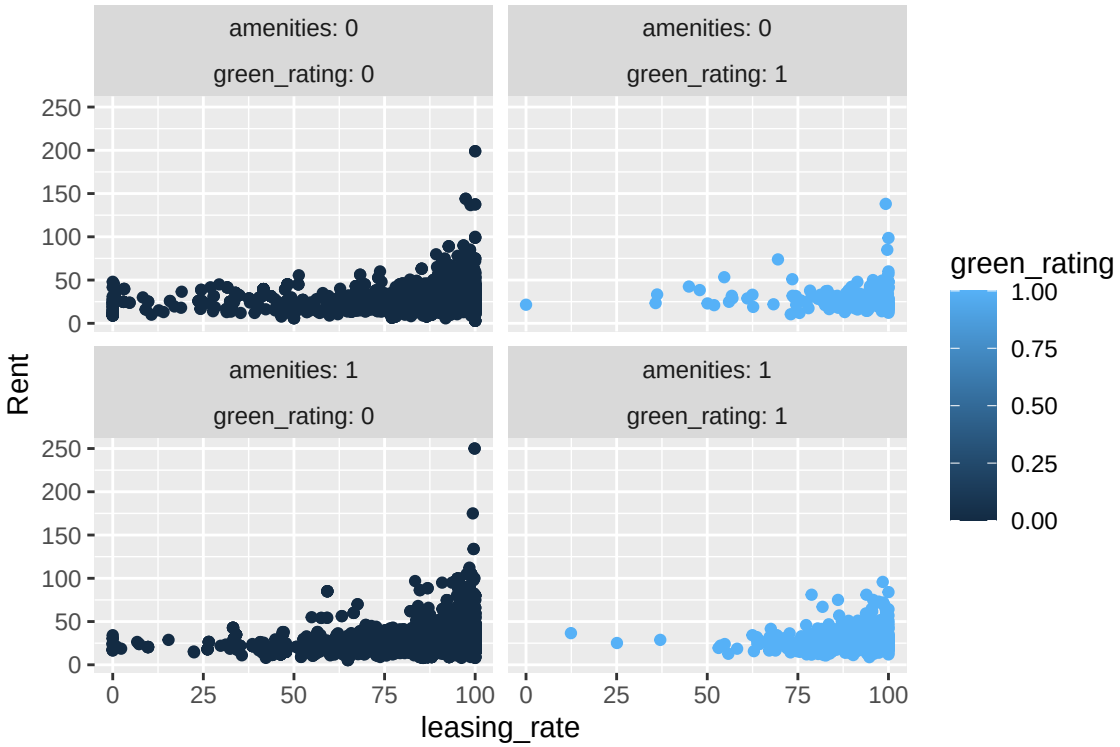


It seems that there are more non-green rated buildings that have the highest rents when compared to green rated buildings. Lets seperate these effects.

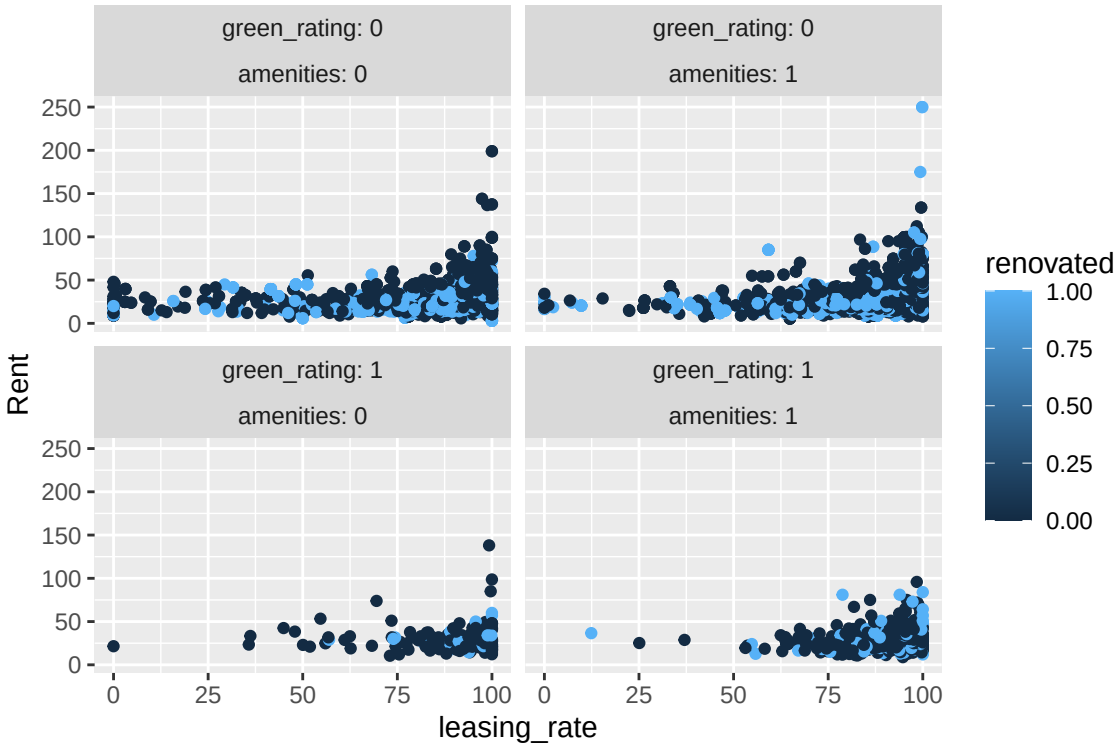
Buildings rated green seem to cluster toward a higher leasing rate yet do not exhibit higher rents. Could the size of the data set skew our insights?



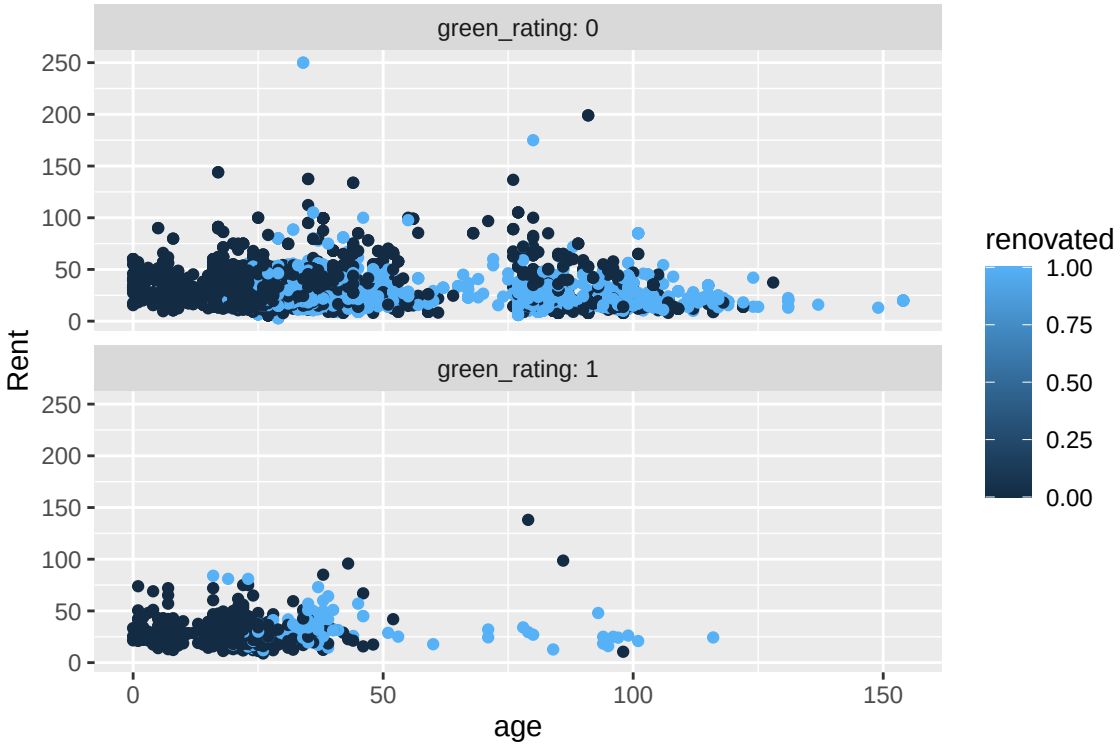
We stratify by whether the building is green certified and if it is considered a high quality building. It is clear that rents for green buildings not in class_a have flat rents when compared to the non-certified counterpart. It is also interesting to see that class for non certified buildings did not change the pattern significantly. Lets check to see if we can stratify by amenities offered.



Replacing class with amenities changes the tail of the non-certified buildings with amenities. It begins to thin out and look like the tail of the green rated buildings in class_a. We also see extereme outliers for rent in the bottom left plot. There is evidence that green rated buildings may not actually yield economic benefit on their own.

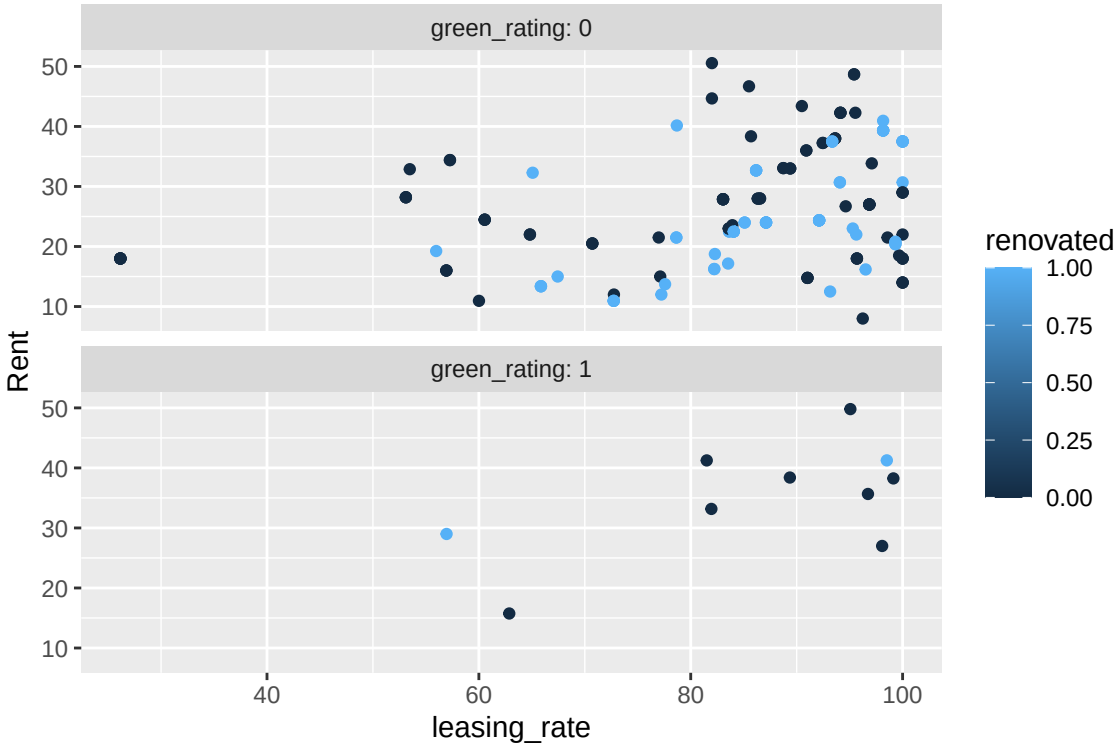


In this situation, the data is faceted by green rating and amenities while seperating the points by the indicator renovated. It is clear from these plots that while there is some variation in rent for green rated buildings with amenities, there exists even more variation for buildings that are not green rated, but do include amenities.



Lastly, we want to see if green rated buildings last much longer than traditional buildings. The X axis was

changed to the age of the building in years. In both instances, it seems that older buildings in general usually need a renovation, which does not come as a surpise. However, the data shows an overwhelming number of non-green certified buildings being much older than green rated buildings. While this may not be an indication of green building longevity, it does scrutinize many of the benefits that green rated buildings are suppose to have.



A way to potentially deal with the confounding of the variables is to shrink the data down to a smaller subset that would be indicative of the investment being made. For example, the data is subsetted by the top 75% of buildings by square footage and buildings that 15 stories. That way, we measure the relationships that are relevant to the investment at hand.

Conclusion

The stats guru's model is straight forward and easy to understand, but has some issues. Mainly, the analysis does not take into account the interrelated dependencies that the various predictors have with each other, so creating an answer with superficial data that does not respect these relationships may lead to an uninformed decision. For example, the analysis takes the median of rent for green rated buildings versus normal buildings, but could not account for buildings of higher quality or those that provided amenities. We cannot assume that a green building provides value, and the data seems to support the idea that a green building on its own does not value when referencing higher rents.

Visual Storing Telling Part 2

R Markdown

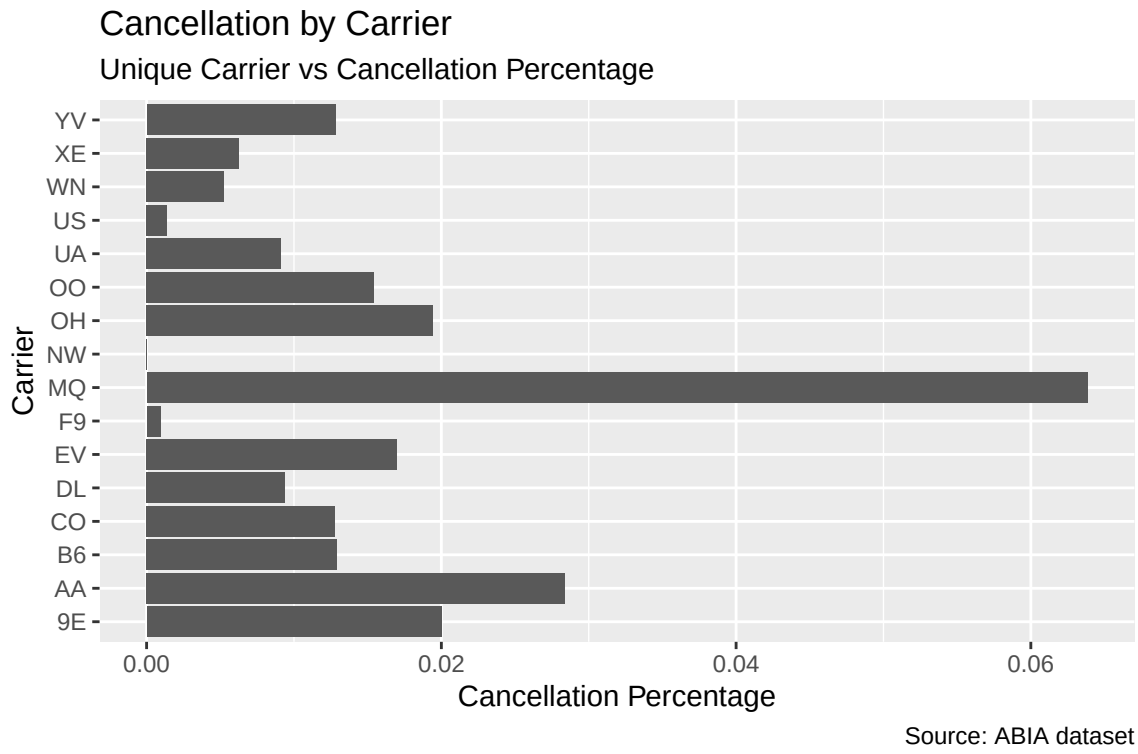
```
ABIA = read.csv('ABIA.csv')
```

#Question: Which Carriers are the most Unreliable?

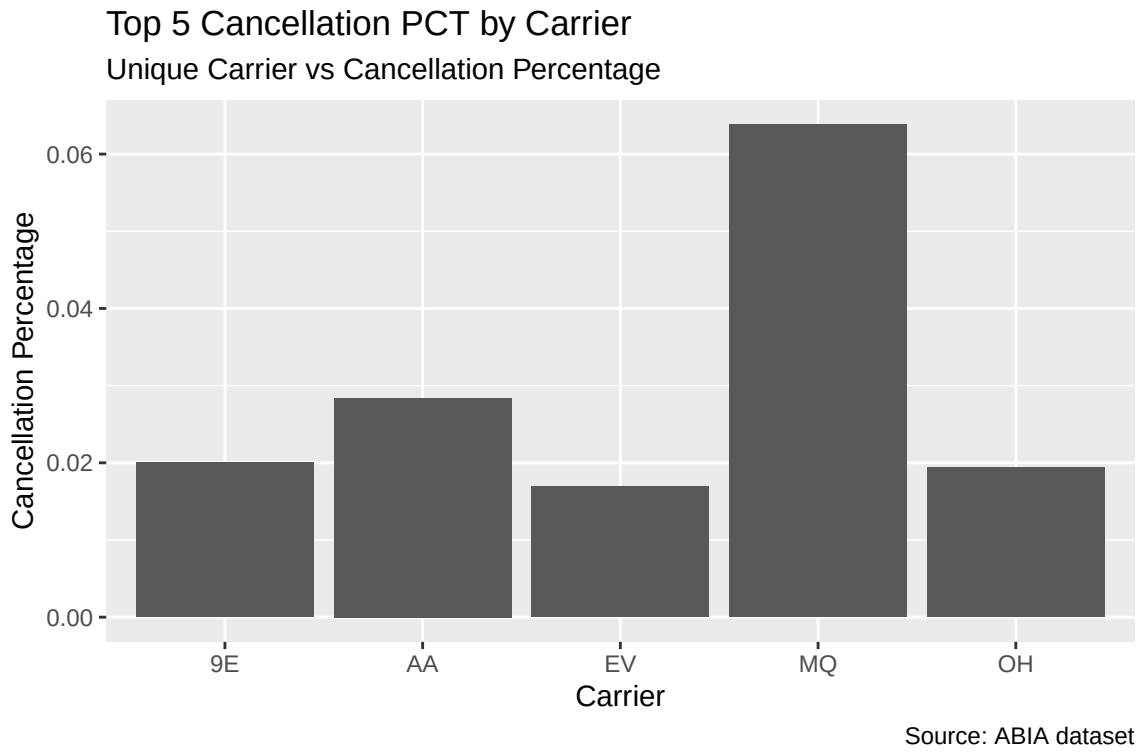
```
d1 = ABIA %>%
  group_by(UniqueCarrier) %>%
  summarize(canc_pctAL = sum(Cancelled=='1')/n())
d1
```

```
## # A tibble: 16 x 2
##    UniqueCarrier canc_pctAL
##    <fct>              <dbl>
##  1 9E              0.0200
##  2 AA              0.0284
##  3 B6              0.0129
##  4 CO              0.0128
##  5 DL              0.00937
##  6 EV              0.0170
##  7 F9              0.000938
##  8 MQ              0.0638
##  9 NW              0
## 10 OH              0.0194
## 11 OO              0.0154
## 12 UA              0.00911
## 13 US              0.00137
## 14 WN              0.00525
## 15 XE              0.00628
## 16 YV              0.0128
```

```
ggplot(data = d1) +
  geom_bar(mapping = aes(x=UniqueCarrier, y=canc_pctAL), stat='identity')+
  coord_flip() +
  labs(title="Cancellation by Carrier",
       subtitle="Unique Carrier vs Cancellation Percentage",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Cancellation Percentage")
```




```
d1 %>%
  arrange(desc(canc_pctAL)) %>%
  slice(1:5) %>%
  ggplot(., aes(x=UniqueCarrier, y=canc_pctAL))+
  geom_bar(stat='identity')+
  labs(title="Top 5 Cancellation PCT by Carrier",
        subtitle="Unique Carrier vs Cancellation Percentage",
        caption="Source: ABIA dataset",
        x="Carrier",
        y="Cancellation Percentage")
```



9E, AA, EV MQ and OH seem to be the most unreliable.

```
#Changining Number notation to month Notation
ABIA$Month2[ABIA$Month == 1] <- 'Jan'
ABIA$Month2[ABIA$Month == 2] <- 'Feb'
ABIA$Month2[ABIA$Month == 3] <- 'Mar'
ABIA$Month2[ABIA$Month == 4] <- 'Apr'
ABIA$Month2[ABIA$Month == 5] <- 'May'
ABIA$Month2[ABIA$Month == 6] <- 'Jun'
ABIA$Month2[ABIA$Month == 7] <- 'Jul'
ABIA$Month2[ABIA$Month == 8] <- 'Aug'
ABIA$Month2[ABIA$Month == 9] <- 'Sep'
ABIA$Month2[ABIA$Month == 10] <- 'Oct'
ABIA$Month2[ABIA$Month == 11] <- 'Nov'
ABIA$Month2[ABIA$Month == 12] <- 'Dec'
ABIA$Month2
ABIA$Month2 = factor(ABIA$Month2, levels = month.abb)
```

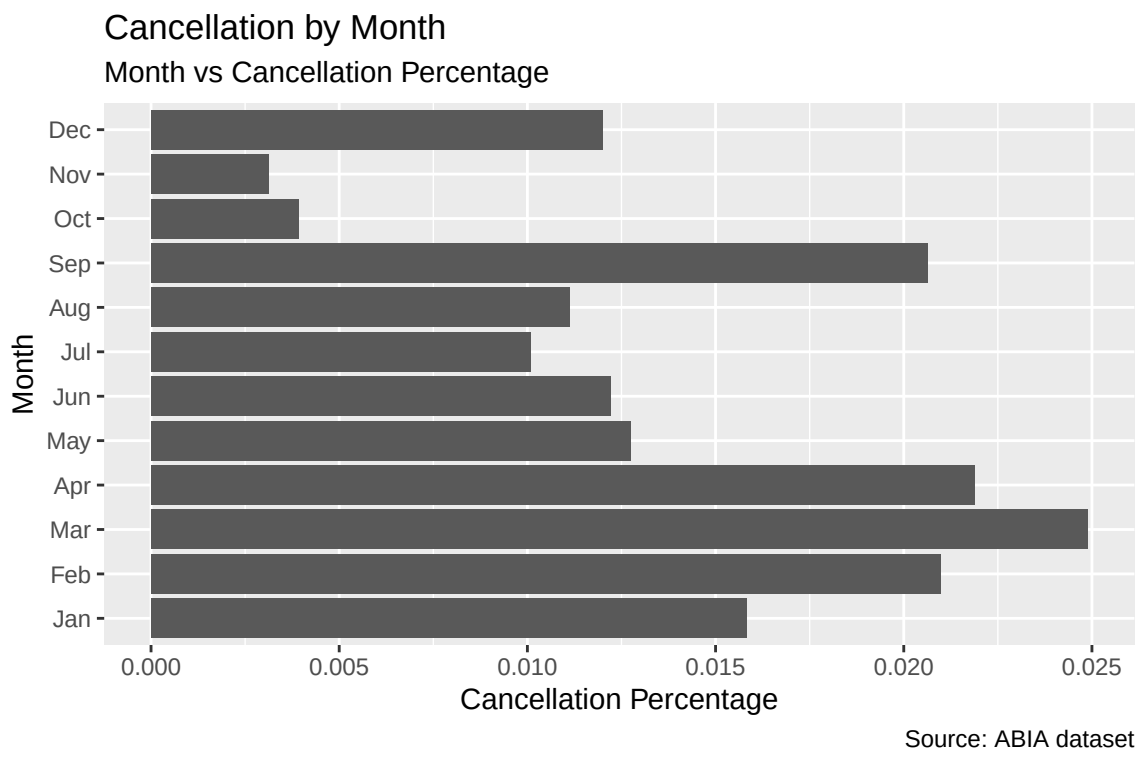
##Lets examine the impact the months have on Cancellation

```
d2 = ABIA %>%
  group_by(Month2) %>%
  summarize(canc_pctM = sum(Cancelled=='1')/n())
d2
```

```
## # A tibble: 12 x 2
##    Month2 canc_pctM
##    <fct>      <dbl>
##  1 Jan      0.0158
```

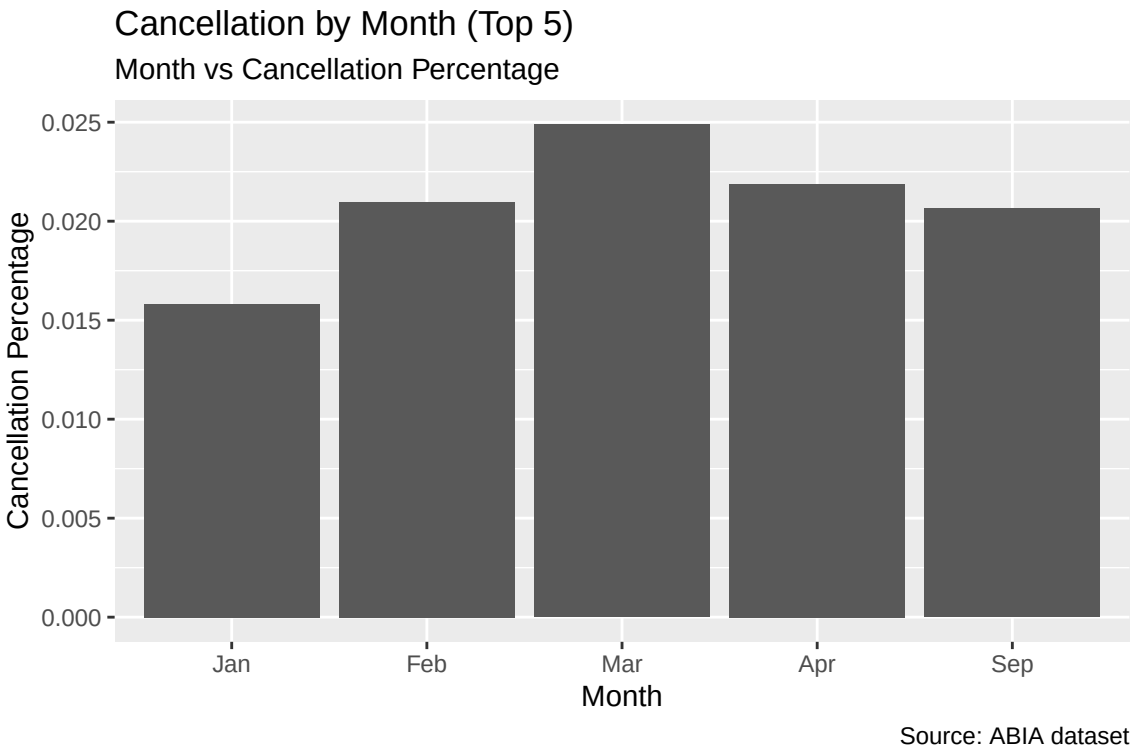
```
##  2 Feb      0.0210
##  3 Mar      0.0249
##  4 Apr      0.0219
##  5 May      0.0127
##  6 Jun      0.0122
##  7 Jul      0.0101
##  8 Aug      0.0111
##  9 Sep      0.0206
## 10 Oct      0.00391
## 11 Nov      0.00313
## 12 Dec      0.0120
```

```
ggplot(data = d2) +
  geom_bar(mapping = aes(x=Month2, y=canc_pctM), stat='identity')+
  coord_flip() +
  labs(title="Cancellation by Month",
   subtitle="Month vs Cancellation Percentage",
   caption="Source: ABIA dataset",
   x="Month",
   y ="Cancellation Percentage")
```




```
d2 %>%
  arrange(desc(canc_pctM)) %>%
  slice(1:5) %>%
  ggplot(., aes(x=Month2, y=canc_pctM))+
  geom_bar(stat='identity') +
  labs(title="Cancellation by Month (Top 5)",
   subtitle="Month vs Cancellation Percentage",
```

```
  caption="Source: ABIA dataset",
  x="Month",
  y ="Cancellation Percentage")
```



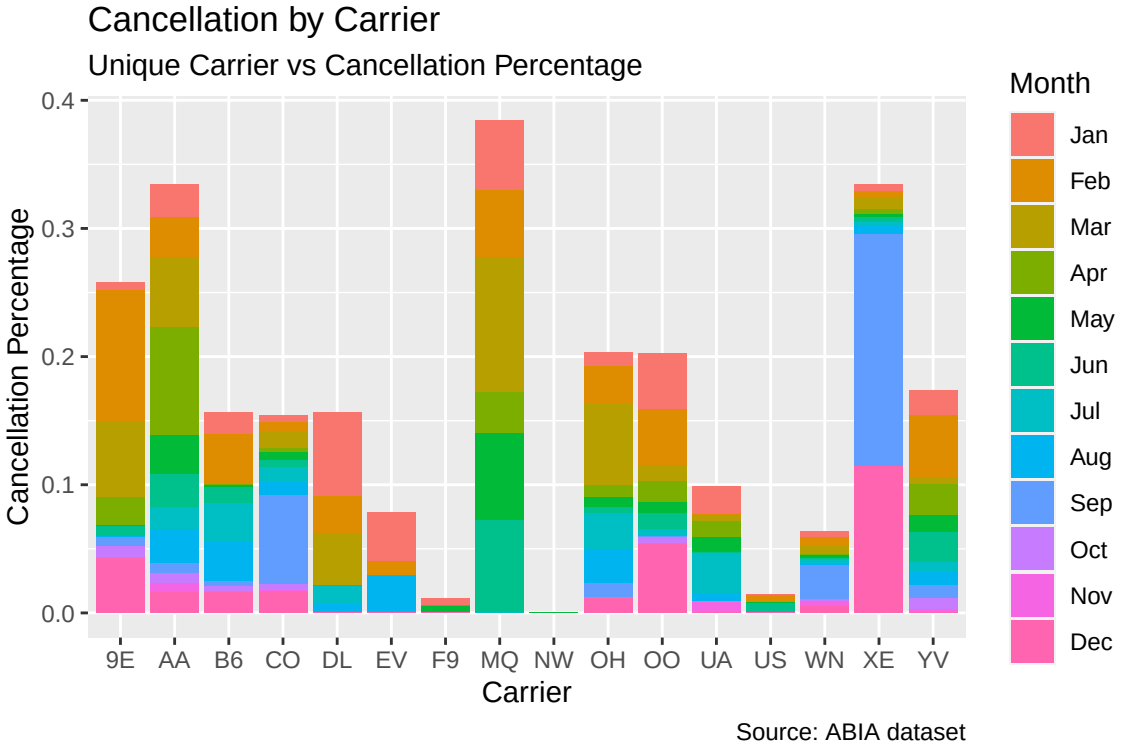
No Surprise that the Winter and Spring Months have the Highest Cancellations. Cold and Rainy...

Lets Put these in the Same Graph

```
d3 = ABIA %>%
  group_by(Month2, UniqueCarrier) %>%
  summarize(canc_pctB = sum(Cancelled=='1')/n())
d3
```

```
## # A tibble: 175 x 3
## # Groups:   Month2 [12]
##    Month2 UniqueCarrier canc_pctB
##    <fct>  <fct>             <dbl>
##  1 Jan    9E              0.00571
##  2 Jan    AA              0.0255
##  3 Jan    B6              0.0165
##  4 Jan    CO              0.00524
##  5 Jan    DL              0.0652
##  6 Jan    EV              0.0376
##  7 Jan    F9              0.00562
##  8 Jan    MQ              0.0543
##  9 Jan    NW              0
## 10 Jan    OH              0.0107
## # ... with 165 more rows
```

```
d3 %>%
  arrange(desc(canc_pctB)) %>%
  ggplot(., aes(x=UniqueCarrier, y=canc_pctB, fill= Month2))+
  geom_bar(stat='identity') +
  labs(title="Cancellation by Carrier",
       subtitle="Unique Carrier vs Cancellation Percentage",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Cancellation Percentage",
       fill = 'Month')
```



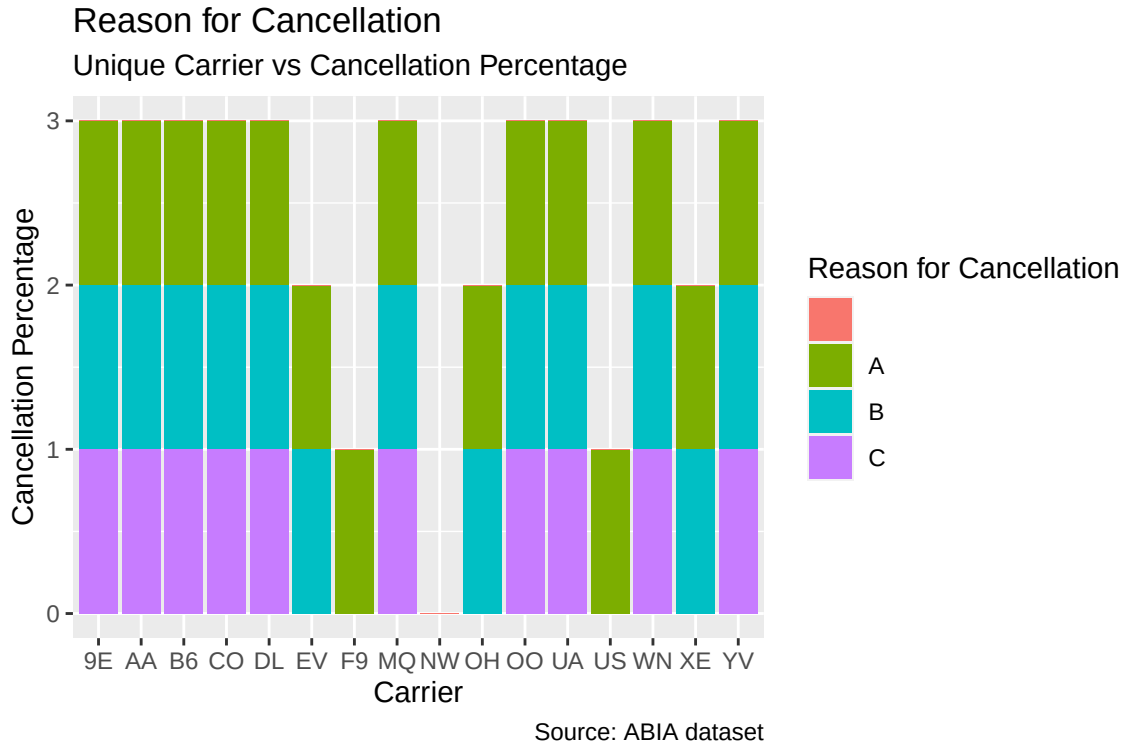
We can see Jan, Feb, and Mar have the most weight in each of the columns. Our top 5 "Unreliable" Airlines: 9E, AA, EV, MQ and OH seem to follow this trend of cancelling more in Cold Months. Interestingly, they do not have the same proportion of Cancellation for each month **9E has a very low Cancellation PCT in Jan but EV has a very high one.**

```
d10 = ABIA %>%
  group_by(CancellationCode, UniqueCarrier) %>%
  summarize(canc_pctCode = sum(Cancelled=='1')/n())
d10
```

```
## # A tibble: 54 x 3
## # Groups:   CancellationCode [4]
##    CancellationCode UniqueCarrier canc_pctCode
##    <fct>            <fct>                <dbl>
##  1 ""               9E                       0
##  2 ""               AA                       0
##  3 ""               B6                       0
```

```
##  4 ""                CO                          0
##  5 ""                DL                          0
##  6 ""                EV                          0
##  7 ""                F9                          0
##  8 ""                MQ                          0
##  9 ""                NW                          0
## 10 ""                OH                          0
## # ... with 44 more rows
```

```
d10 %>%
  arrange(desc(canc_pctCode)) %>%
  ggplot(., aes(x=UniqueCarrier, y=canc_pctCode, fill= CancellationCode))+
  geom_bar(stat='identity') +
  labs(title="Reason for Cancellation",
       subtitle="Unique Carrier vs Cancellation Percentage",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Cancellation Percentage",
       fill = 'Reason for Cancellation')
```



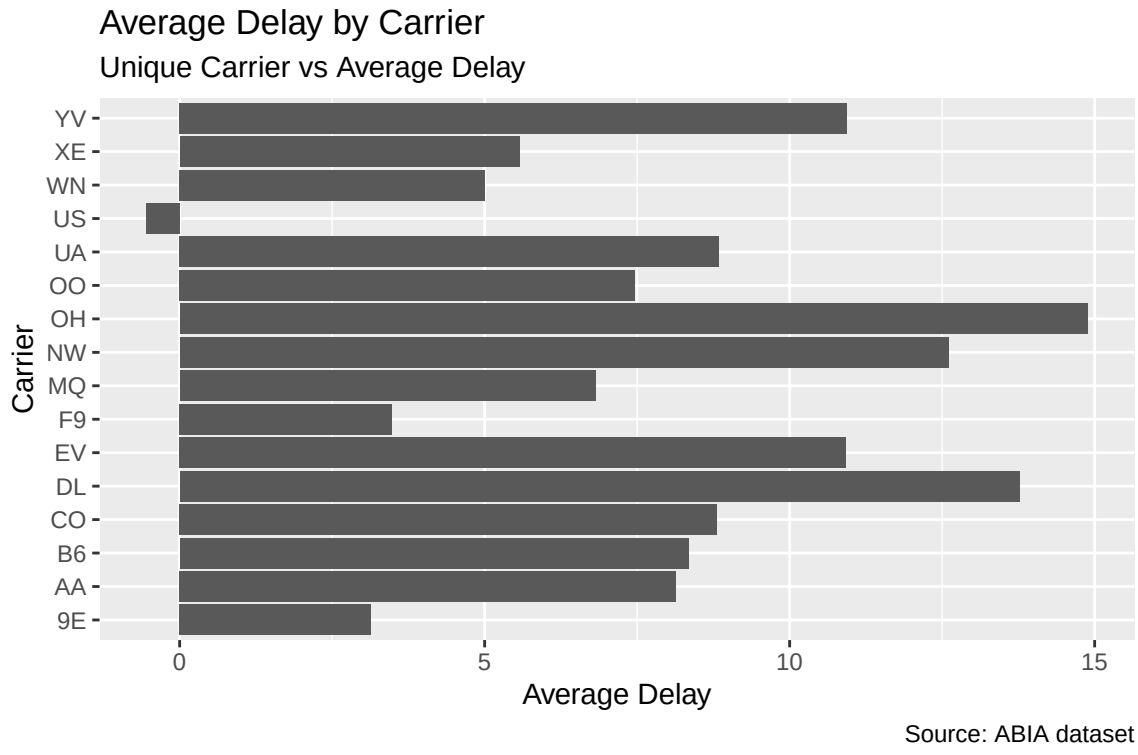
A is Carrier B is Weather C is NAS (type of weather delays that could be reduced with corrective action by the airports or the Federal Aviation Administration) Pretty even Split for most airlines

##Now Which Airlines have the most Delay??

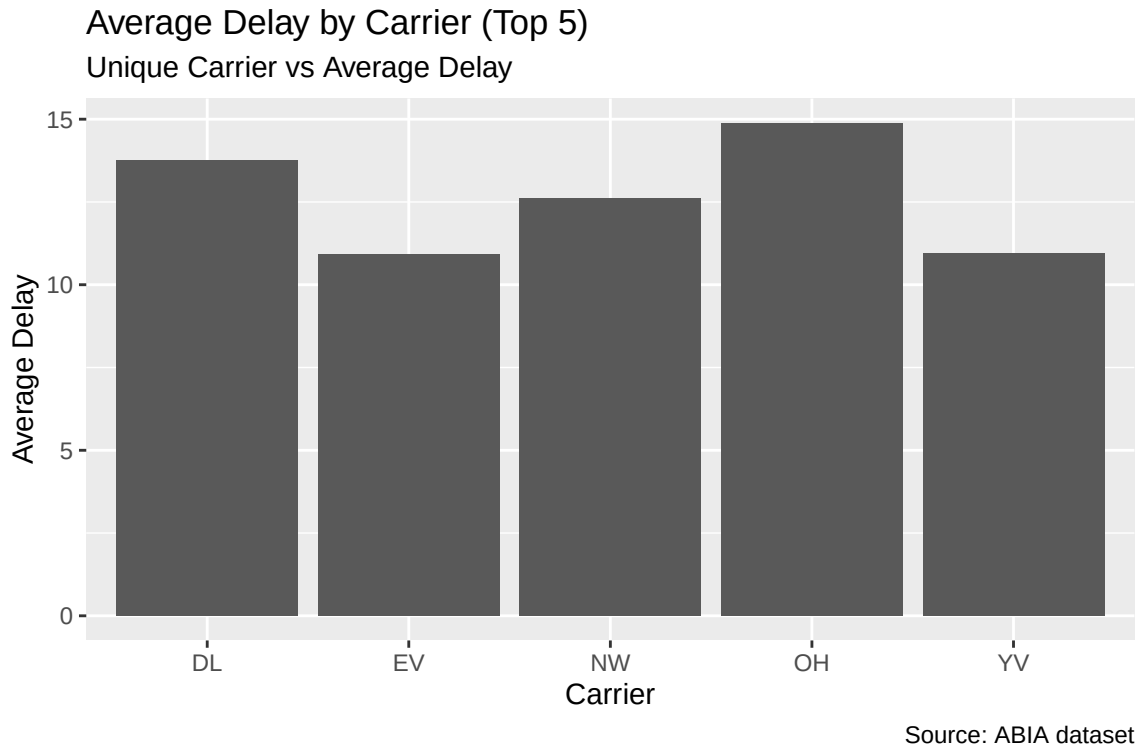
```
d4 = ABIA %>%
  group_by(UniqueCarrier) %>%
  summarise(AVGDelay = mean(ArrDelay, na.rm = T))
d4
```

```
## # A tibble: 16 x 2
##    UniqueCarrier AVGDelay
##    <fct>            <dbl>
##  1 9E               3.14
##  2 AA               8.14
##  3 B6               8.34
##  4 CO               8.81
##  5 DL              13.8
##  6 EV              10.9
##  7 F9               3.48
##  8 MQ               6.82
##  9 NW              12.6
## 10 OH              14.9
## 11 OO               7.46
## 12 UA               8.84
## 13 US              -0.542
## 14 WN               5.00
## 15 XE               5.57
## 16 YV              10.9
```

```
ggplot(data = d4) +
  geom_bar(mapping = aes(x=UniqueCarrier, y=AVGDelay), stat='identity')+
  coord_flip() +
  labs(title="Average Delay by Carrier",
       subtitle="Unique Carrier vs Average Delay",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Average Delay")
```



```
d4 %>%
  arrange(desc(AVGDelay)) %>%
  slice(1:5) %>%
  ggplot(., aes(x=UniqueCarrier, y=AVGDelay))+
  geom_bar(stat='identity') +
  labs(title="Average Delay by Carrier (Top 5)",
       subtitle="Unique Carrier vs Average Delay",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Average Delay")
```



We see EV and OH show up again...

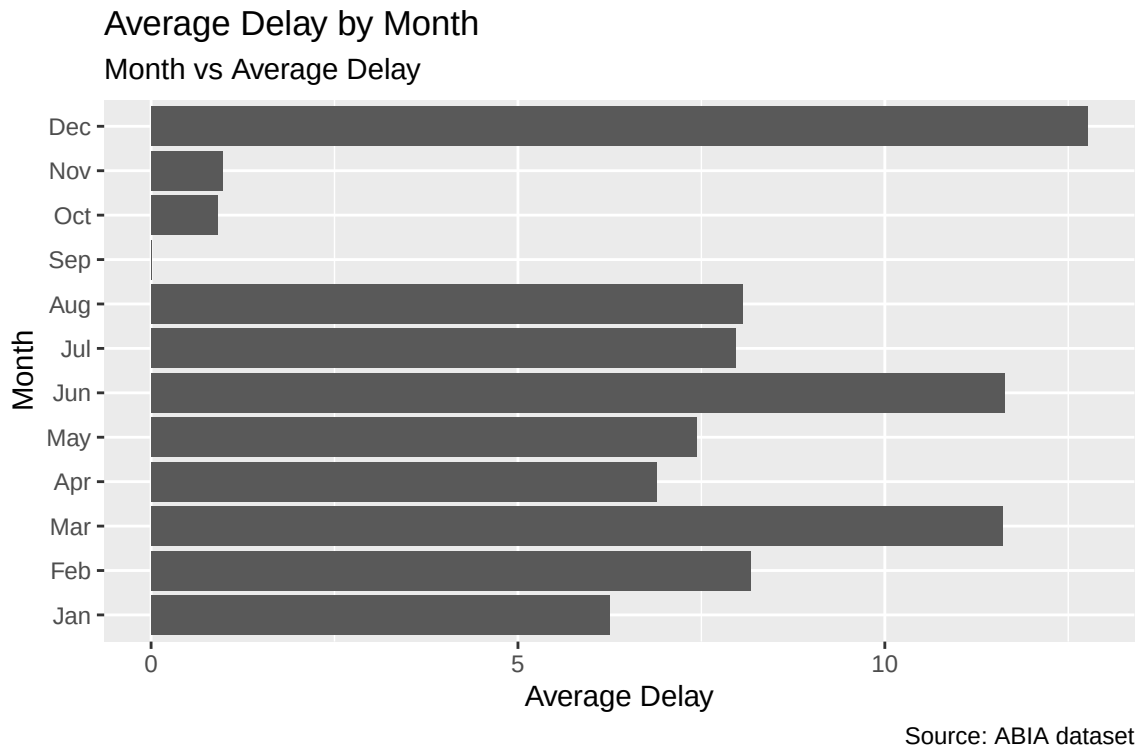
##Do months have an effect on delay too???

```
d5 = ABIA %>%
  group_by(Month2) %>%
  summarise(AVGDelay = mean(ArrDelay, na.rm = T))
d5
```

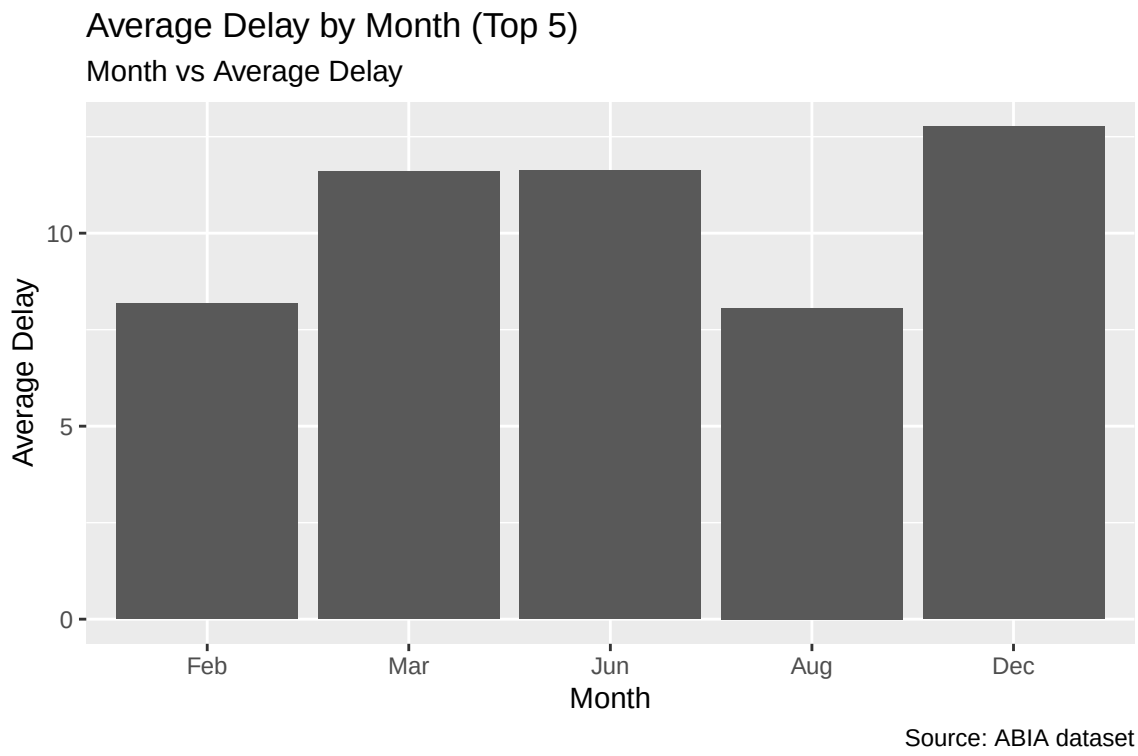
```
## # A tibble: 12 x 2
##    Month2  AVGDelay
##    <fct>      <dbl>
##  1 Jan     6.25
##  2 Feb     8.17
##  3 Mar    11.6
##  4 Apr     6.89
##  5 May     7.43
##  6 Jun    11.6
```

```
##  7 Jul     7.97
##  8 Aug     8.06
##  9 Sep    -0.000685
## 10 Oct     0.907
## 11 Nov     0.969
## 12 Dec    12.8
```

```
ggplot(data = d5) +
  geom_bar(mapping = aes(x=Month2, y=AVGDelay), stat='identity')+
  coord_flip()+
  labs(title="Average Delay by Month",
       subtitle="Month vs Average Delay",
       caption="Source: ABIA dataset",
       x="Month",
       y ="Average Delay")
```




```
d5 %>%
  arrange(desc(AVGDelay)) %>%
  slice(1:5) %>%
  ggplot(., aes(x=Month2, y=AVGDelay))+
  geom_bar(stat='identity')+
  labs(title="Average Delay by Month (Top 5)",
       subtitle="Month vs Average Delay",
       caption="Source: ABIA dataset",
       x="Month",
       y ="Average Delay")
```



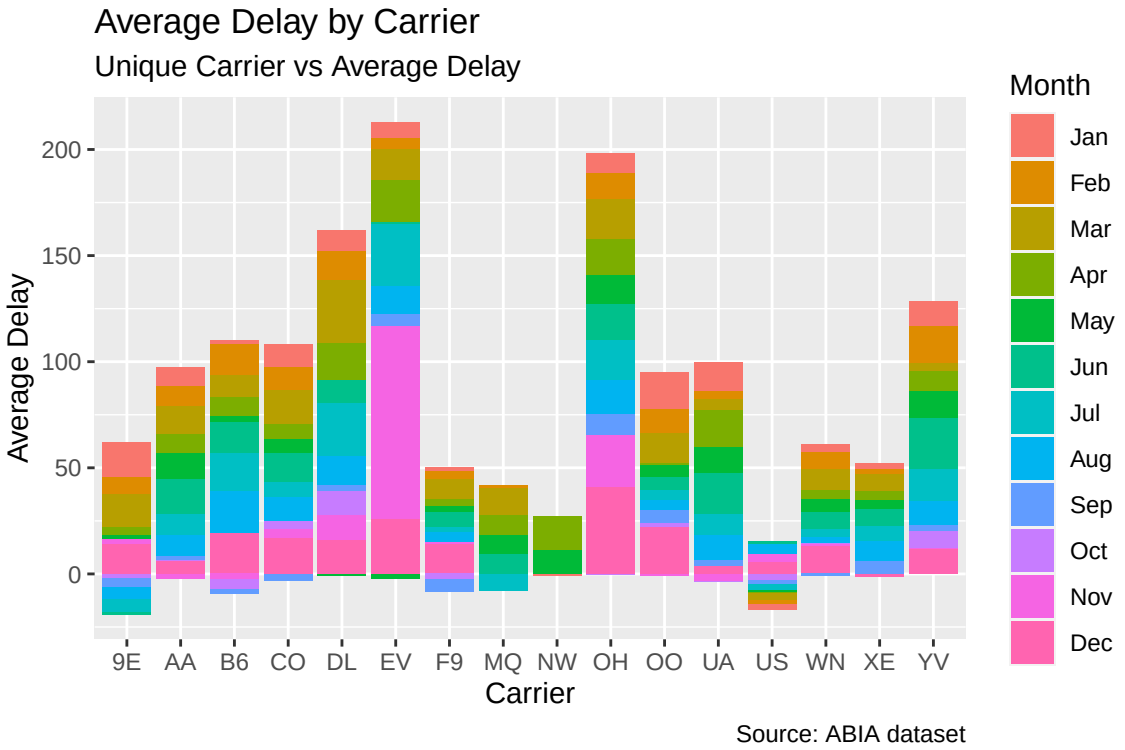
We can see a different set of months that have the most delay. However, Feb and March show up again. Were EV and OH victims of these months?

```
d6 = ABIA %>%
  group_by(UniqueCarrier, Month2) %>%
  summarise(AVGDelay2 = mean(ArrDelay, na.rm = T))
d6
```

```
## # A tibble: 175 x 3
## # Groups:   UniqueCarrier [16]
##    UniqueCarrier Month2 AVGDelay2
##    <fct>         <fct>      <dbl>
##  1 9E            Jan        16.5
##  2 9E            Feb         7.99
##  3 9E            Mar        15.3
##  4 9E            Apr         4.02
##  5 9E            May         1.69
##  6 9E            Jun        -1.46
##  7 9E            Jul        -5.93
##  8 9E            Aug        -5.57
##  9 9E            Sep        -4.17
## 10 9E            Oct        -2.06
## # ... with 165 more rows
```

```
d6 %>%
  arrange(desc(AVGDelay2)) %>%
  ggplot(., aes(x=UniqueCarrier, y=AVGDelay2, fill= Month2))+
  geom_bar(stat='identity')+
```

```
  labs(title="Average Delay by Carrier",
       subtitle="Unique Carrier vs Average Delay",
       caption="Source: ABIA dataset",
       x="Carrier",
       y ="Average Delay",
       fill = 'Month')
```



It seems something went wrong for EV in November. OH had a bad November and December which is better since we predicted December was a bad month for all airlines.

Conclusion so far: Don't Fly EV (ExpressJet)

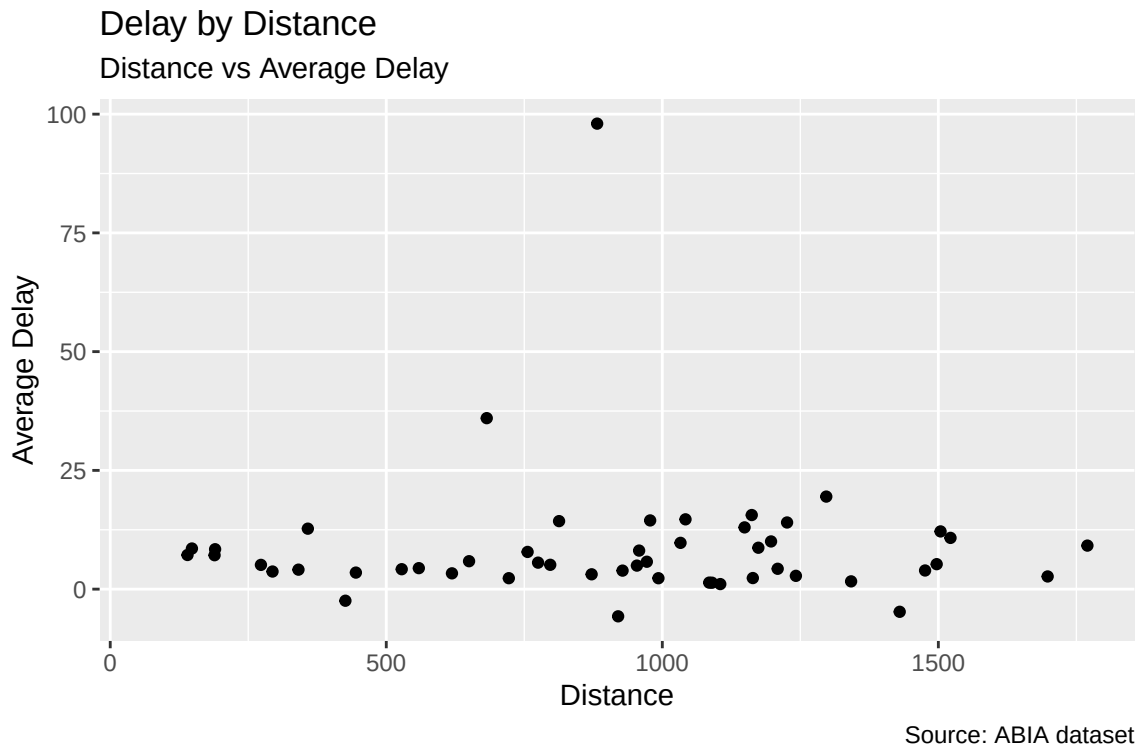
##Could other Factors influence Delay

```
d7 = ABIA %>%
  group_by(Distance) %>%
  summarise(AVGDelay = mean(ArrDelay, na.rm = T))
d7
```

```
## # A tibble: 53 x 2
##    Distance AVGDelay
##       <int>    <dbl>
##  1       66   NaN
##  2      140     7.19
##  3      148     8.55
##  4      189     7.15
##  5      190     8.40
##  6      273     5.11
##  7      294     3.71
```

```
##  8      341     4.10
##  9      358    12.7
## 10      426    -2.44
## # ... with 43 more rows
```

```
ggplot(data = d7) +
  geom_point(mapping = aes(x = Distance, y = AVGDelay))+
  labs(title="Delay by Distance",
       subtitle="Distance vs Average Delay",
       caption="Source: ABIA dataset",
       x="Distance",
       y ="Average Delay")
```



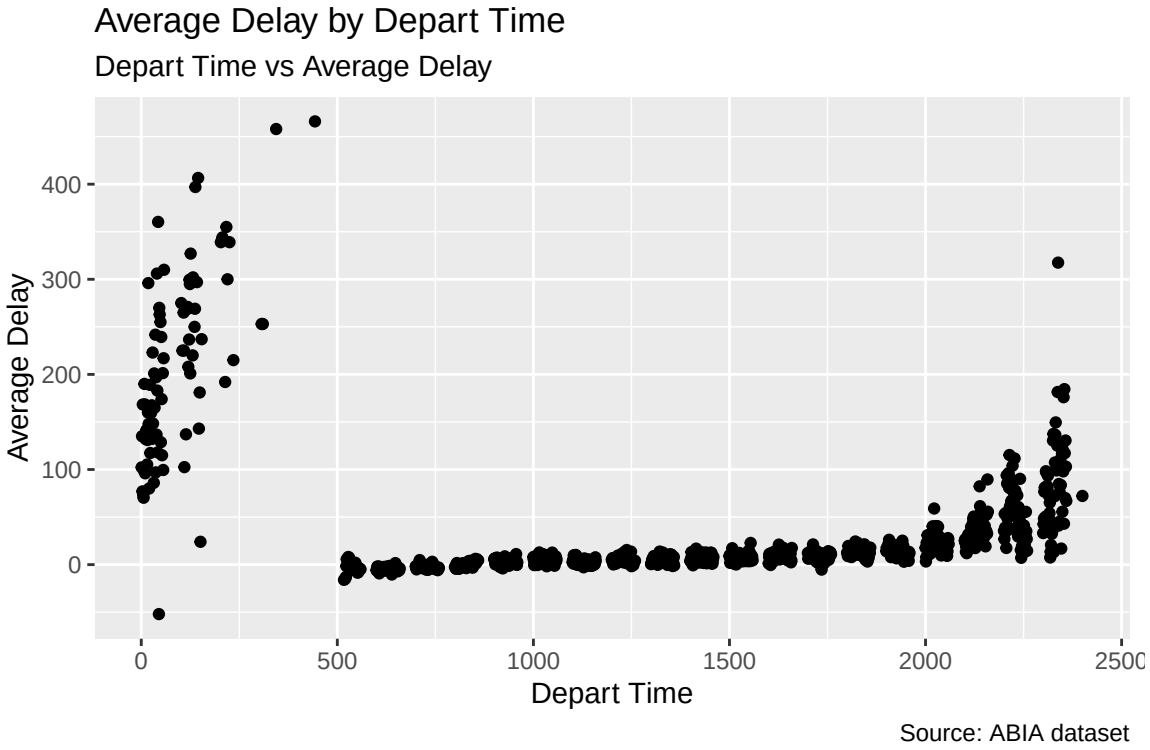
Distance doesn't have an impact.

```
d8 = ABIA %>%
  group_by(DepTime) %>%
  summarise(AVGDelay = mean(ArrDelay, na.rm = T))
d8
```

```
## # A tibble: 1,209 x 2
##    DepTime AVGDelay
##      <int>    <dbl>
##  1       1    102.
##  2       2    135
##  3       3     77
##  4       4    168.
##  5       5     73.5
```

```
##  6       6     70.3
##  7       7     98.7
##  8       8    190
##  9       9    168.
## 10      10     96
## # ... with 1,199 more rows
```

```
ggplot(data = d8) +
  geom_point(mapping = aes(x = DepTime, y = AVGDelay))+
  labs(title="Average Delay by Depart Time",
       subtitle="Depart Time vs Average Delay",
       caption="Source: ABIA dataset",
       x="Depart Time",
       y ="Average Delay")
```



Depart time does have an impact on Delay, however this can be explained. If a flight experiences a Delay, it is more likely that it will depart at a later more unconventional time.

Sorry ExpressJet, you're out of luck!

Conclusion: Don't Fly ExpressJet

Portfolio Modeling

First we will go ahead and set up the required libraries

```
library(mosaic)
library(quantmod)
library(foreach)
```

Scenario 1 : Here we are going ahead with Marketing based growth equites

Portfolios that we will go ahead with are:

1) PXH : Invesco FTSE RAFI Emerging Markets ETF

2) JPEM : JPMorgan Diversified Return Emerging Markets Equity ETF

3) ECON : Columbia Emerging Markets Consumer ETF

4) EDIV : SPDR S&P Emerging Markets Dividend ETF

5) DEM : WisdomTree Emerging Markets Equity Income Fund

```
mystocks = c("PXH", "ECON", "DGRE", "EDIV", "DEM")
getSymbols(mystocks, from = "2015-01-01")
```

```
## [1] "PXH"  "ECON" "DGRE" "EDIV" "DEM"
```

```
options("getSymbols.warning4.0"=FALSE)
```

We would need to adjust for splits and dividends

```
options(warn=-1)
PXHa = adjustOHLC(PXH)
```

```
DGREa = adjustOHLC(DGRE)
```

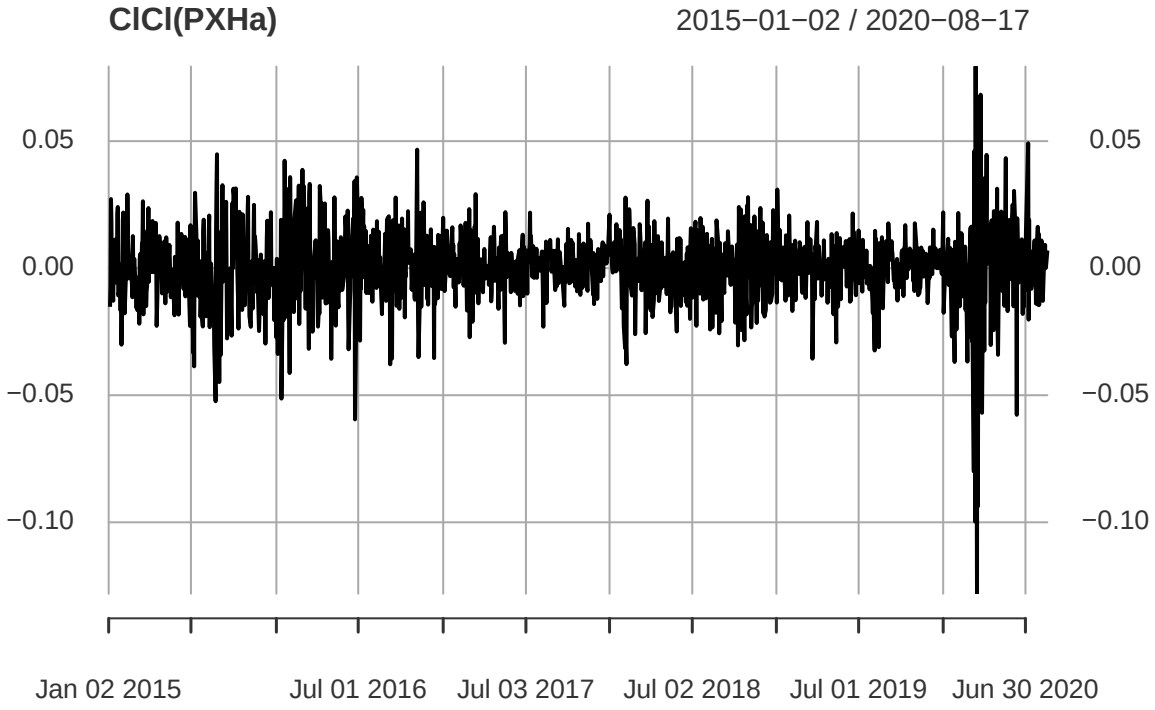
```
ECONa = adjustOHLC(ECON)
```

```
EDIVa = adjustOHLC(EDIV)
```

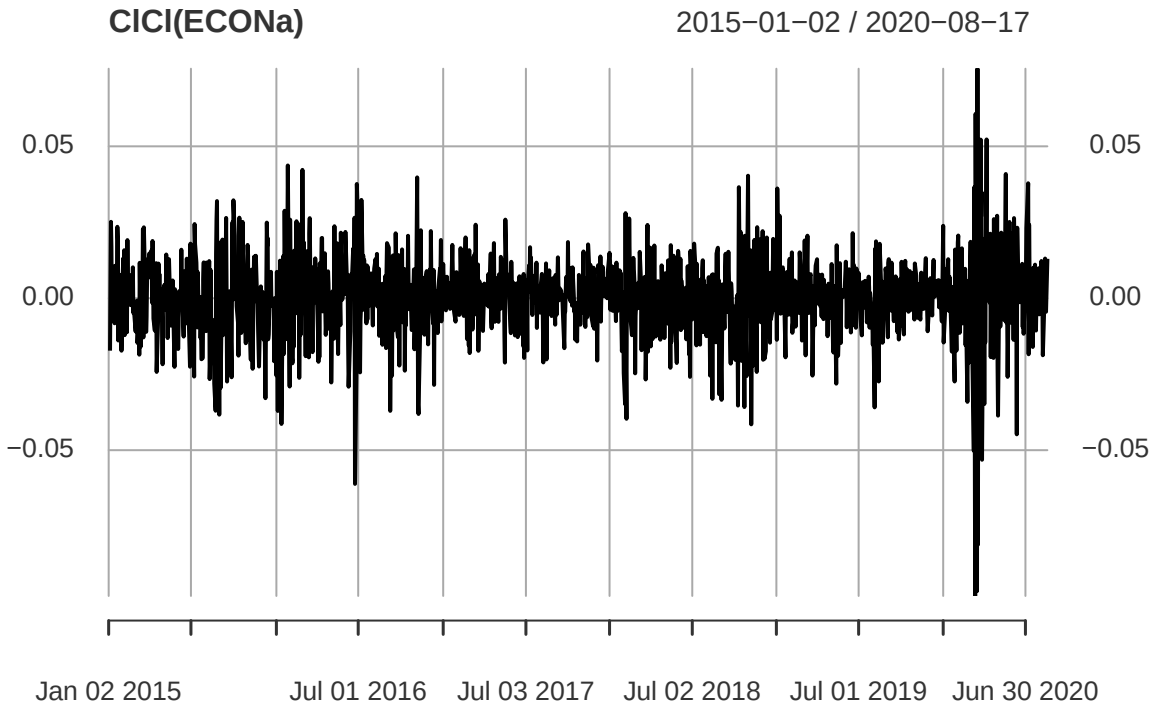
```
DEMa = adjustOHLC(DEM)
```

After adjusting for splits and dividends, we look up for close to close changes on a daily level for the selected stocks

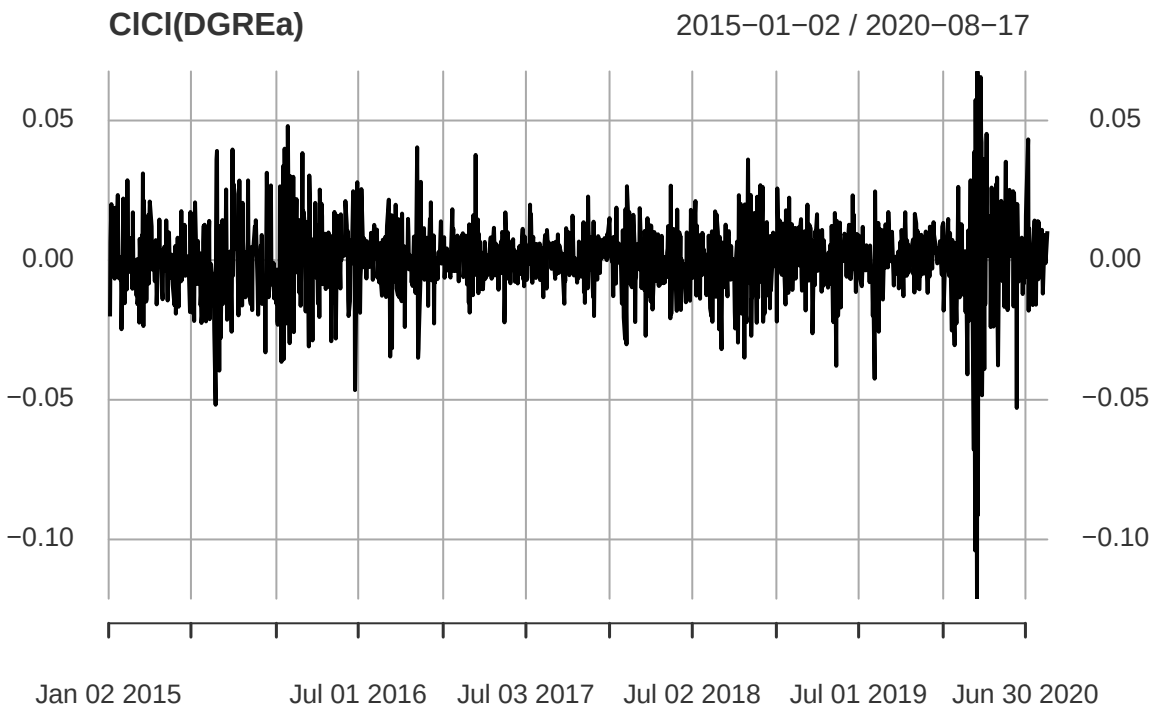
```
plot(ClCl(PXHa))
```



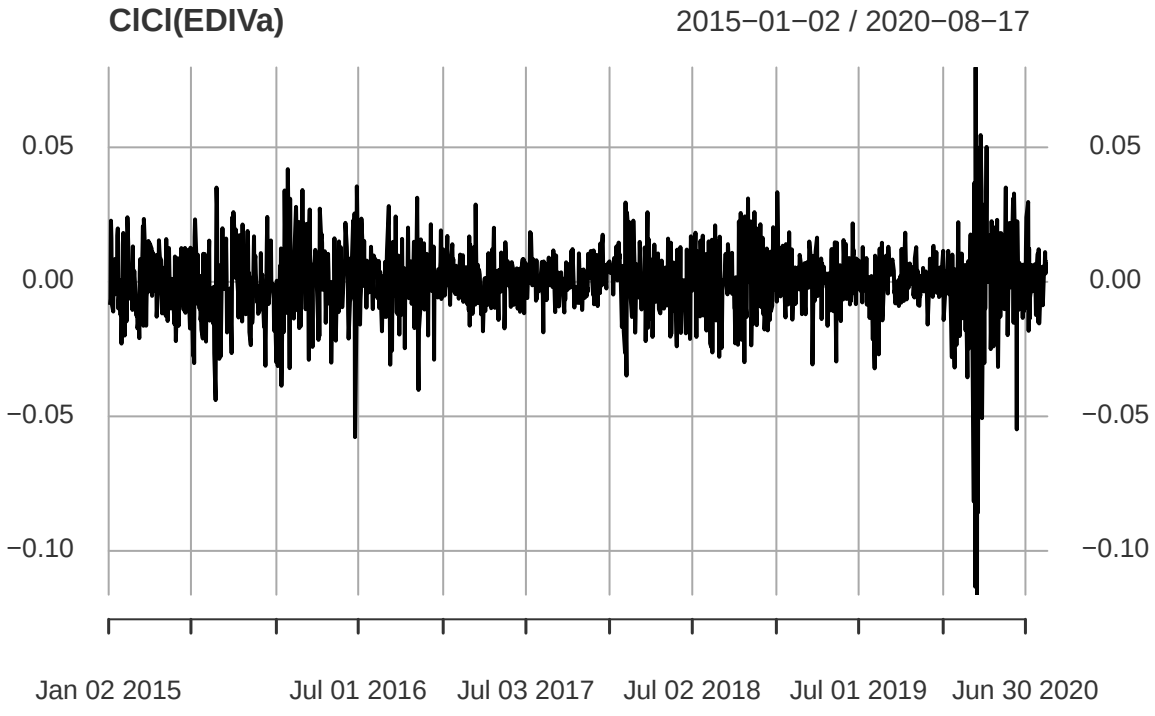

```
plot(ClCl(ECONa))
```



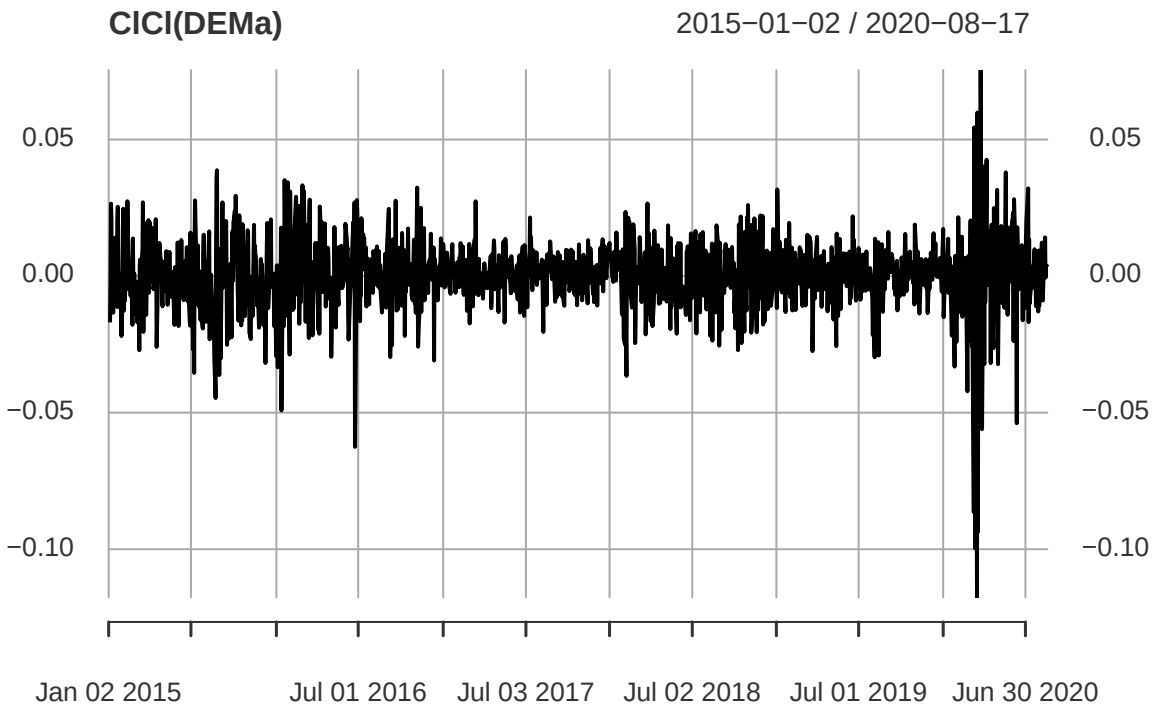
```
plot(ClCl(DGREa))
```




```
plot(ClCl(EDIVa))
```



```
plot(ClCl(DEMa))
```



Now we combine all the data into a single dataframe and remove the NA values

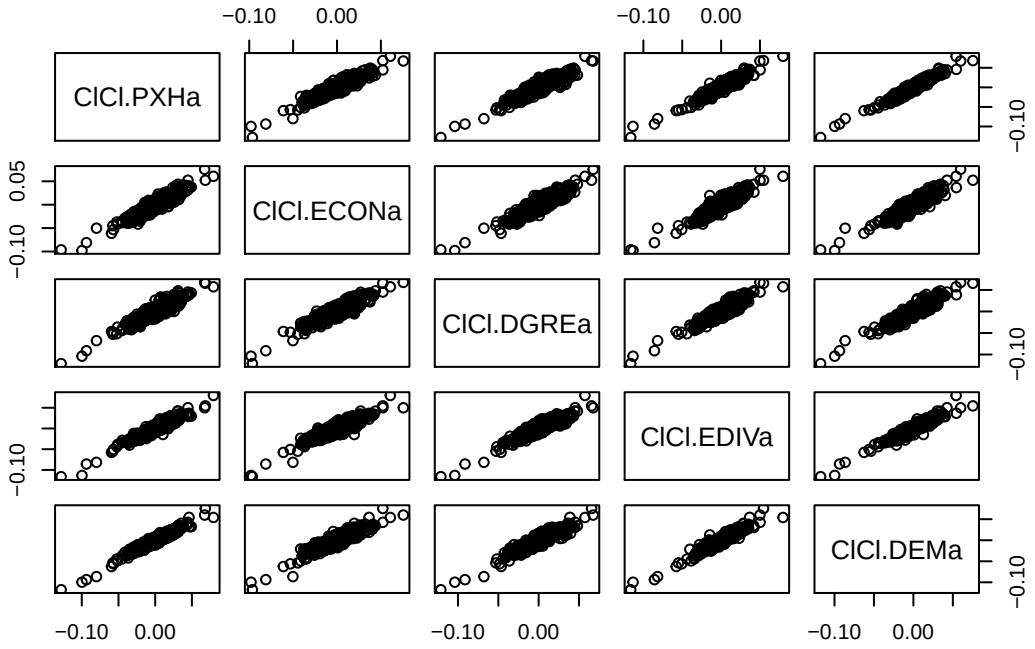
```
all_returns = cbind(ClCl(PXHa),ClCl(ECONa),ClCl(DGREa), ClCl(EDIVa), ClCl(DEMa))
head(all_returns)
```

```
##               ClCl.PXHa   ClCl.ECONa   ClCl.DGREa   ClCl.EDIVa    ClCl.DEMa
## 2015-01-02           NA           NA           NA           NA           NA
## 2015-01-05 -0.015209180 -0.017248295 -0.020142677 -0.008549440 -0.016961275
## 2015-01-06 -0.006618809  0.006122449  0.006509679 -0.006244514 -0.001701093
## 2015-01-07  0.027207107  0.025152130  0.013615862  0.022740935  0.026533494
## 2015-01-08  0.018918919  0.017016264  0.020065401  0.016091281  0.020630852
## 2015-01-09 -0.007427109 -0.005836653 -0.004938231 -0.008350158 -0.010687709
```

```
all_returns = as.matrix(na.omit(all_returns))
N = nrow(all_returns)
```

As the next step we check if there is a correlation amongst all the stocks

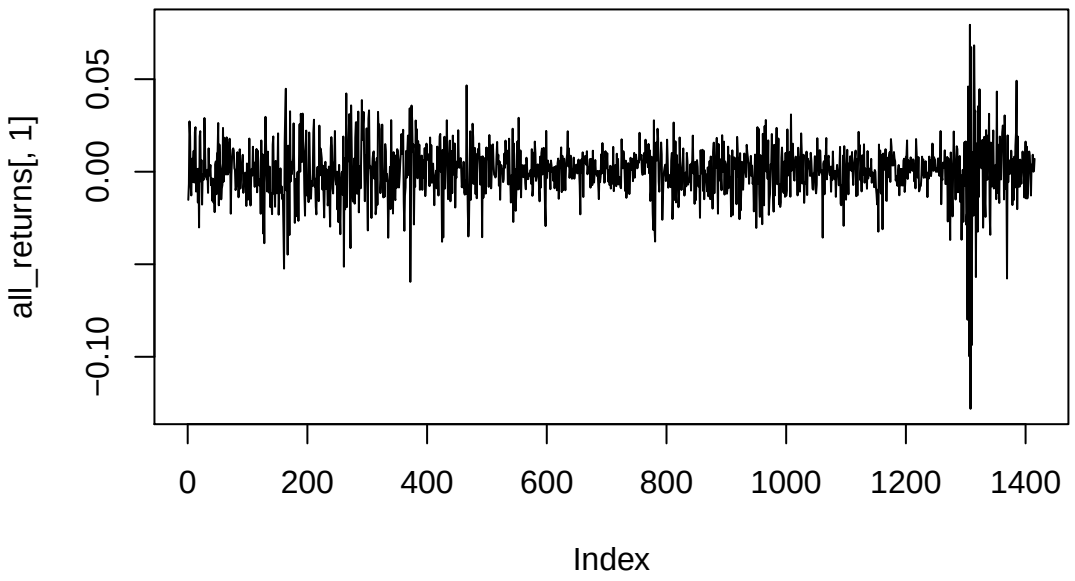
```
pairs(all_returns)
```



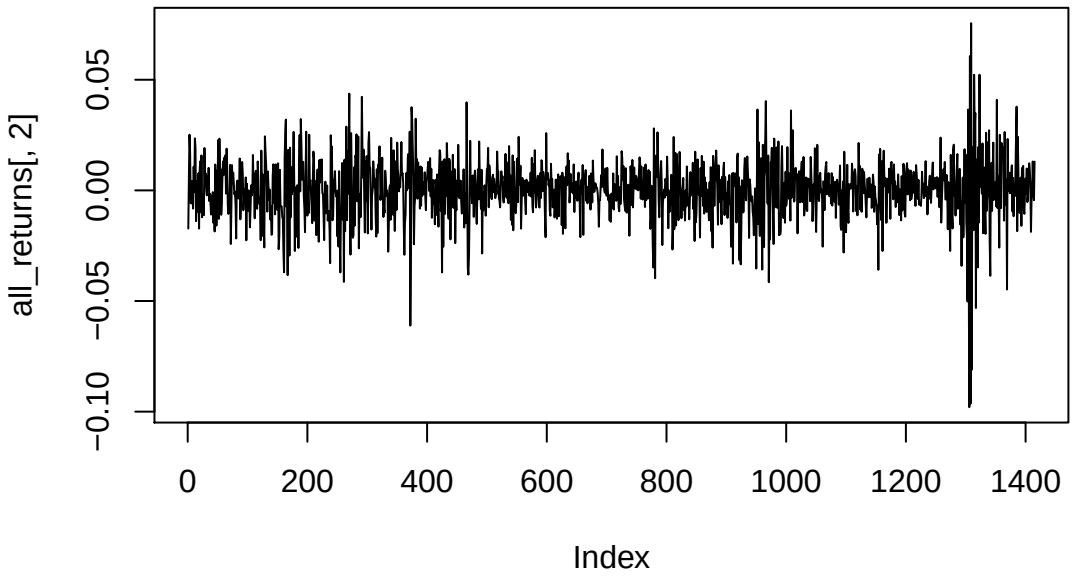
The correlation plot seems to suggest that there seems to be some sort of correlation amongst the stocks

Now we plot the market returns over time

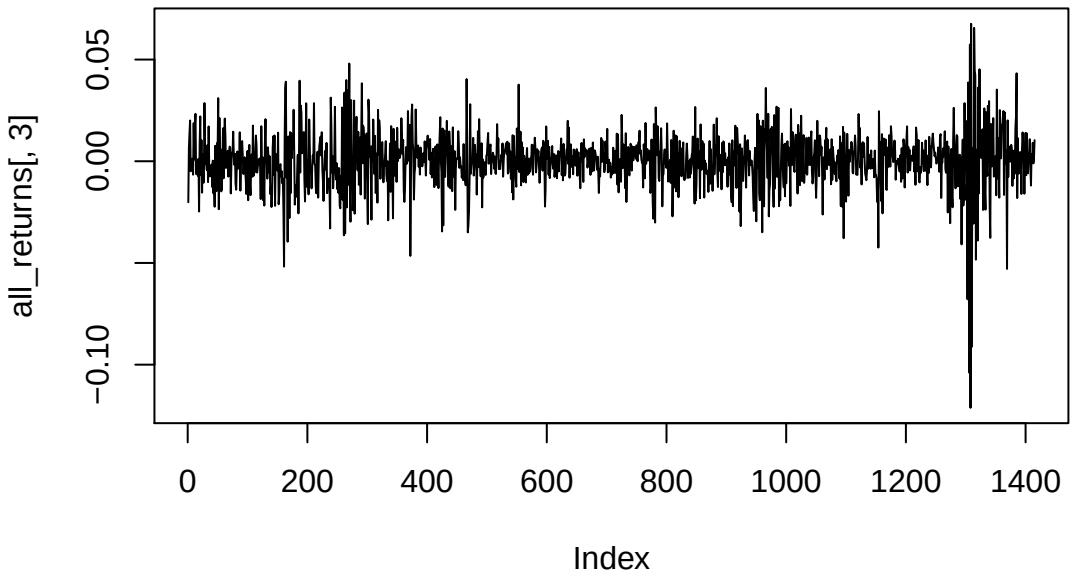
```
plot(all_returns[,1], type='l')
```



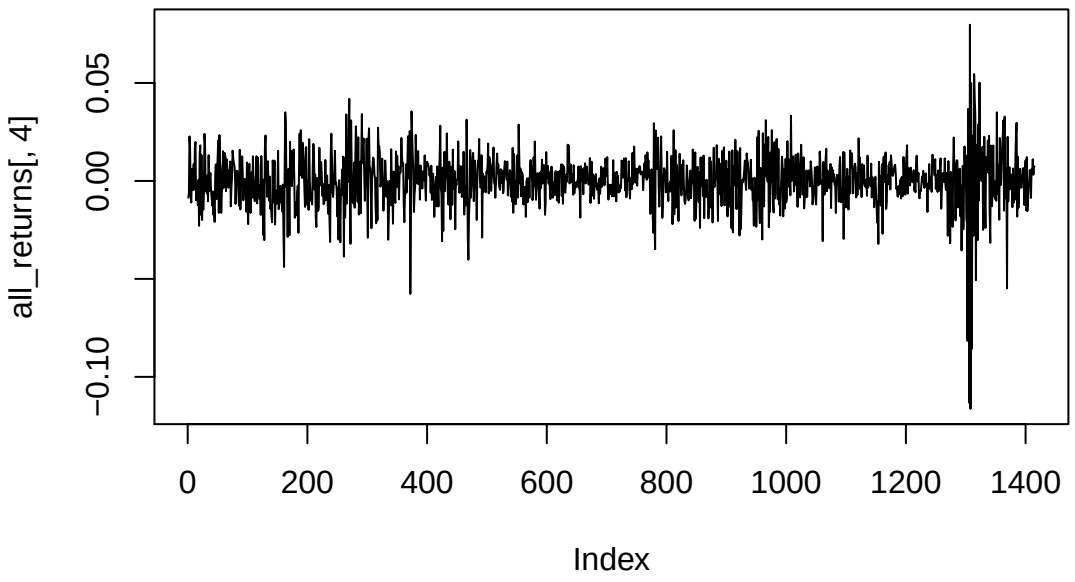

```
plot(all_returns[,2], type='l')
```



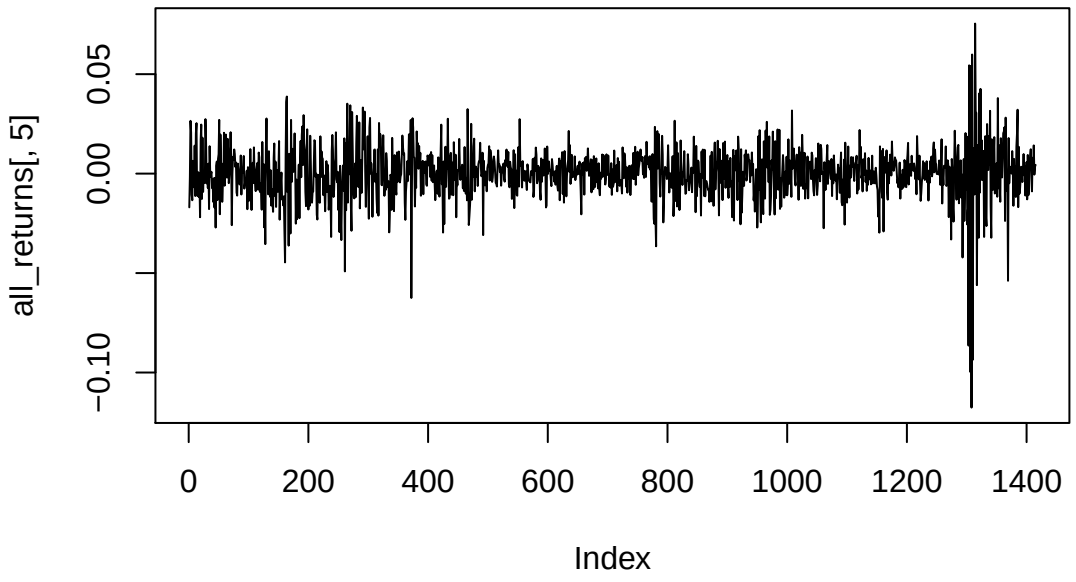
```
plot(all_returns[,3], type='l')
```




```
plot(all_returns[,4], type='l')
```



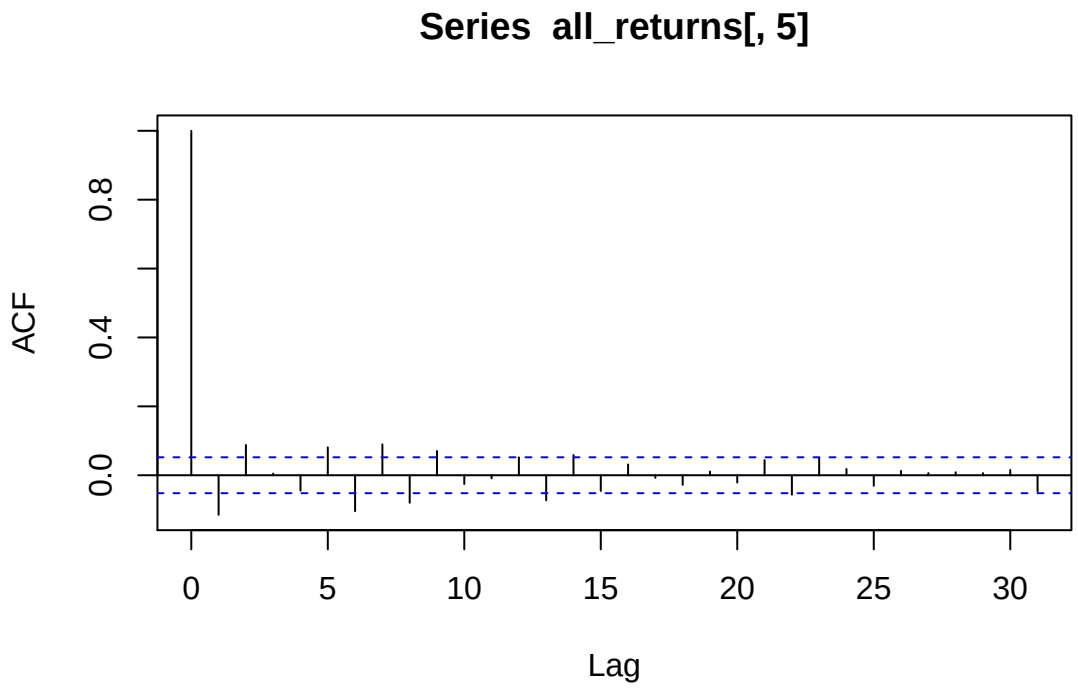
```
plot(all_returns[,5], type='l')
```



The dip in the market returns near the Mar 2020 period can be attributed to the ongoing Covid situation

Now we look at the auto correlation plot

```
acf(all_returns[,5])
```



As it turns out there is not much correlation there!

Now we simulate for a single day return from our previous historical values

```
return.today = resample(all_returns, 1, orig.ids=FALSE)
```

After getting a single day simulation we can go ahead with building our very own portfolio

```
total_wealth = 100000
my_weights = c(0.2,0.2,0.2, 0.2, 0.2)
holdings = total_wealth*my_weights
holdings = holdings*(1 + return.today)
```

In the above code we take the total wealth as $100000 and distribute it equally among all the stocks and also get the new combination of holding based on our returns from the single day simulation

After getting the holdings we calculate the new total wealth

```
holdings
```

```
##            ClCl.PXHa ClCl.ECONa ClCl.DGREa ClCl.EDIVa ClCl.DEMa
## 2018-11-02     20000   20177.49   20212.67   20040.54  20057.98
```

```
total_wealth = sum(holdings)
total_wealth
```

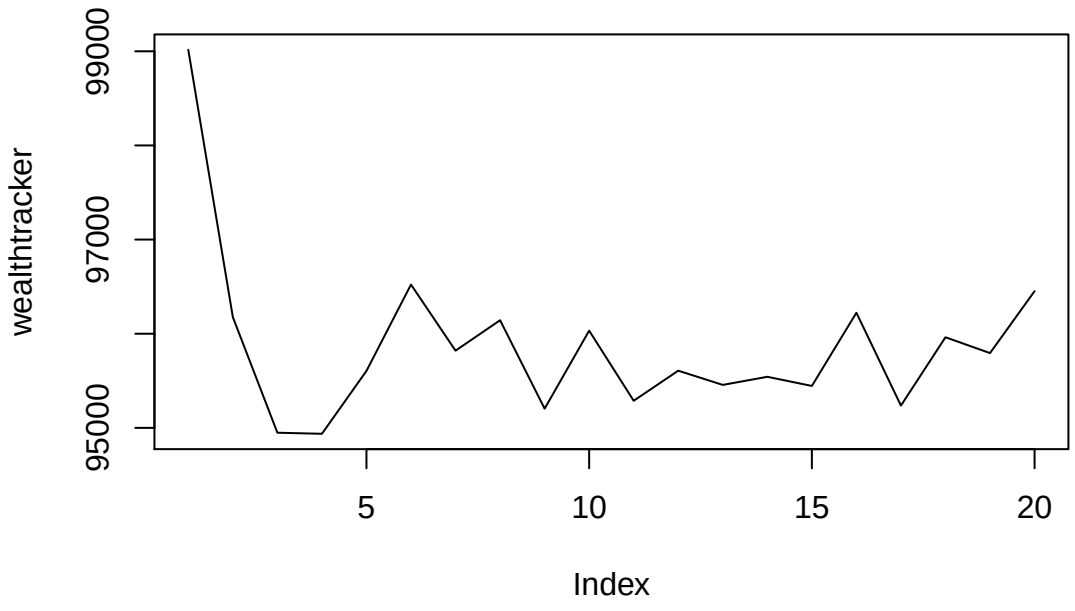
```
## [1] 100488.7
```

We can repeat the above process for 4 trading weeks and try to see what it gives

```
total_wealth = 100000
weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
holdings = weights * total_wealth
n_days = 20  # capital T in the notes
wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
for(today in 1:n_days) {
  return.today = resample(all_returns, 1, orig.ids=FALSE)
  holdings = holdings + holdings*return.today
  total_wealth = sum(holdings)
  wealthtracker[today] = total_wealth
}
total_wealth
```

```
## [1] 96452.07
```

```
plot(wealthtracker, type='l')
```



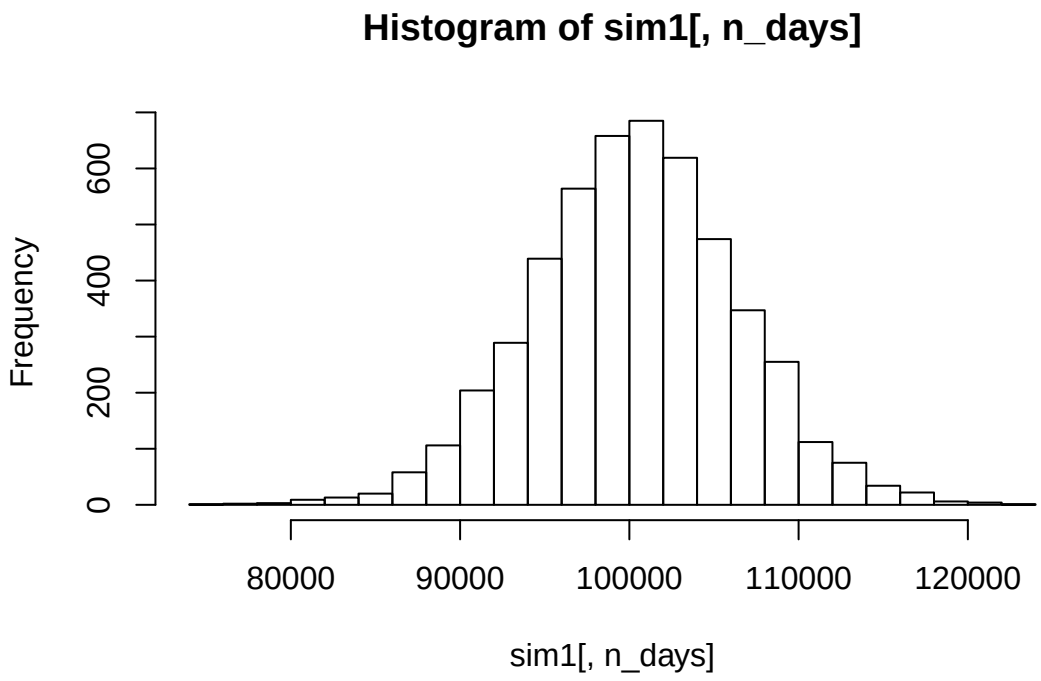
Now we simulate the given code for many times and see what is the frequency of the returns

```
initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
  holdings = weights * total_wealth
  n_days = 20
  wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
  }
  wealthtracker
}
```

```
head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 100241.38 100543.77 102006.65 101151.13  99738.26 101793.45 101527.40
## result.2 100282.63  99621.98  97811.37  98610.15  97247.67  97133.85  95339.02
## result.3 100177.82 100126.45 101382.20 102269.39 101851.00 101558.17 100538.01
## result.4 100850.48 100367.85  99949.99  89660.73  90256.08  89964.74  90168.76
## result.5 100296.59  99913.05 102308.36 103584.73 104944.95 108256.97 109149.27
## result.6  98742.71  99973.63  99354.41  98548.11 100131.53 100394.91  99736.88
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 100531.96  99777.34  99271.55 100202.06  98034.46  98416.15  97815.26
## result.2  94090.46  94503.49  93761.58  94807.82  94762.49  93992.85  94897.17
## result.3 100034.93  99331.21 100494.94 101369.13 108076.21 107897.11 107181.96
## result.4  89928.56  90414.47  89726.71  90296.66  92317.95  93311.45  94482.69
## result.5 109743.90 108820.10 111263.68 109703.79 110161.45 115333.92 115007.73
## result.6  99362.32  97401.56  97759.89  98558.35  99162.47  97161.79  96803.04
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  97525.53  97973.64  97675.55  97156.55  96859.74  96983.05
## result.2  94414.64  94469.50  95459.56  95477.67  94558.66  95362.61
## result.3 106830.89 106438.13 107045.45 107070.46 107339.28 106886.77
## result.4  95092.55  97265.12  97027.43  96433.53  96817.16  97829.37
## result.5 111569.31 112748.50 113310.45 115304.74 116332.65 116033.39
## result.6  96310.38  96726.61  96760.11  96862.41  97335.54  97952.76
```

```
hist(sim1[,n_days], 25)
```



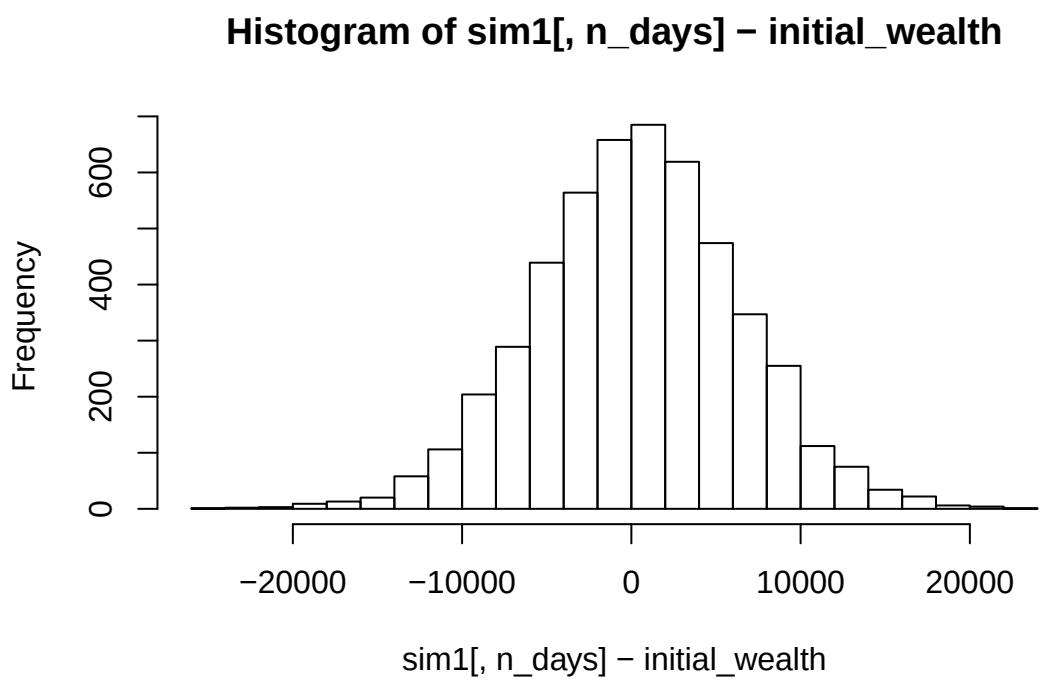

```r
mean(sim1[,n_days])
```

```
## [1] 100370
```

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

```
## [1] 370.0021
```

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




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

```
##        5%
## -9515.239
```

The 5% VAR at risk for this portfolio comes out as above

Scenario 2: Here we are considering the currency ETF's

Currency ETFs offer investors exposure to a single currency or a basket of currencies. The funds are comprised of currency futures contracts.

UUP : Invesco DB US Dollar Index Bullish Fund

FXE Invesco CurrencyShares Euro Currency Trust

FXY Invesco CurrencyShares Japanese Yen Trust

```
mystocks = c("UUP","FXE","FXY")
getSymbols(mystocks, from = "2015-01-01")
```

```
## [1] "UUP" "FXE" "FXY"
```

```
options("getSymbols.warning4.0"=FALSE)
```

Adjusting for splits and dividends

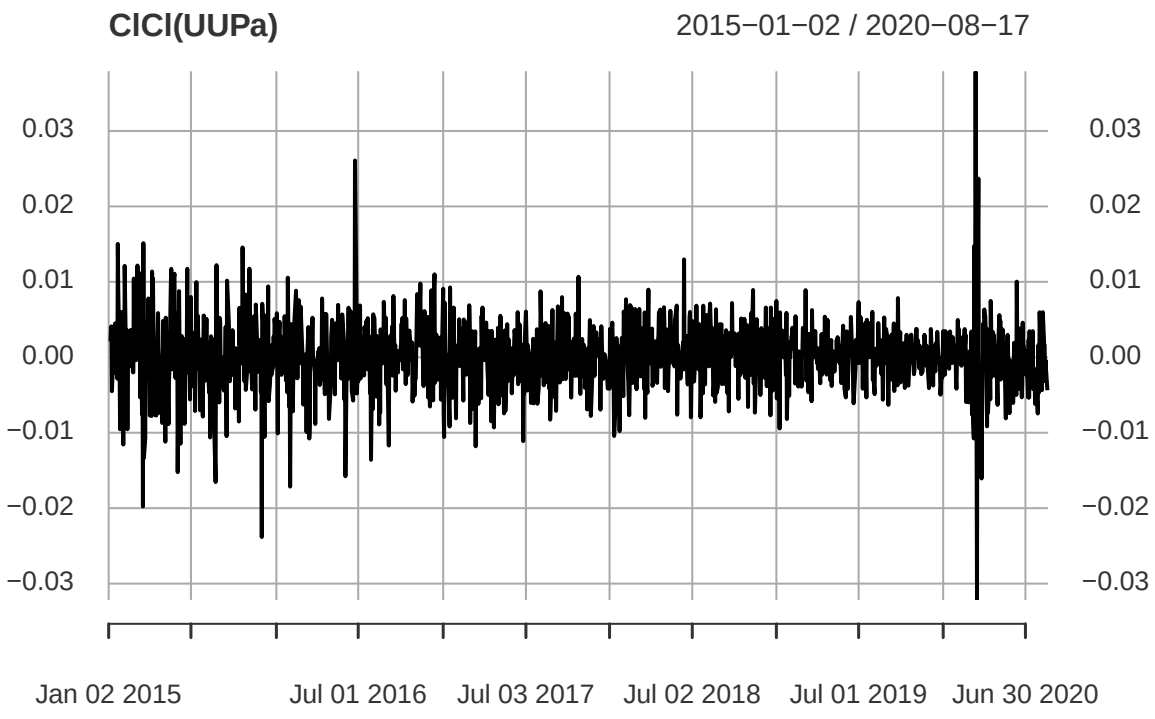
```
options(warn=-1)
UUPa = adjustOHLC(UUP)
```

```
FXEa = adjustOHLC(FXE)
```

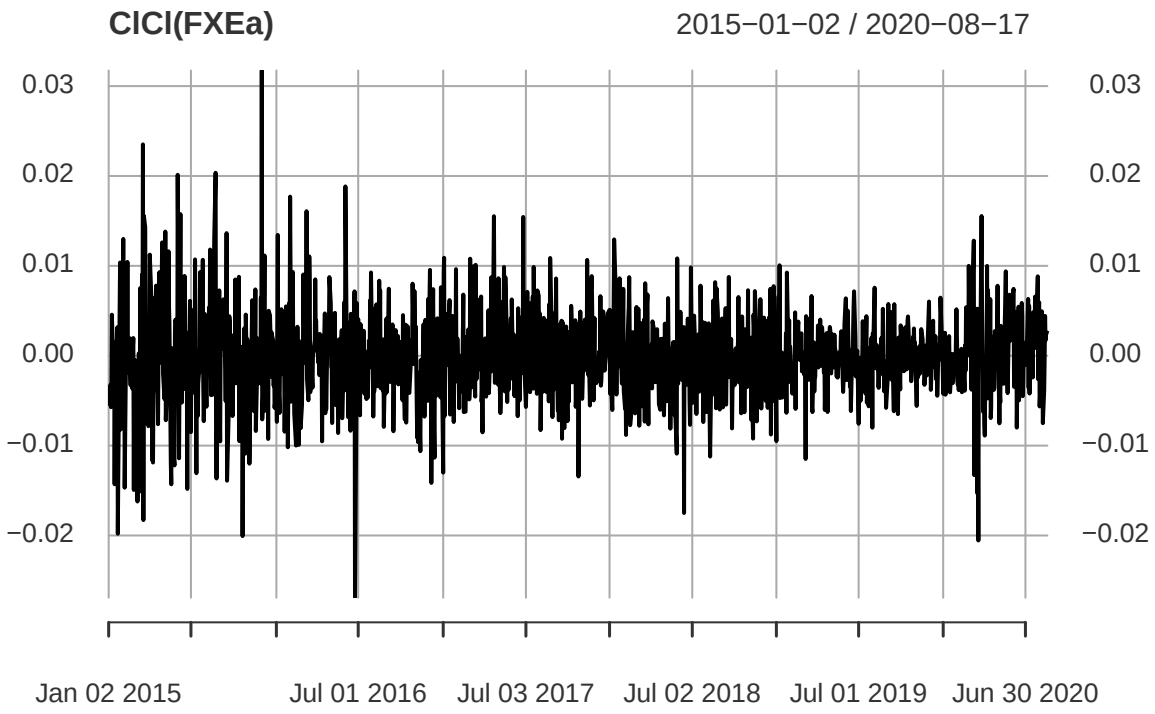
```
FXYa = adjustOHLC(FXY)
```

Plotting the close to close values for adjusted splits and dividends

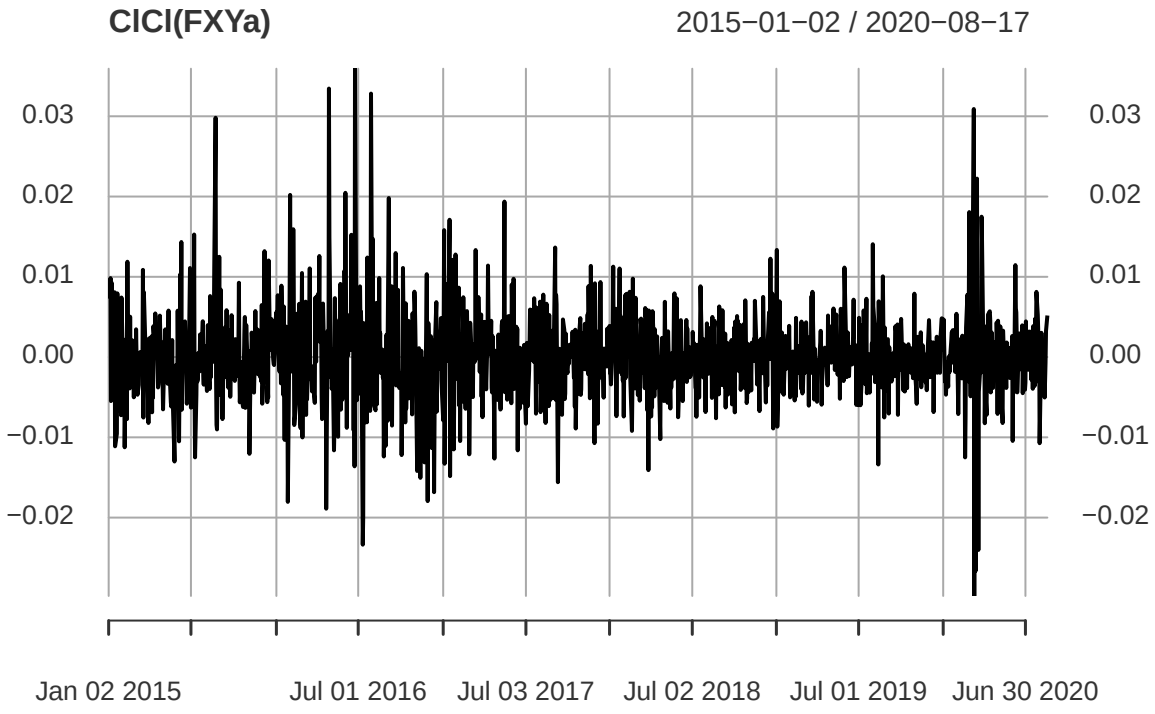
```
plot(ClCl(UUPa))
```




```
plot(ClCl(FXEa))
```




```
plot(ClCl(FXYa))
```



It is evident from the graph that the there is a fluctuation in the close to close values for the above given plortfolios sometime during the Mar 2020

Now we create a single data frame for all the 3 ETS returns

```
all_returns = cbind(ClCl(UUPa),ClCl(FXEa),ClCl(FXYa))
head(all_returns)
```

```
##               ClCl.UUPa    ClCl.FXEa    ClCl.FXYa
## 2015-01-02           NA           NA           NA
## 2015-01-05  0.002066074 -0.005583723  0.007309180
## 2015-01-06  0.002061814 -0.003232916  0.009838925
## 2015-01-07  0.003703704 -0.005718726 -0.005480416
## 2015-01-08  0.004100082 -0.003519581 -0.004163654
## 2015-01-09 -0.004491670  0.004565808  0.009222823
```

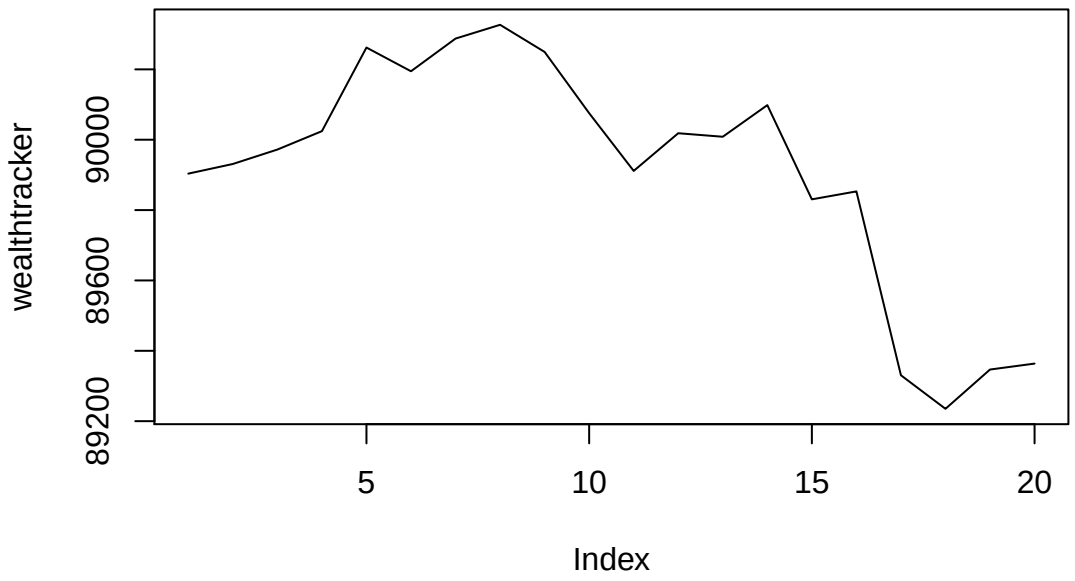
```
all_returns = as.matrix(na.omit(all_returns))
N = nrow(all_returns)
```

After getting all the ETFs into a single portfolio , we try to simulate a single trading day return and run it over 4 trading weeks to to get our total wealth over the given period of time

```
total_wealth = 100000
weights = c(0.3, 0.3, 0.3)
holdings = weights * total_wealth
n_days = 20  # capital T in the notes
wealthtracker = rep(0, n_days) # Set up a placeholder to track total wealth
for(today in 1:n_days) {
  return.today = resample(all_returns, 1, orig.ids=FALSE)
  holdings = holdings + holdings*return.today
  total_wealth = sum(holdings)
  wealthtracker[today] = total_wealth
}
total_wealth
```

```
## [1] 89363.65
```

```
plot(wealthtracker, type='l')
```



Now we run the code against many time to get a value of what the returns might look in the long run

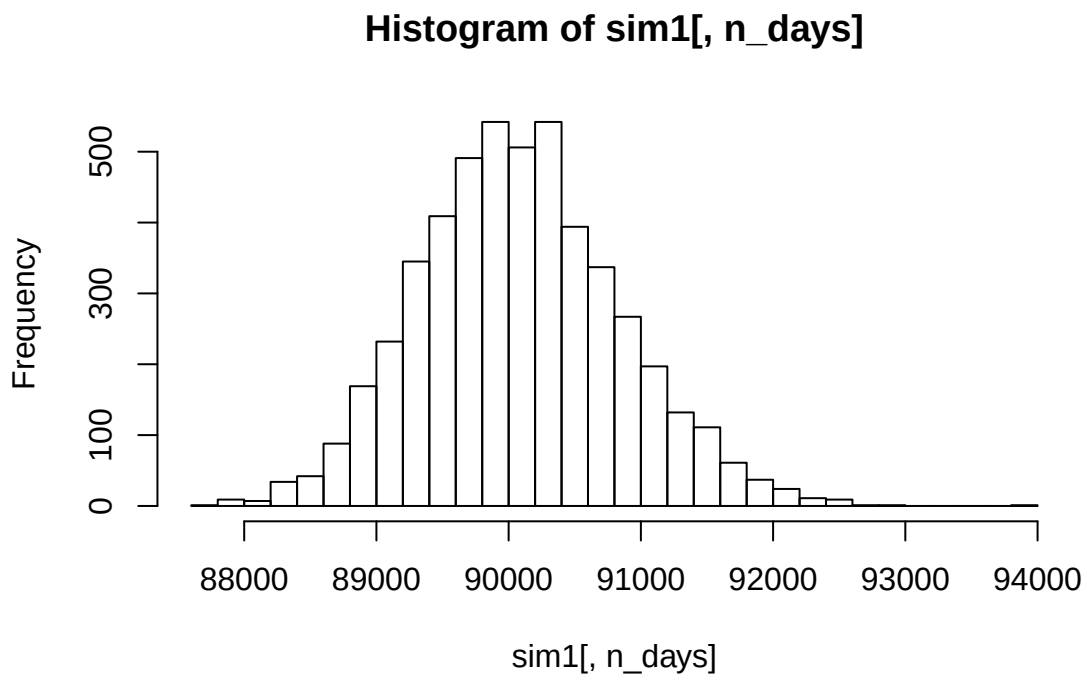
```
initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.3, 0.3, 0.3)
  holdings = weights * total_wealth
  n_days = 20
  wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
  }
  wealthtracker
}

head(sim1)
```

```
##              [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## result.1 89943.64 89872.45 89785.88 89656.37 89755.60 89992.73 89810.96
## result.2 89906.44 89807.59 89979.67 89911.39 89867.96 89724.91 89613.66
## result.3 90191.03 90122.65 90072.74 90260.90 90222.77 89969.80 89943.54
## result.4 90011.74 90145.81 90158.40 90012.49 89856.29 90116.74 90003.89
```

```
## result.5 89994.21 89581.50 89526.25 89399.23 89525.19 89710.33 89885.96
## result.6 89853.78 89872.12 89856.63 89965.07 89884.61 89855.37 89745.35
##              [,8]     [,9]    [,10]    [,11]    [,12]    [,13]    [,14]
## result.1 89662.83 89552.05 89385.98 89397.62 89345.64 89357.41 89457.82
## result.2 89596.93 89616.51 89995.66 89788.69 89492.79 89464.24 89482.33
## result.3 89933.52 90063.43 90332.64 89846.71 90039.58 90044.38 89934.76
## result.4 90357.50 90323.82 90400.87 90387.87 90364.35 90337.56 90212.75
## result.5 89859.15 89824.70 89661.36 89934.56 90093.80 90006.14 89904.61
## result.6 89793.15 90216.03 90155.21 90016.37 90175.98 90122.16 90115.55
##             [,15]    [,16]    [,17]    [,18]    [,19]    [,20]
## result.1 89477.35 89489.36 89390.13 89386.91 89391.46 89303.39
## result.2 89635.97 89403.36 89650.66 89529.44 89598.63 89623.62
## result.3 89874.01 90201.58 90350.76 90302.13 90371.63 90443.38
## result.4 90116.38 90058.17 90095.49 90298.25 90410.19 90375.69
## result.5 90002.31 90181.90 90333.93 90279.95 90222.73 90177.24
## result.6 89927.05 90040.97 90087.83 90215.25 90265.58 90223.38
```

```
hist(sim1[,n_days], 25)
```



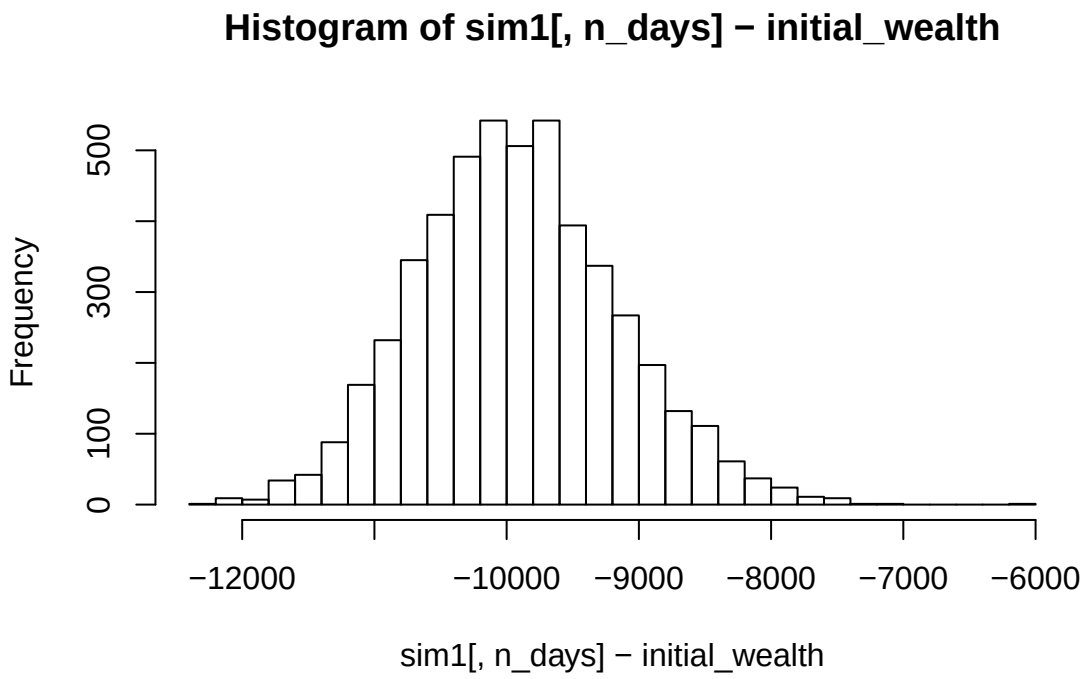

```
mean(sim1[,n_days])
```

```
## [1] 90083.06
```

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

```
## [1] -9916.937
```

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




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

```
##        5%
## -11105.44
```

And this gives us the 5% VAR for the currency portfolio that we have

Scenario 3: Here we are considering oil and gas ETF's

Oil & Gas ETFs invest directly in oil or gas and/or their subsidiary commodities. Note that these funds almost always utilize futures exposure to invest in their respective commodities.

USO United States Oil Fund

DBO Invesco DB Oil Fund

BNO United States Brent Oil Fund

```
mystocks = c("USO","DBO","BNO")
getSymbols(mystocks, from = "2015-01-01")
```

```
## [1] "USO" "DBO" "BNO"
```

```
options("getSymbols.warning4.0"=FALSE)
```

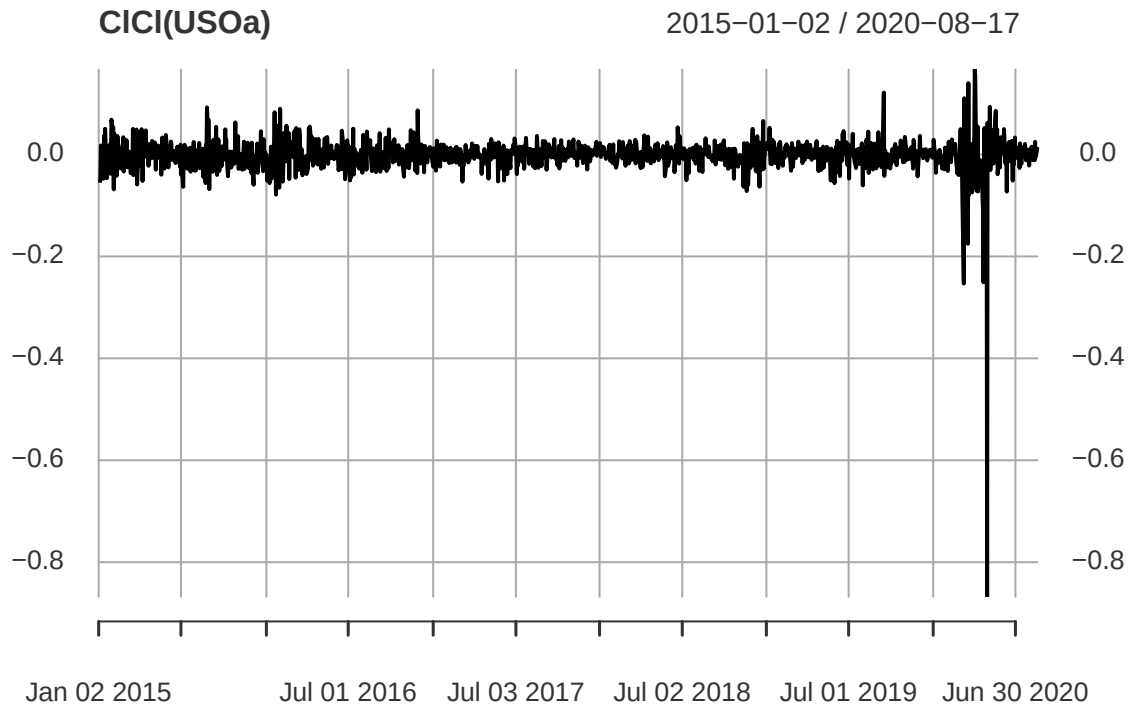
```
options(warn=-1)
USOa = adjustOHLC(USO)
```

```
DBOa = adjustOHLC(DBO)
```

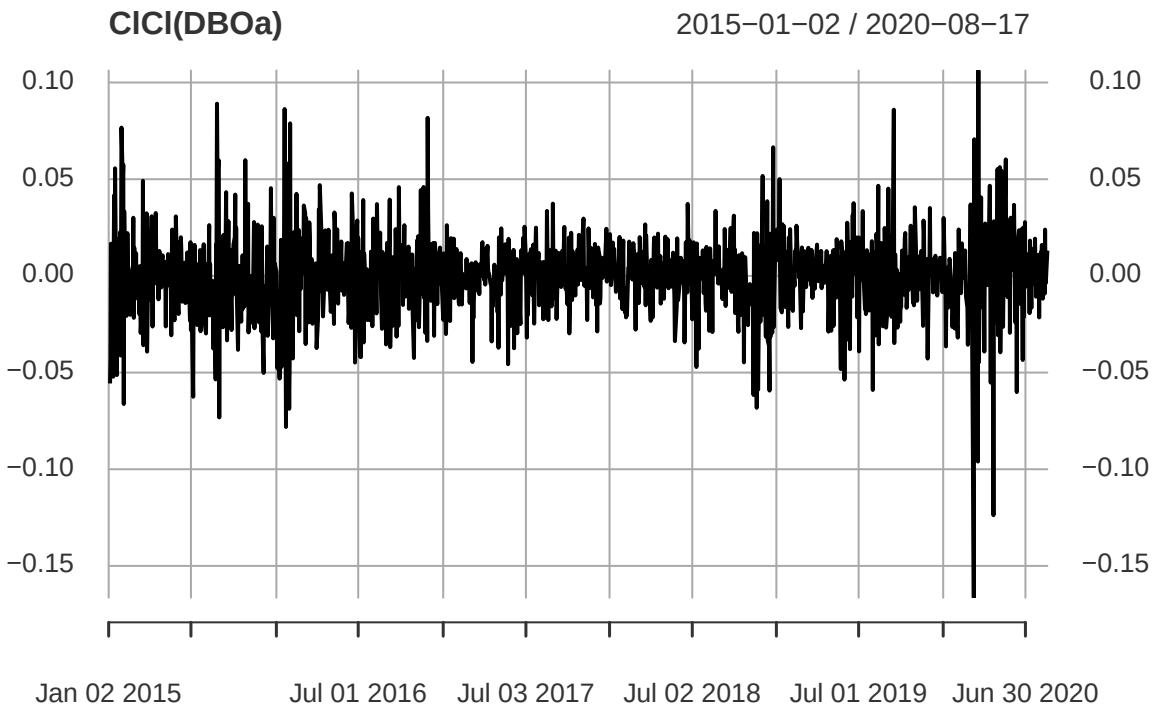
```
BNOa = adjustOHLC(BNO)
```

Now we plot the close to close value for each of the stocks

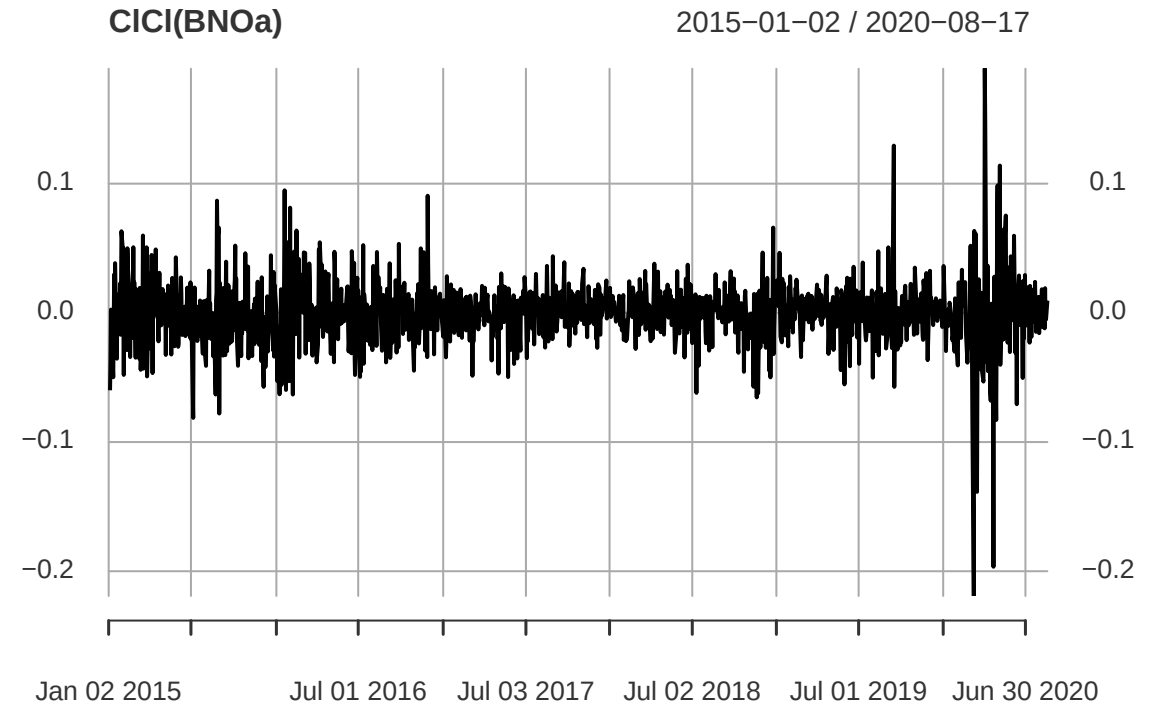
```
plot(ClCl(USOa))
```




```
plot(ClCl(DBOa))
```




```
plot(ClCl(BNOa))
```



As expected all the stocks show high amount of fluctuations during the covid pandemic

Now we get all the returns into a single data frame

```
all_returns = cbind(ClCl(USOa),ClCl(DBOa),ClCl(BNOa))
head(all_returns)
```

```
##               ClCl.USOa   ClCl.DBOa    ClCl.BNOa
## 2015-01-02           NA          NA           NA
## 2015-01-05 -0.055304099 -0.05562827 -0.059945459
## 2015-01-06 -0.039382735 -0.04227304 -0.037681206
## 2015-01-07  0.017728623  0.01664255  0.002509990
## 2015-01-08  0.009798496  0.01067616  0.002003055
## 2015-01-09 -0.014555183 -0.01408451 -0.017491254
```

```
all_returns = as.matrix(na.omit(all_returns))
N = nrow(all_returns)
```

After getting the stocks into a single dataframe we can now directly run our simulation model for n number of times over 4 trading weeks period

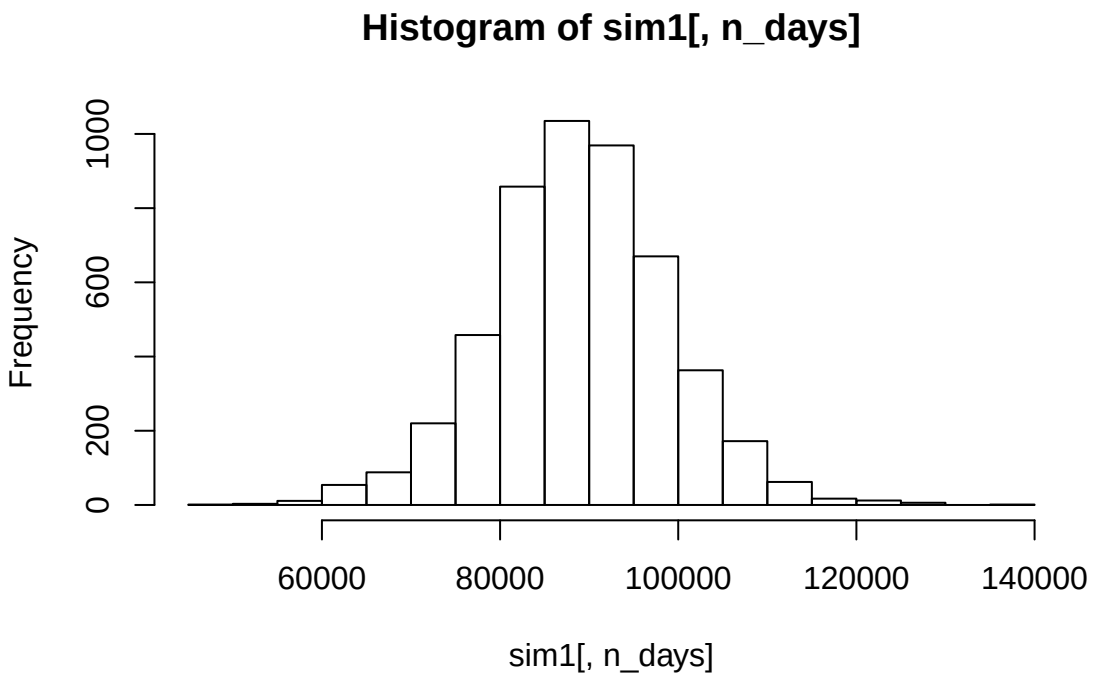
```
initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.3, 0.3, 0.3)
  holdings = weights * total_wealth
  n_days = 20
  wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
  }
  wealthtracker
}

head(sim1)
```

```
##              [,1]     [,2]     [,3]     [,4]     [,5]     [,6]     [,7]
## result.1 88933.51 89150.35 90503.68 91545.41 92202.78 91168.31 88384.97
## result.2 89732.68 95041.96 93830.29 94287.68 95207.69 95958.40 92686.81
## result.3 88586.00 89180.61 87342.46 88742.32 90713.34 91038.57 90907.69
## result.4 91340.82 91780.58 92168.15 93048.84 91618.27 93738.75 94526.34
## result.5 87708.56 86825.57 85633.71 87559.01 91097.21 90213.53 89997.08
## result.6 95944.52 96026.25 96645.41 99108.73 97704.86 94671.36 96160.19
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  90019.66  90986.61  86978.71  86131.81  84655.70  84535.87  84561.12
## result.2  91403.57  93344.35  90661.78  91421.83  90299.63  87222.26  87261.93
## result.3  93316.90  88666.73  90125.08  85025.27  83059.45  81722.18  81519.92
```

```
## result.4  91546.48  87042.22  85981.42  89771.35  90122.34  90327.53  93576.42
## result.5  86224.23  86072.51  86148.24  86038.16  85145.73  84490.40  84212.22
## result.6 100923.77 101177.52 102045.64 103629.28 104757.83 103119.38 105373.61
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  83941.97  87837.36  87681.50  87226.83  90951.73  92884.82
## result.2  87314.74  86287.96  86365.63  87379.05  88344.39  89953.17
## result.3  81298.25  81617.85  82791.41  82544.49  84003.83  87795.23
## result.4  90025.72  92430.29  94358.10  92090.71  93591.61  96236.52
## result.5  84261.32  83541.93  81709.67  81074.75  79709.48  77543.16
## result.6 106538.35 108738.89 108328.64 107868.25 109723.28 111198.23
```

```
hist(sim1[,n_days], 25)
```



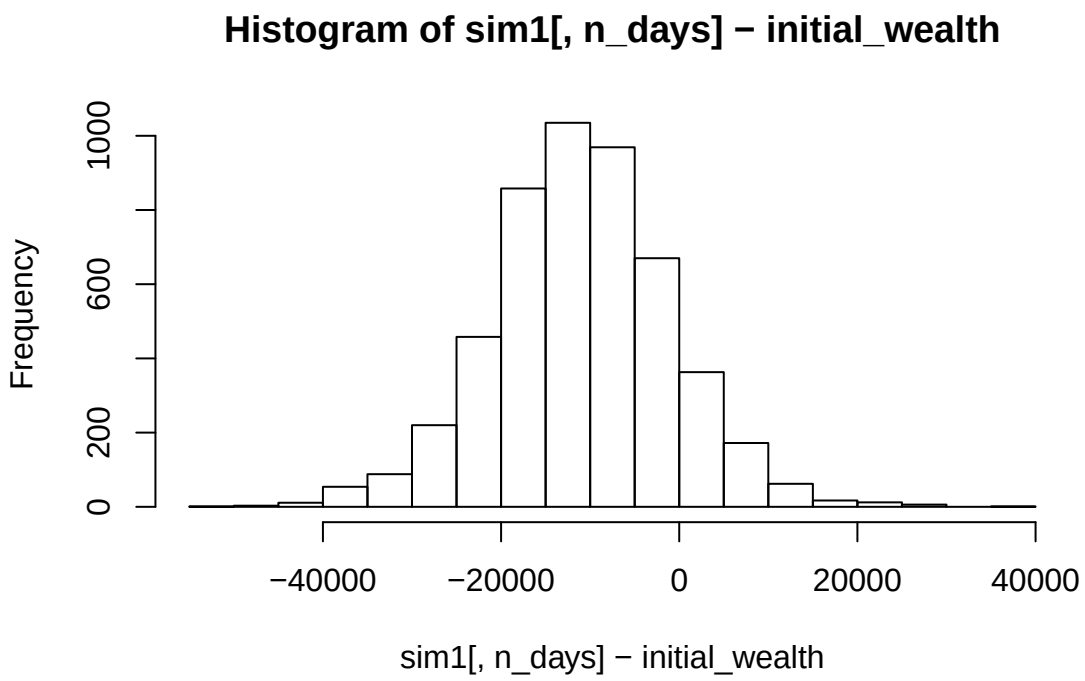

```
mean(sim1[,n_days])
```

```
## [1] 88972.87
```

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

```
## [1] -11027.13
```

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



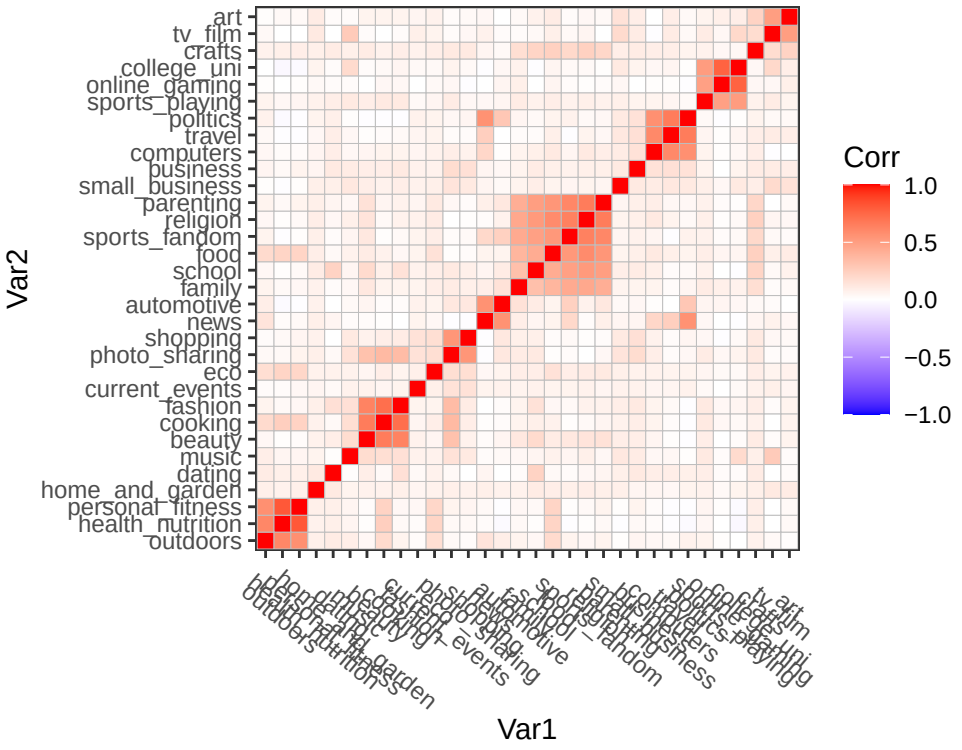

```
quantile(sim1[,n_days]- initial_wealth, prob=0.05)
```

```
##       5%
## -27129.2
```

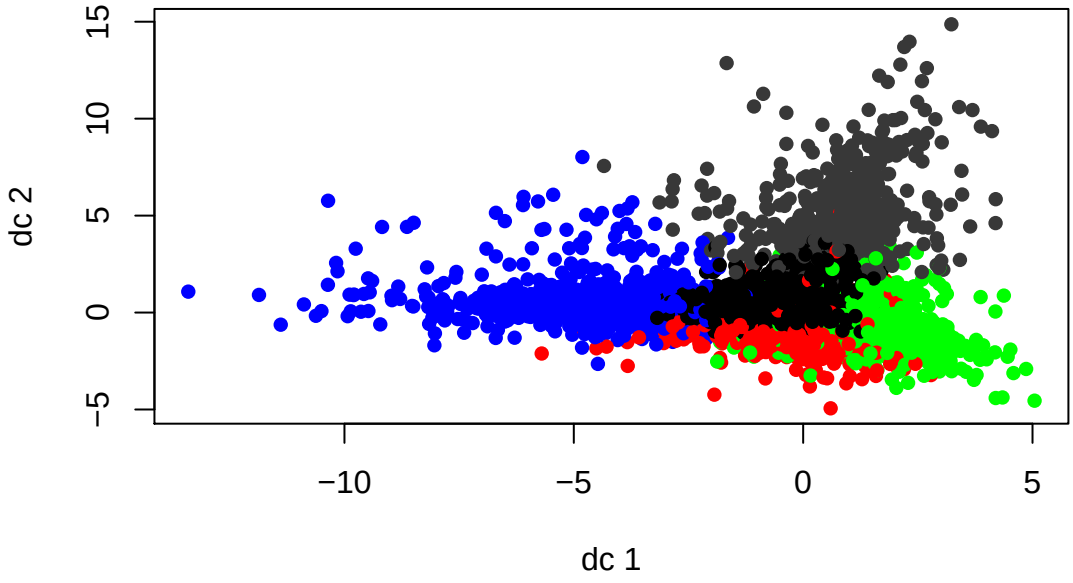
And above is the 5% VAR for oil and gas based ETFs

Market Segmentation

This correlation uses hierachical clustering to show what variables are correlated the most. It will serve as our baseline for our clustering model.



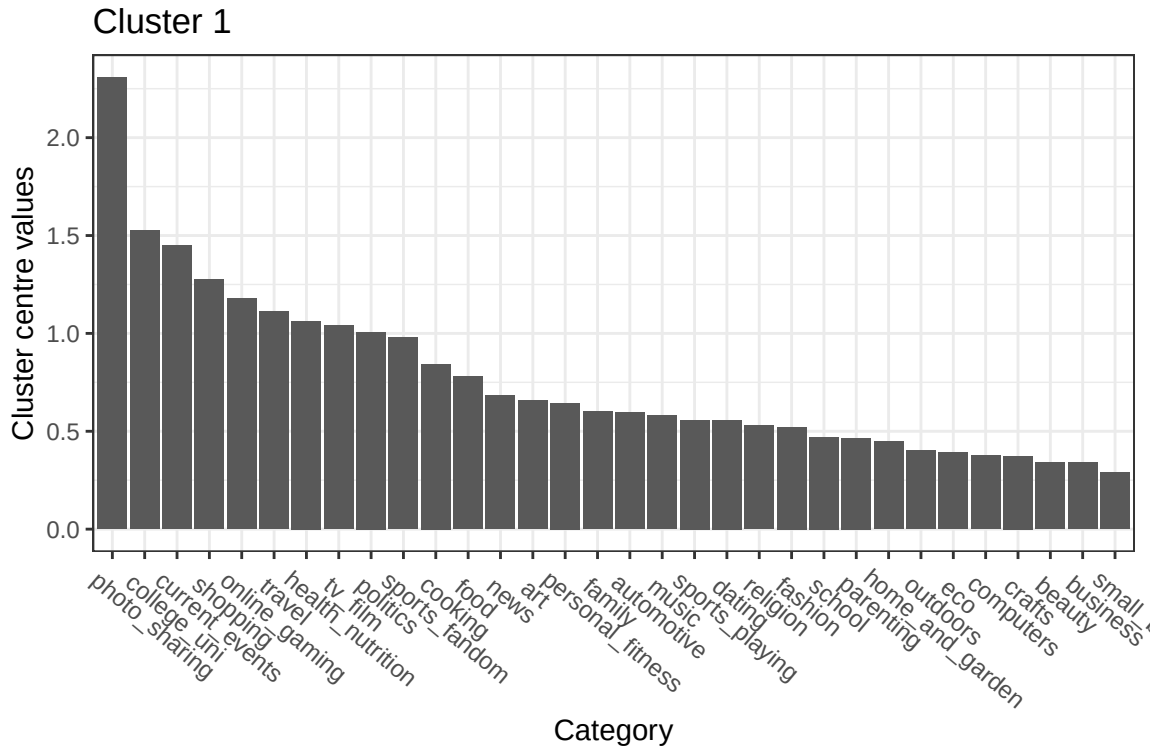
Below are graphs that reflect the complexity versus variance explained trade-off. We also decided to focus on a clustering equal to five since the data seperates itself fairly well at k=5.

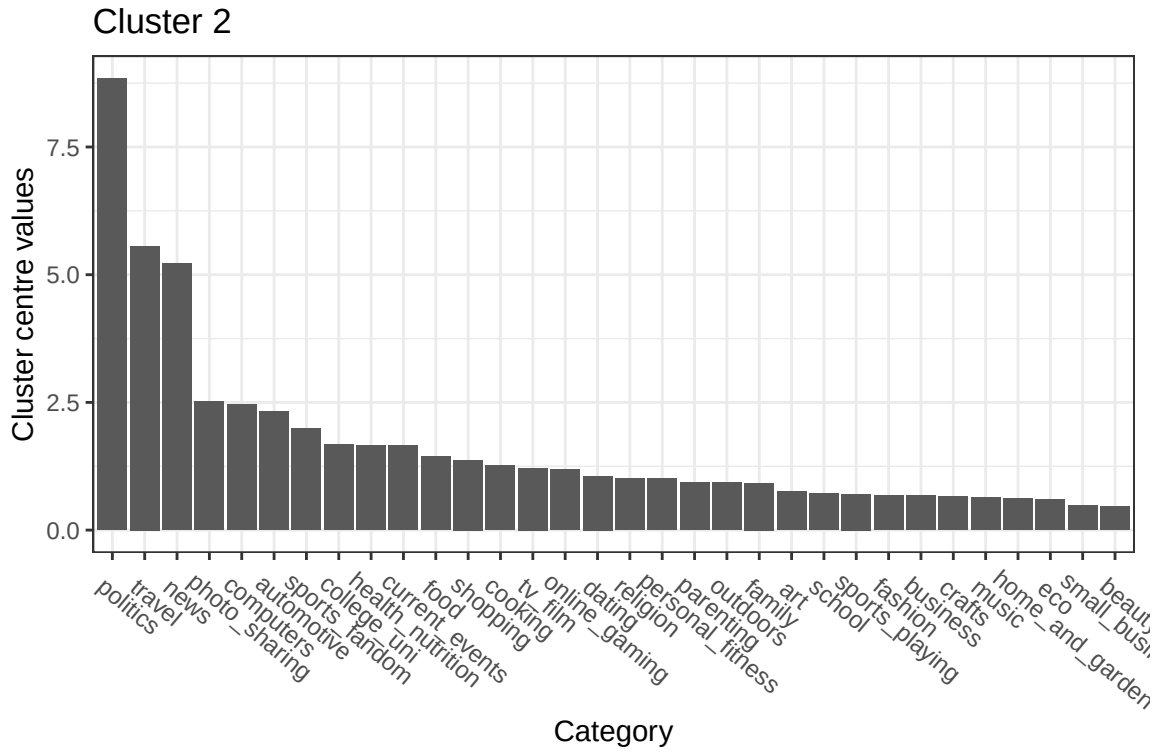


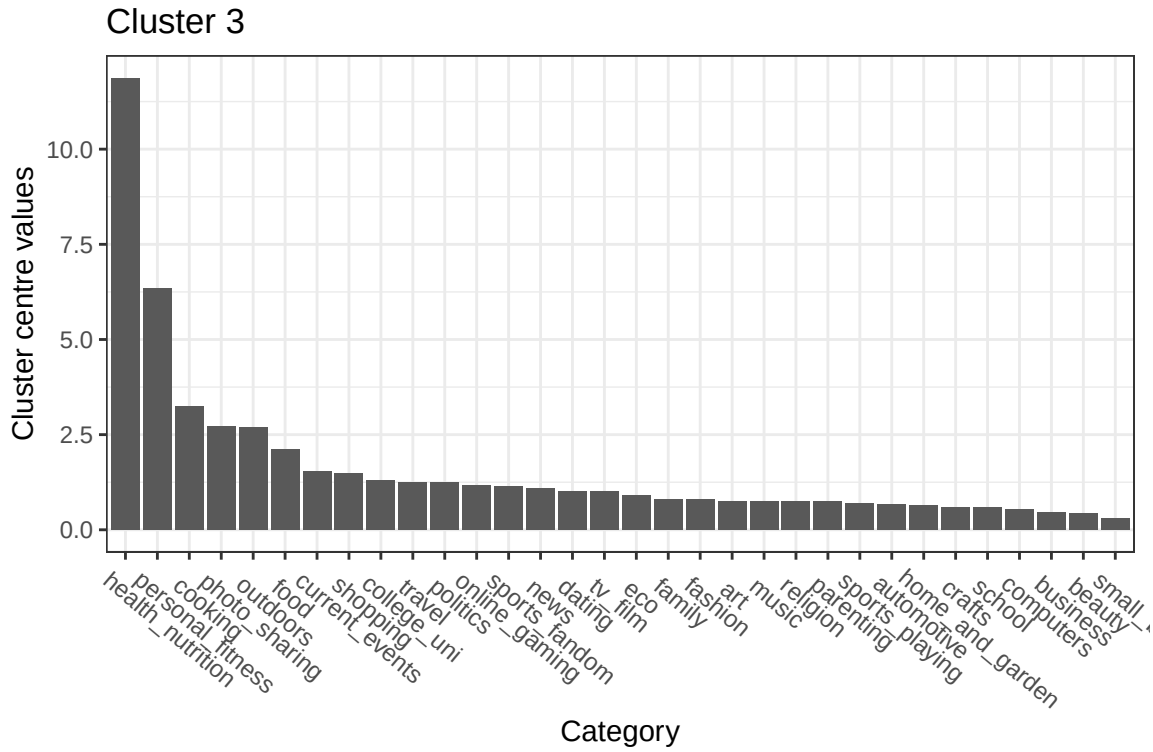

```
##     Cluster_1       Cluster_2       Cluster_3       Cluster_4
```

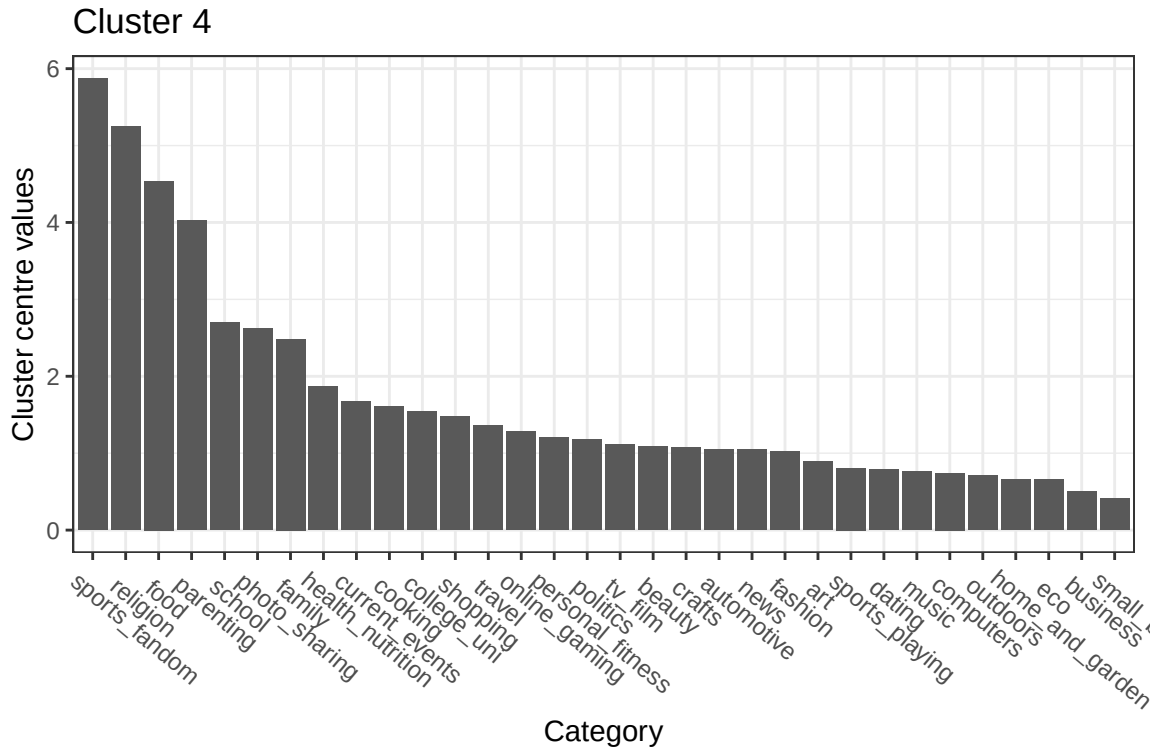
```
##  Min.   :0.2883   Min.   :0.4677   Min.   : 0.2958   Min.   :0.4104
##  1st Qu.:0.4611   1st Qu.:0.7001   1st Qu.: 0.6835   1st Qu.:0.8011
##  Median :0.5976   Median :1.0372   Median : 0.9641   Median :1.1410
##  Mean   :0.7619   Mean   :1.6633   Mean   : 1.6156   Mean   :1.6874
##  3rd Qu.:1.0150   3rd Qu.:1.6661   3rd Qu.: 1.3449   3rd Qu.:1.7246
##  Max.   :2.3079   Max.   :8.8441   Max.   :11.8692   Max.   :5.8742
##    Cluster_5
##  Min.   : 0.5221
##  1st Qu.: 0.8421
##  Median : 1.0638
##  Mean   : 1.7844
##  3rd Qu.: 1.5655
##  Max.   :10.5810
```

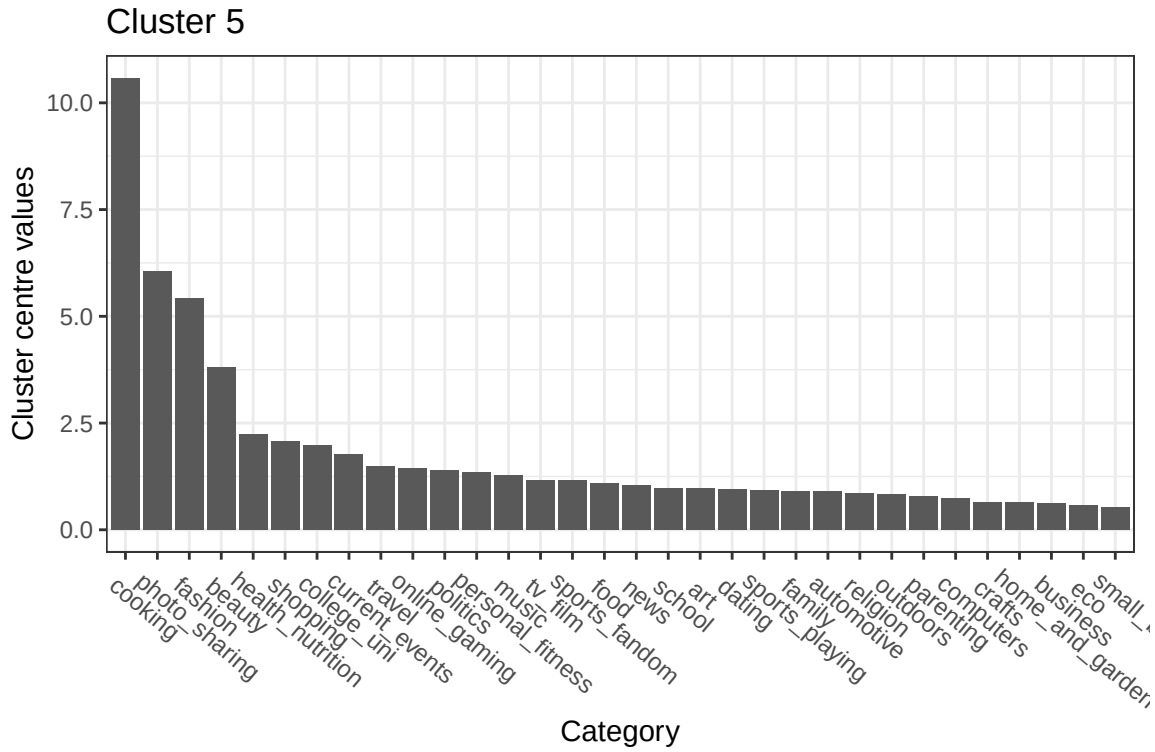
To create these plots, we first had to create a data frame of each cluster, rename the resulting data frame columns, then add back the names of the categories by adding another column so that it can be used to sort the x values of the resulting plots.




Cluster 2
Cluster centre values
7.5
5.0
2.5
0.0
politics
travel
news
photo_sharing
computers
automotive
sports_fandom
college_uni
health_nutrition
current_events
food
shopping
cooking
tv_film
online_gaming
dating
religion
personal_fitness
parenting
outdoors
family
art
school
sports_playing
fashion
business
crafts
music
home_and_garden
eco
small_busi
beauty
Category


Cluster 3
Cluster centre values
10.0
7.5
5.0
2.5
0.0
health_nutrition
personal_fitness
cooking
photo_sharing
outdoors
food
current_events
shopping
college_uni
travel
politics
online_gaming
sports_fandom
news
dating
tv_film
eco
family
fashion
art
music
religion
parenting
sports_playing
automotive
home_and_garden
crafts
school
computers
business
beauty
small_
Category


Cluster 4
Cluster centre values
6
4
2
0
sports_fandom
religion
food
parenting
school
photo_sharing
family
health_nutrition
current_events
cooking
college_uni
shopping
travel
online_gaming
personal_fitness
politics
tv_film
beauty
crafts
automotive
news
fashion
art
sports_playing
dating
music
computers
outdoors
home_and_garden
eco
business
Category


Cluster 5
Cluster centre values
10.0
7.5
5.0
2.5
0.0
cooking
photo_sharing
fashion
beauty
health_nutrition
shopping
college_uni
current_events
travel
online_gaming
politics
personal_fitness
music
tv_film
sports_fandom
food
news
school
art
dating
sports_playing
family
automotive
religion
outdoors
parenting
computers
crafts
home_and_garden
business
eco
Category

Conclusion

We plan on using these clusters as market segments based on the the major categories found in each cluster.

Cluster 1 includes tweets considered politics, travel and news. Politics and travel do appear closely together in the hierarchical correlation matrix, but news does not. These individuals may include business professionals who pay close attention to the news and politcal information. They are also known to travel often.

Cluster 2 includes sports fandom, religion, food and parenting. This is the biggest correlation matrix found in our baseline analysis. This cluster may be geared toward family oriented participants who share these values, but it is odd to see sports fandom in this mix. It seems like an outlier when compared to the other categories.

Cluster 3 includes cooking, photo sharing, fashion and beauty. Photo sharing is not included in the correlation matrix cluster that was observed for cooking, fashion, and beauty. This cluster may be based on gender, since fashion and beauty would be more associated with females rather than males.

Cluster 4 includes categories such as photo sharing, college/universities, current_events, shopping, and online gaming. While the category of current_events seems out of place in these categories, we imagine cluster 2 to include younger individuals in college who are more more active on the internet.

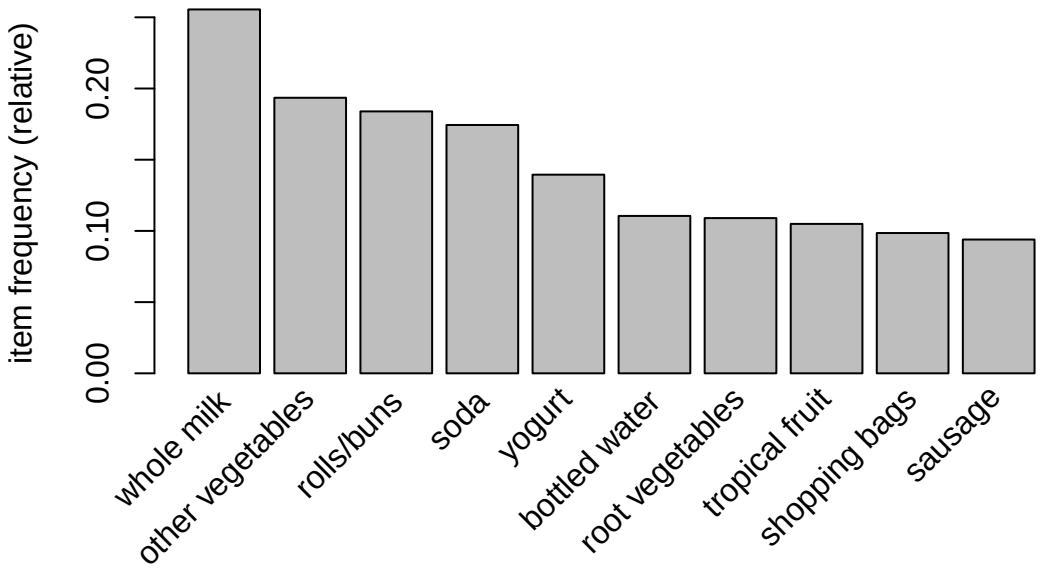
Cluster 5 is dominated by health/nutrition then followed by personal fitness and cooking. The correlation matrix captures two of these factors closely together, but cooking is replaced with outdoors instead. It seems to be that these individuals are very health and fitness focused.

Association Rule Mining

Read in groceries dataset, make each line contain all the items in the transaction

We split each transaction into baskets, and transformed the data into a "transaction class". We found that whole milk is the most frequent item in the transactions, occuring in 2513 baskets out of the 9835 transaction.

Our plot shows the ten most frequent items in the transactions



Let us explore rules with low support > 0.02, confidence > 0.1 and the length being 2 to find relationships between two items.

We ended up with 120 rules, because we had low support and higher confidence. For the first rule, given someone had frozen vegetables in their basket we are 42% positive that they'll also get whole milk. Whole milk is a very common occurence as shown by the data before.

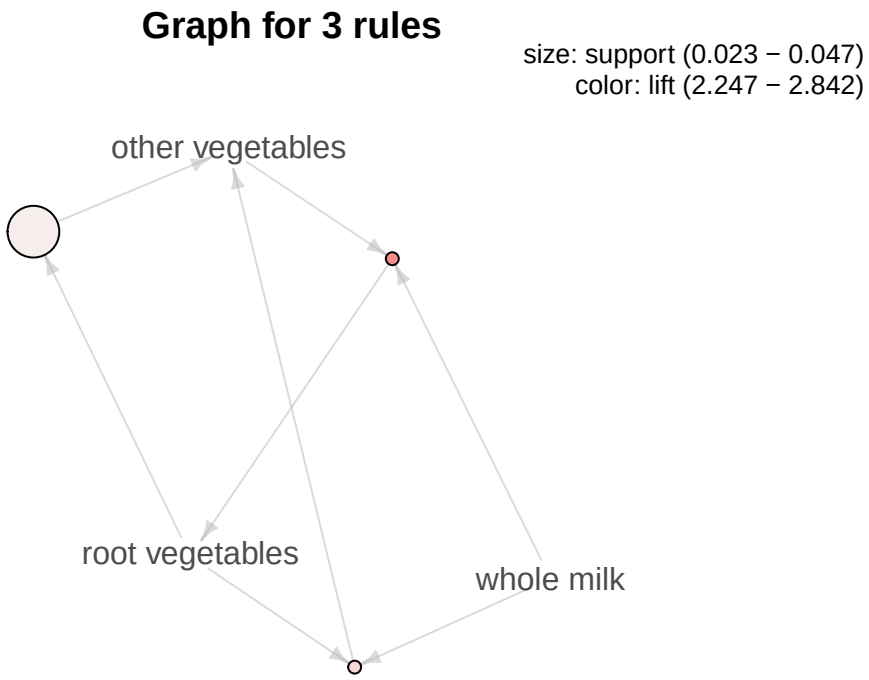
Let us decrease support and also increase confidence with support > 0.01 and confidence > 0.2

With a higher confidence we lowered the number of rules. We decided to plot by lift as it shows a change in probability of getting an item when another is in the basket.

```
##      lhs                            rhs                support
## [1]  {frozen vegetables}         => {whole milk}       0.02043721
## [2]  {beef}                      => {whole milk}       0.02125064
## [3]  {curd}                      => {whole milk}       0.02613116
## [4]  {pork}                      => {other vegetables} 0.02165735
## [5]  {pork}                      => {whole milk}       0.02216573
## [6]  {frankfurter}               => {whole milk}       0.02053889
## [7]  {brown bread}               => {whole milk}       0.02521607
## [8]  {margarine}                 => {whole milk}       0.02419929
## [9]  {butter}                    => {other vegetables} 0.02003050
## [10] {butter}                    => {whole milk}       0.02755465
## [11] {newspapers}                => {whole milk}       0.02735130
## [12] {domestic eggs}             => {other vegetables} 0.02226741
## [13] {domestic eggs}             => {whole milk}       0.02999492
## [14] {fruit/vegetable juice}     => {whole milk}       0.02663955
## [15] {whipped/sour cream}        => {other vegetables} 0.02887646
## [16] {whipped/sour cream}        => {whole milk}       0.03223183
## [17] {pip fruit}                 => {other vegetables} 0.02613116
## [18] {pip fruit}                 => {whole milk}       0.03009659
```

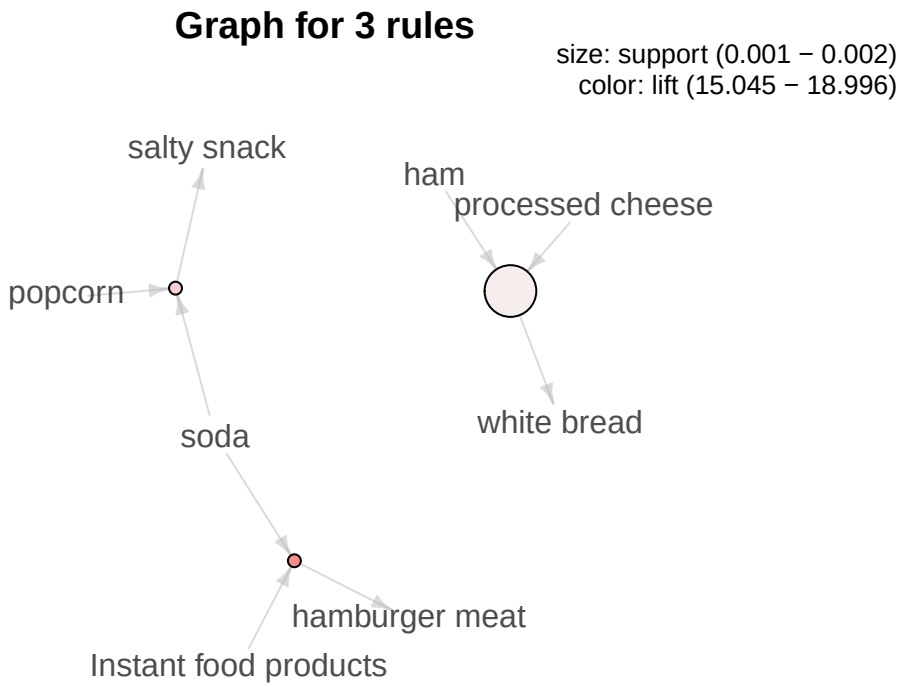
```
## [19] {pastry}                            => {whole milk}       0.03324860
## [20] {citrus fruit}                      => {other vegetables} 0.02887646
## [21] {citrus fruit}                      => {whole milk}       0.03050330
## [22] {sausage}                           => {rolls/buns}       0.03060498
## [23] {sausage}                           => {whole milk}       0.02989324
## [24] {bottled water}                     => {whole milk}       0.03436706
## [25] {tropical fruit}                    => {other vegetables} 0.03589222
## [26] {tropical fruit}                    => {whole milk}       0.04229792
## [27] {root vegetables}                   => {other vegetables} 0.04738180
## [28] {root vegetables}                   => {whole milk}       0.04890696
## [29] {yogurt}                            => {other vegetables} 0.04341637
## [30] {yogurt}                            => {whole milk}       0.05602440
## [31] {rolls/buns}                        => {whole milk}       0.05663447
## [32] {other vegetables}                  => {whole milk}       0.07483477
## [33] {other vegetables,root vegetables} => {whole milk}       0.02318251
## [34] {root vegetables,whole milk}        => {other vegetables} 0.02318251
## [35] {other vegetables,whole milk}       => {root vegetables}  0.02318251
## [36] {other vegetables,yogurt}           => {whole milk}       0.02226741
## [37] {whole milk,yogurt}                 => {other vegetables} 0.02226741
##      confidence coverage   lift     count
## [1]  0.4249471  0.04809354 1.663094 201
## [2]  0.4050388  0.05246568 1.585180 209
## [3]  0.4904580  0.05327911 1.919481 257
## [4]  0.3756614  0.05765125 1.941476 213
## [5]  0.3844797  0.05765125 1.504719 218
## [6]  0.3482759  0.05897306 1.363029 202
## [7]  0.3887147  0.06487036 1.521293 248
## [8]  0.4131944  0.05856634 1.617098 238
## [9]  0.3614679  0.05541434 1.868122 197
## [10] 0.4972477  0.05541434 1.946053 271
## [11] 0.3426752  0.07981698 1.341110 269
## [12] 0.3509615  0.06344687 1.813824 219
## [13] 0.4727564  0.06344687 1.850203 295
## [14] 0.3684951  0.07229283 1.442160 262
## [15] 0.4028369  0.07168277 2.081924 284
## [16] 0.4496454  0.07168277 1.759754 317
## [17] 0.3454301  0.07564820 1.785237 257
## [18] 0.3978495  0.07564820 1.557043 296
## [19] 0.3737143  0.08896797 1.462587 327
## [20] 0.3488943  0.08276563 1.803140 284
## [21] 0.3685504  0.08276563 1.442377 300
## [22] 0.3257576  0.09395018 1.771048 301
## [23] 0.3181818  0.09395018 1.245252 294
## [24] 0.3109476  0.11052364 1.216940 338
## [25] 0.3420543  0.10493137 1.767790 353
## [26] 0.4031008  0.10493137 1.577595 416
## [27] 0.4347015  0.10899847 2.246605 466
## [28] 0.4486940  0.10899847 1.756031 481
## [29] 0.3112245  0.13950178 1.608457 427
## [30] 0.4016035  0.13950178 1.571735 551
## [31] 0.3079049  0.18393493 1.205032 557
## [32] 0.3867578  0.19349263 1.513634 736
## [33] 0.4892704  0.04738180 1.914833 228
## [34] 0.4740125  0.04890696 2.449770 228
```

```
## [35] 0.3097826  0.07483477 2.842082 228
## [36] 0.5128806  0.04341637 2.007235 219
## [37] 0.3974592  0.05602440 2.054131 219
```



Lets try to decrease support further and increase confidence even more with support > 0.005, confidence > 0.6 and length > 2

We then plot the rules for 3 rules by the lift. From the plot we see that people who buy processed cheese and ham are more likely to buy white bread, possibly for sandwiches.



In Conclusion: 1. Whole milk is the most common item purchased by customers. 2. People who buy processed cheese and ham are more likely to buy white bread. 3. People who buy popcorn and soda are more likely to buy salty snacks.

Author Attribution

The idea for this problem was to compile all of the texts, clean the data down so that it is easier to analyze then combine the names of the authors that corresponds to each text file. Then we could comput a TF-IDF for each document that would be associated for each author which would be used to predict which document belongs to a certain author. Some models we considered using were k-mean clustering and tree models. This is where we would implement the test data set into our models to generate a prediction accuracy and compare across the models we considered.