Homework Question Set

Raymond Li, Emily McCullough, Jaquelin Solis

8/16/2021

1. Green Buildings

We wanted to help our developer make the correct investment choice, as well as validating the stat guru's investigation. On the surface, the guru's logic makes sense, but he does not consider outliers and other possible confounding variables. We took it upon ourselves to compare different aspects of buildings such as rent, leasing rate, building size, class, renovations, number of stories, and amenities to investigate their effects on the the net potential value of building a green building.

Reading in libraries and dataset

```
library(ggplot2)
library(tidyverse)
green <- read_csv('greenbuildings.csv')
```

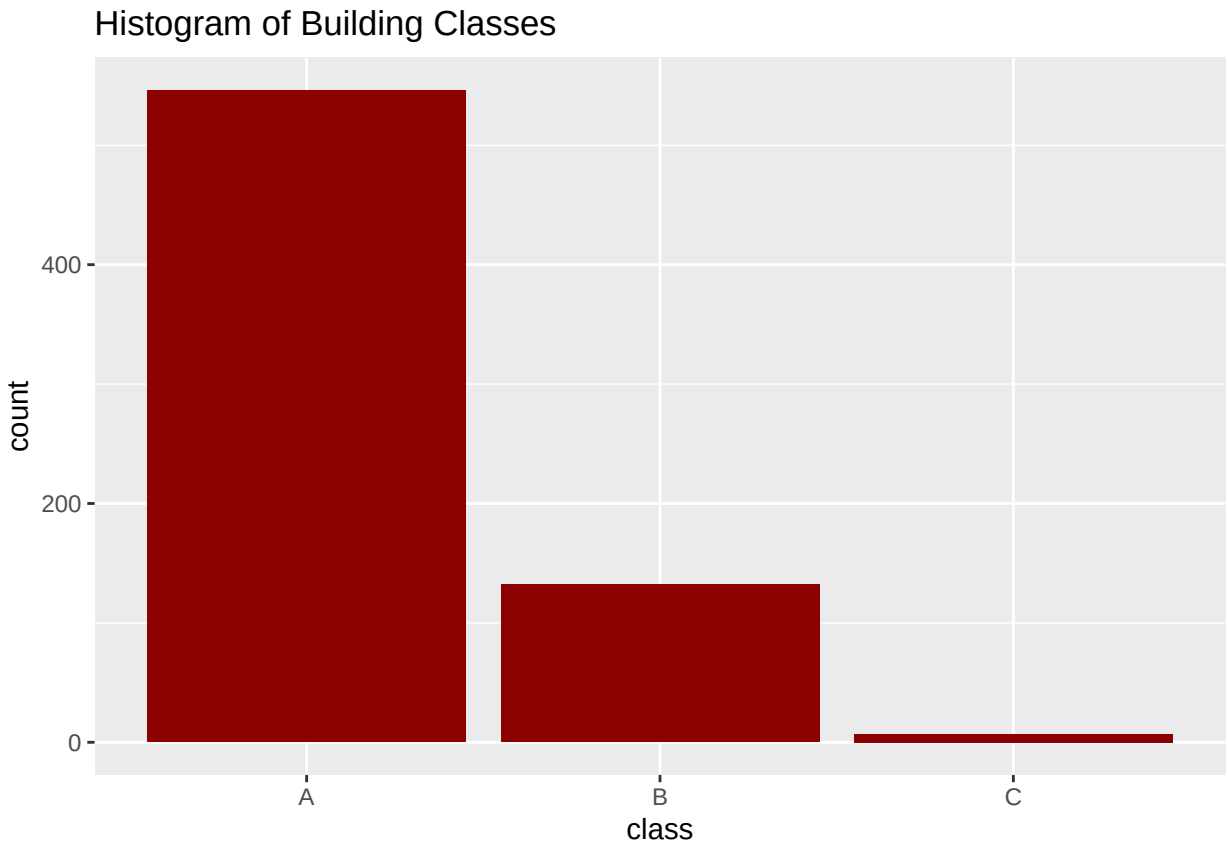
Subset green buildings and non-green buildings

```
green$class <- ifelse(green$class_a == 1, 'A', ifelse(green$class_b, 'B', 'C'))
green_only = subset(green, green_rating==1)
nongreen_only = subset(green, green_rating == 0)
```

Find how many green buildings are in each class

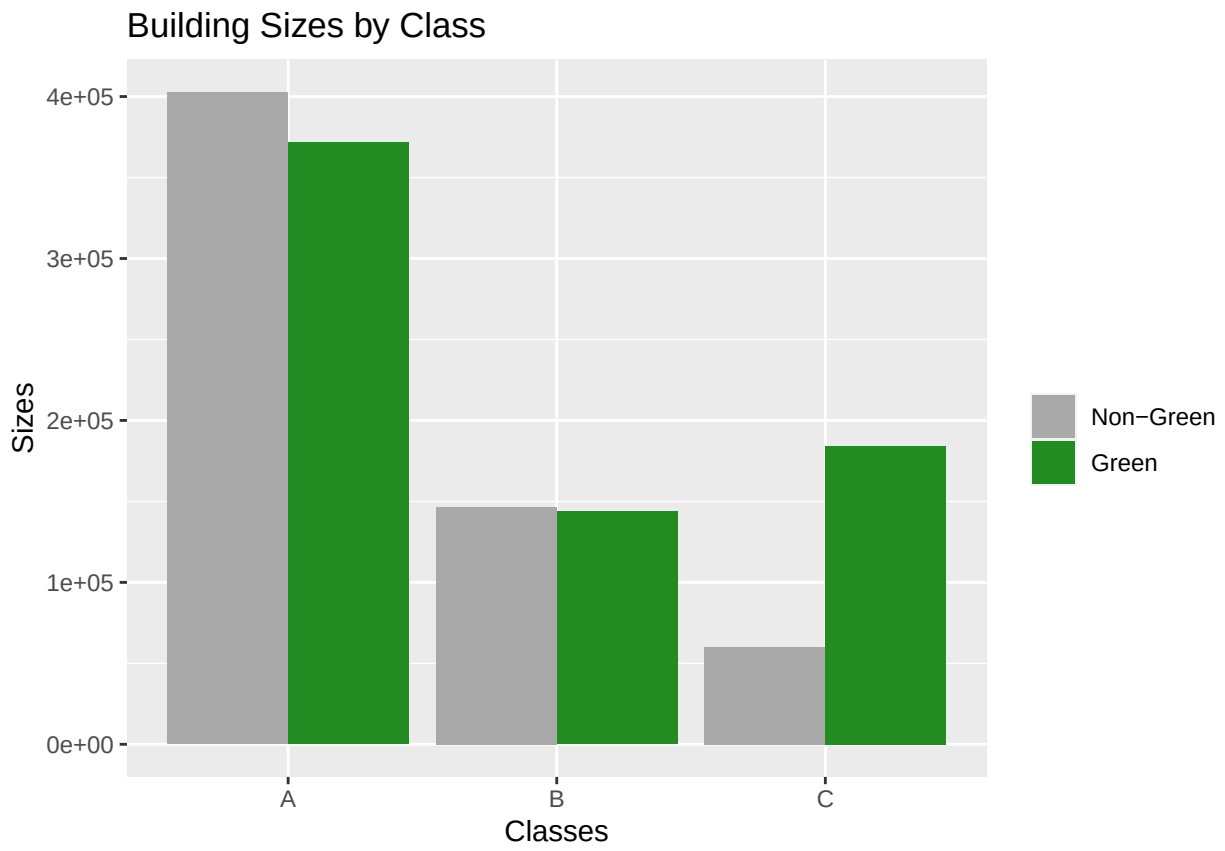
```
hist <- ggplot(green_only,aes(x = class))

hist + geom_histogram(stat = 'count', fill = "darkred" ) + ggtitle('Histogram of Building Classes')
```

```
## Warning: Ignoring unknown parameters: binwidth, bins, pad
```



The majority of green buildings are in Class A, the most desirable class. This also fits with the type of building we would expect in the area of Austin the developer is looking at. Therefore, we can reasonably assume the developer would be looking to build a Class A building and we should conduct our analysis on this class.

```
mean_sizes <- aggregate(green$size, by = list(id1 = green$class, id2 = green$green_rating), FUN = mean)

g <- ggplot(mean_sizes,aes(x = id1, y = x, fill = as.factor(id2)))

g + geom_bar(position = 'dodge', stat = 'identity')+
  scale_fill_manual("id2",labels = c('Non-Green', 'Green'), values = c('#A9A9A9', '#228B22')) +
  xlab('Classes') + ylab('Sizes') + ggtitle('Building Sizes by Class') +
  guides(fill=guide_legend(title=""))
```

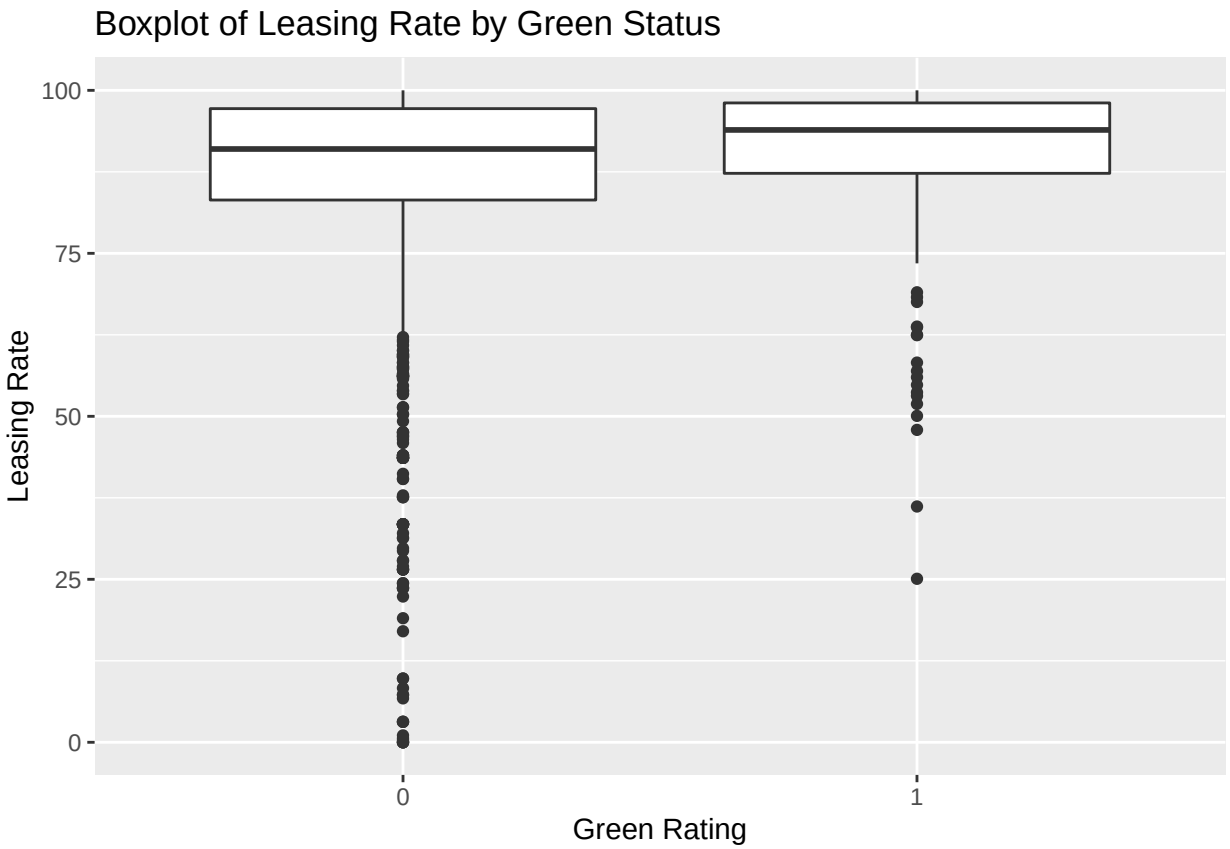


There is not a significant difference in sizes between green and non-green buildings in Class A. This is an indicator that the total rent for green vs non-green buildings may not be different.

To explore this, let's look at leasing rates.

```
#get all buildings in Class A

green_classA <- subset(green, green$class_a== 1)
```

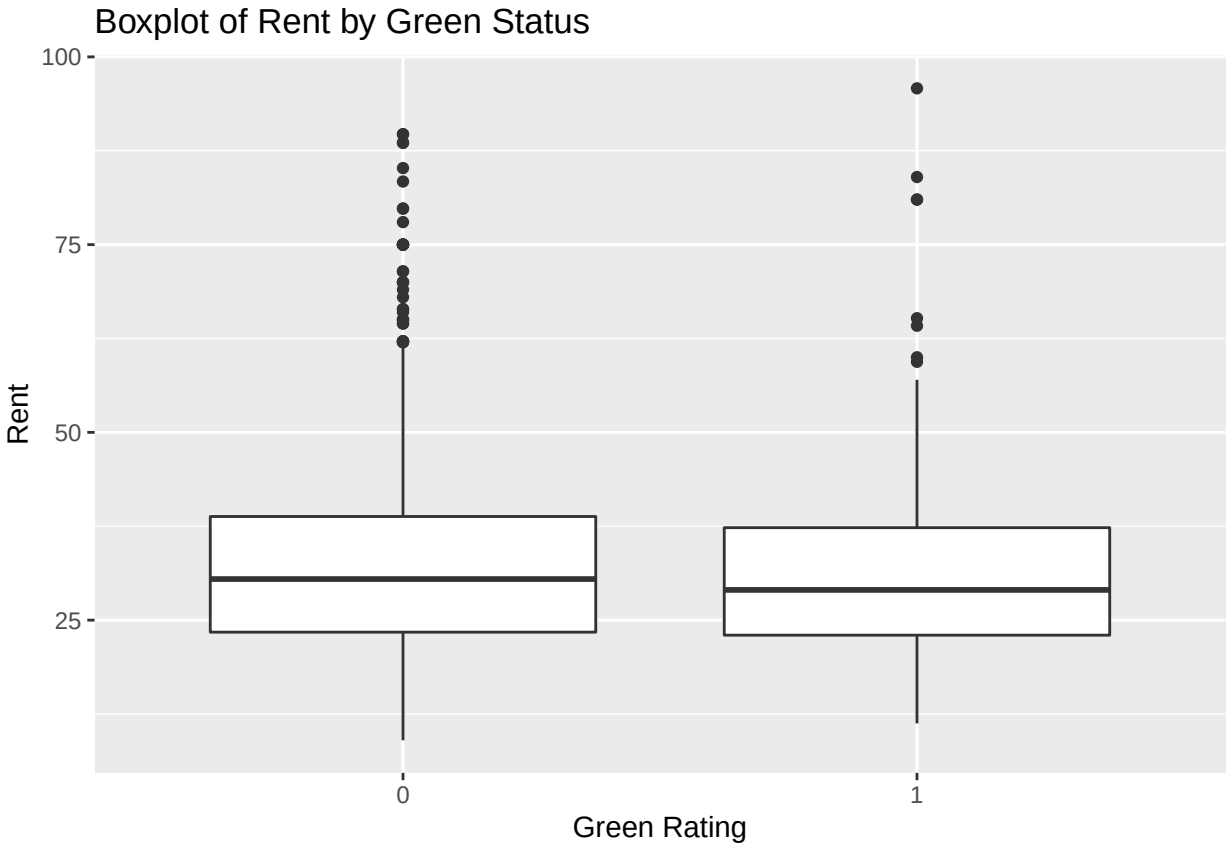
```
#subset the first through the third quartile in sizes
data_size <- subset(green_classA, green_classA$size > 50891 & green_classA$size < 294212)

box <- ggplot(data_size,aes(x = as.factor(green_rating), y = leasing_rate))

box + geom_boxplot() + ggtitle('Boxplot of Leasing Rate by Green Status') + ylab('Leasing Rate') +
  xlab('Green Rating')
```



The median leasing rate for Class A non-green buildings is 91% and the median for green buildings is about 94%. This is a marginal difference between the two.

Let's look at the difference in rent prices.

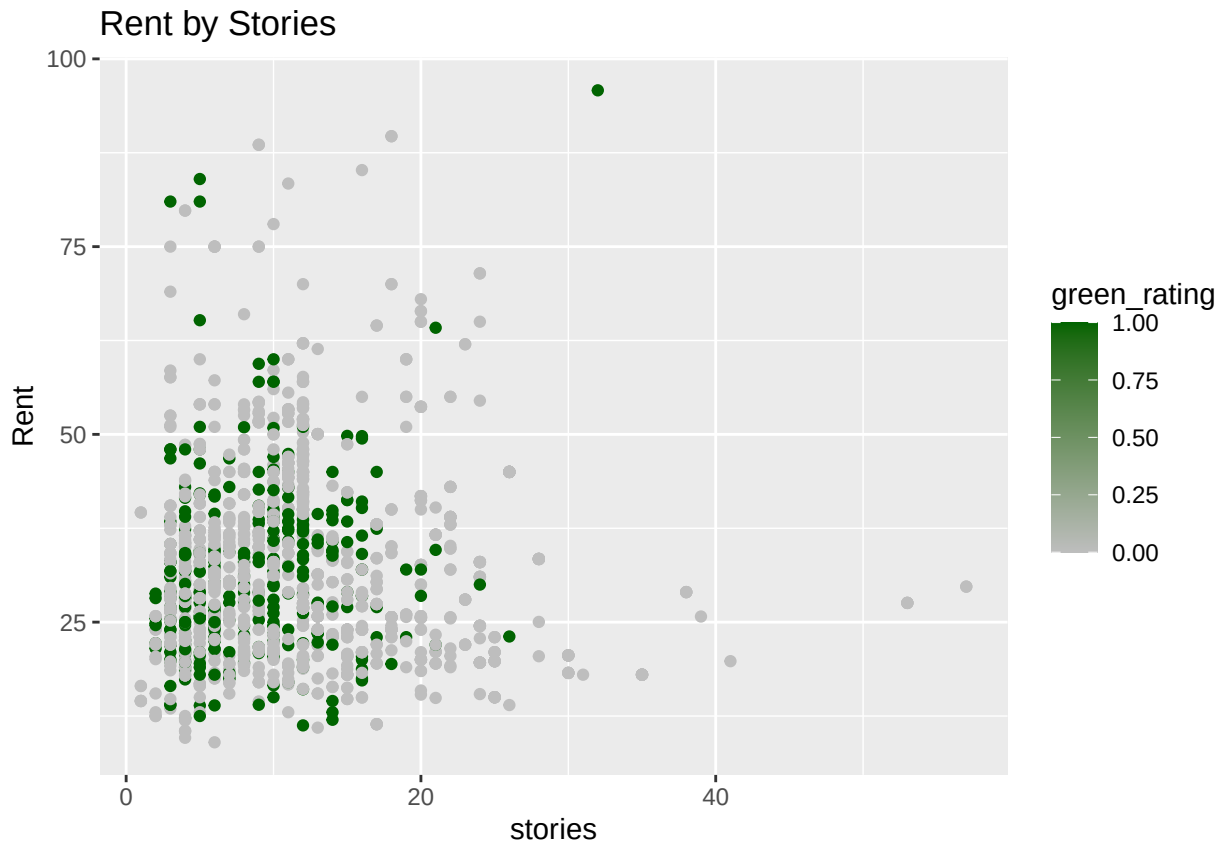
```
box <- ggplot(data_size,aes(x = as.factor(green_rating), y = Rent))

box + geom_boxplot() + ggtitle('Boxplot of Rent by Green Status') + ylab('Rent') +
  xlab('Green Rating')
```



The median rent price for Class A non-green buildings is $30.47 per square foot and the median for green buildings is $29.03 square foot. Rent for green buildings is actually cheaper than for green buildings, contrary to the stats guru's analysis.

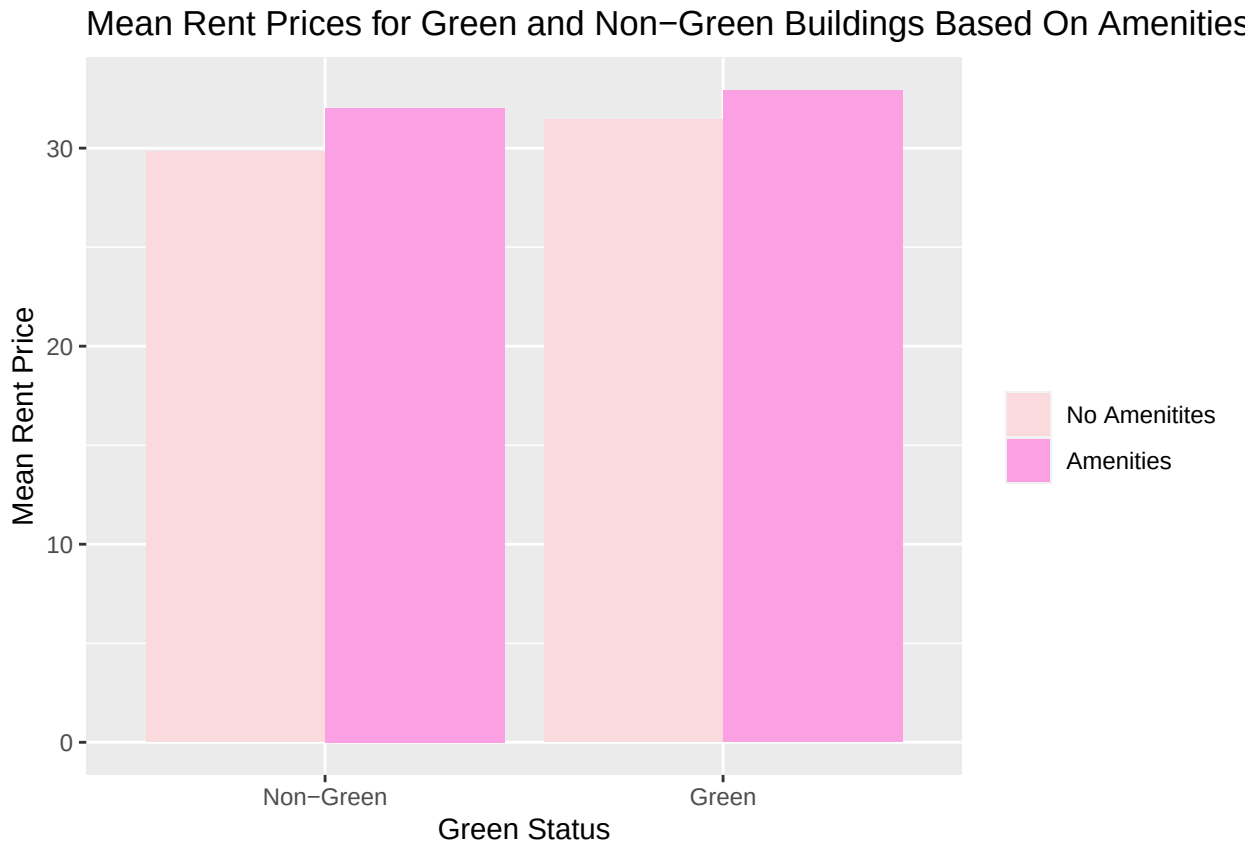
Through this analysis, we can conclude that the stats guru's analysis may be incorrect. Let's explore possible confounding variables.

```
ggplot(data = data_size, aes(x = stories, y = Rent, group = green_rating)) + geom_point(aes(color = gre
```



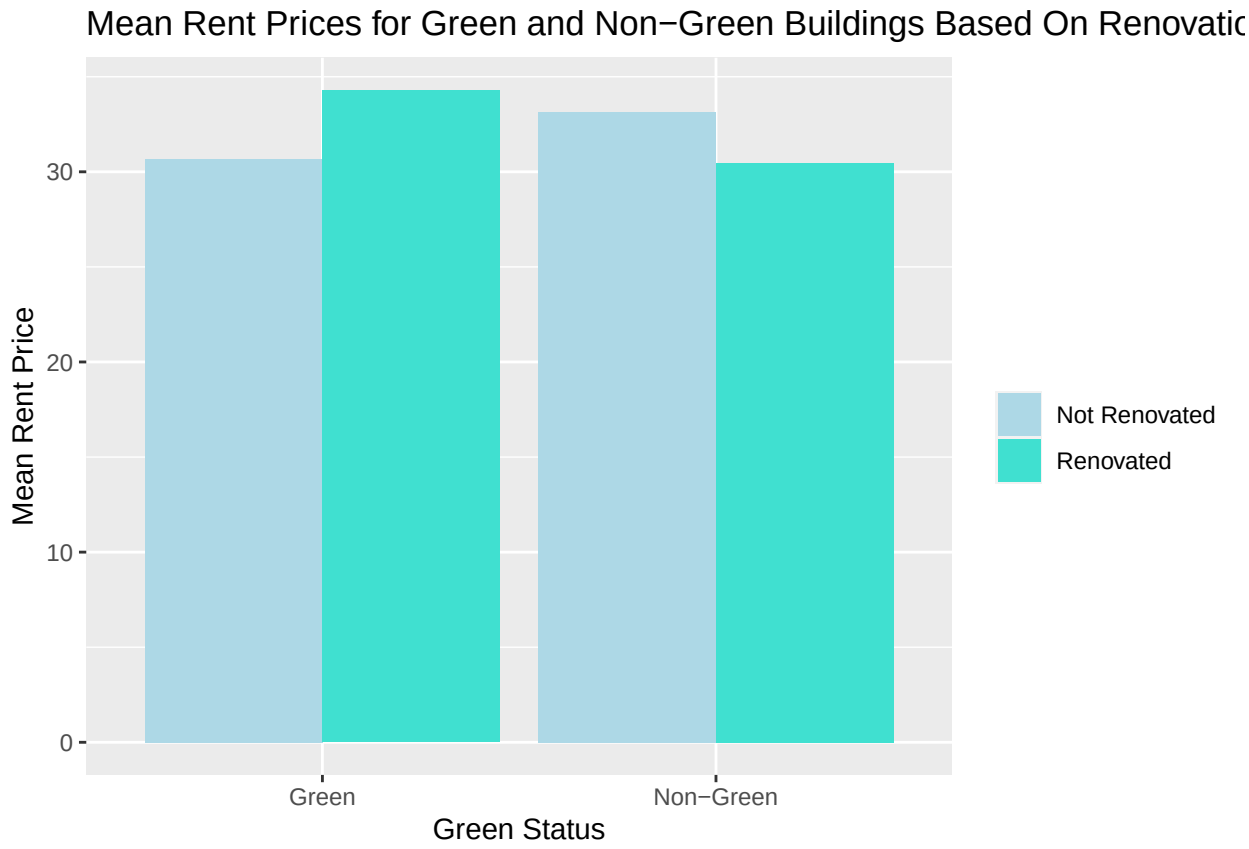
First we tried stories of the building. From this scatter plot, we can't conclude whether there is a significant difference in stories between green and non green buildings nor their rent prices.

```
mean_rent <- aggregate(data_size$Rent, by = list(id1 = data_size$green_rating, id2 = data_size$amenitie

mean_rent$green <- ifelse(mean_rent$id1 == 0, 'Non-Green', 'Green')

g <- ggplot(mean_rent,aes(x = green, y = x, fill = as.factor(id2)))

g + geom_bar(position = 'dodge', stat = 'identity')+
  scale_fill_manual("id2",labels = c('No Amenitites', 'Amenities'), values = c('#fadadd', '#fba0e3')) +
  xlab('Green Status') + ylab('Mean Rent Price') + ggtitle('Mean Rent Prices for Green and Non-Green Bu
  guides(fill=guide_legend(title="")) + scale_x_discrete(labels = c('Non-Green', 'Green'))
```



We can see that the presence of amenities affect rent prices in both green and non-green buildings similarly.

```
mean_rent <- aggregate(data_size$Rent, by = list(id1 = data_size$green_rating, id2 = data_size$renovate

mean_rent$green <- ifelse(mean_rent$id1 == 0, 'Non-Green', 'Green')

g <- ggplot(mean_rent,aes(x = green, y = x, fill = as.factor(id2)))

g + geom_bar(position = 'dodge', stat = 'identity')+
  scale_fill_manual("id2",labels = c('Not Renovated', 'Renovated'), values = c('lightblue', '#40E0D0'))
  xlab('Green Status') + ylab('Mean Rent Price') + ggtitle('Mean Rent Prices for Green and Non-Green Bu
  guides(fill=guide_legend(title=""))
```



We can see that the mean rent price of renovated non-green buildings is lower, while it is the opposite case for green buildings. This is a possible confounding variable, as the renovation status of a non-green building brings down the mean rent price and vice versa for green buildings. 25% of non-green Class A buildings are renovated and 20% of green Class A buildings are renovated, so this is a possibility.

2. ABIA Visual Storytelling

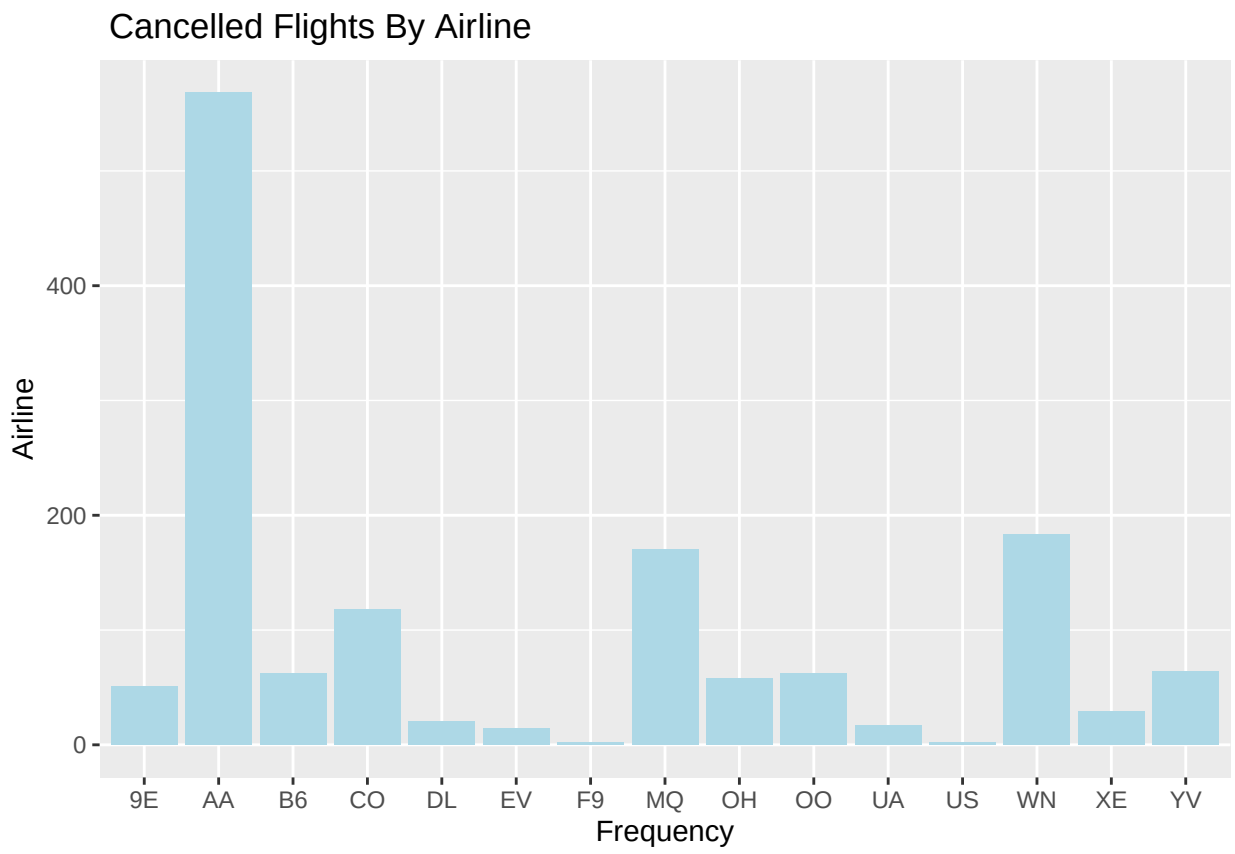
Load in libraries and data

```
library(dplyr)
library(ggplot2)
library(maps)
library(usmap)
library(mosaic)
library(treemap)


abia <- read.csv("ABIA.csv")
```

How many flights were canceled in total by airline?

```
cancel <- abia %>% filter(abia$Cancelled == 1)

total_bar <- ggplot(cancel, aes(UniqueCarrier, Cancelled))
total_bar + geom_bar(stat = 'identity', fill  = "lightblue") +
```

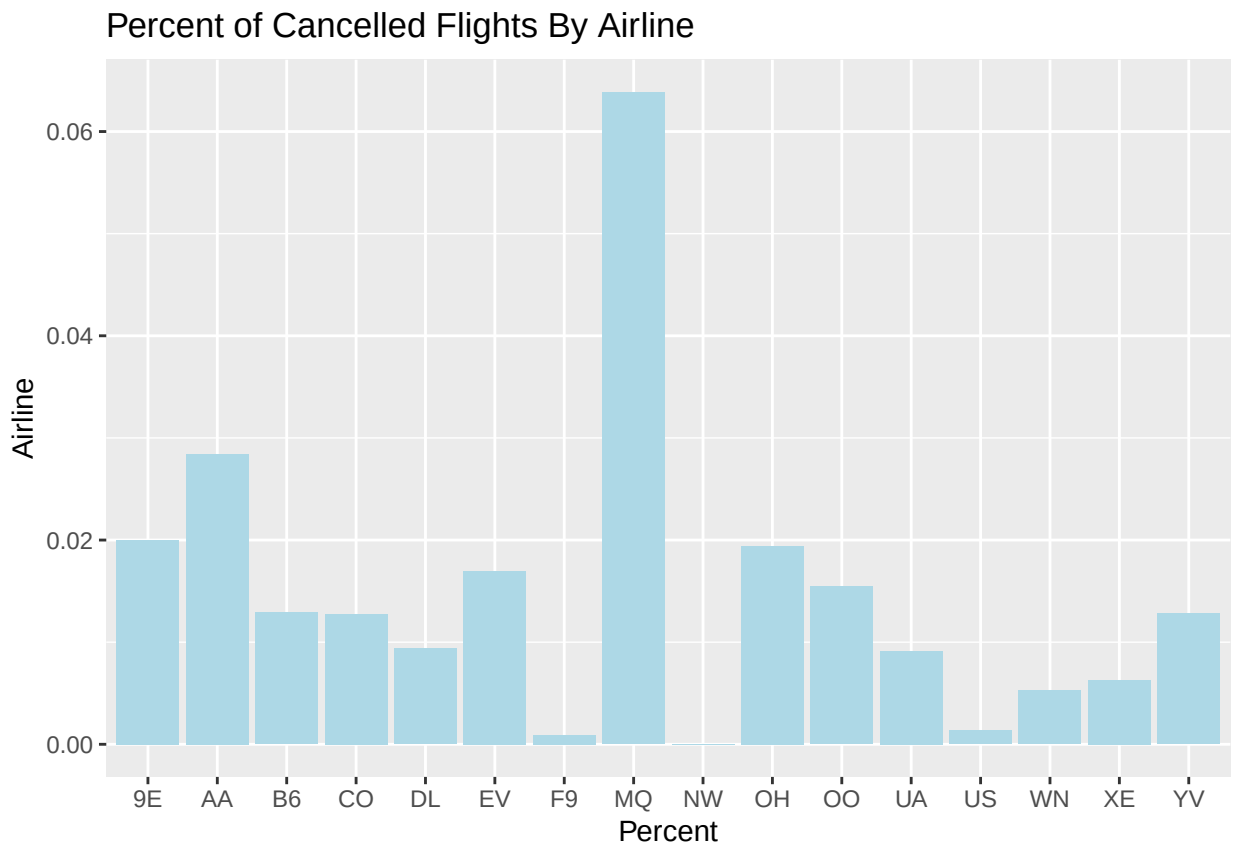
```
  ggtitle(' Cancelled Flights By Airline') +
  xlab("Frequency") +
  ylab('Airline')
```



We can see that American Airlines had the most total flights canceled.

What percentage of flights were canceled by airline?

```
total_carrier_counts <- aggregate(abia$UniqueCarrier, by=list(abia$UniqueCarrier), FUN=length)

counts <- table(cancel$UniqueCarrier)

counts <- t(t(counts))

counts <- append(counts, 0, after = 8)

percentage <- counts/total_carrier_counts['x']

total_carrier_counts['x'] <- percentage

canceled_bar <- ggplot(total_carrier_counts, aes(Group.1, x))
canceled_bar + geom_bar(stat = 'identity', fill  = "lightblue") +
  ggtitle('Percent of Cancelled Flights By Airline') +
  xlab("Percent") +
  ylab('Airline')
```

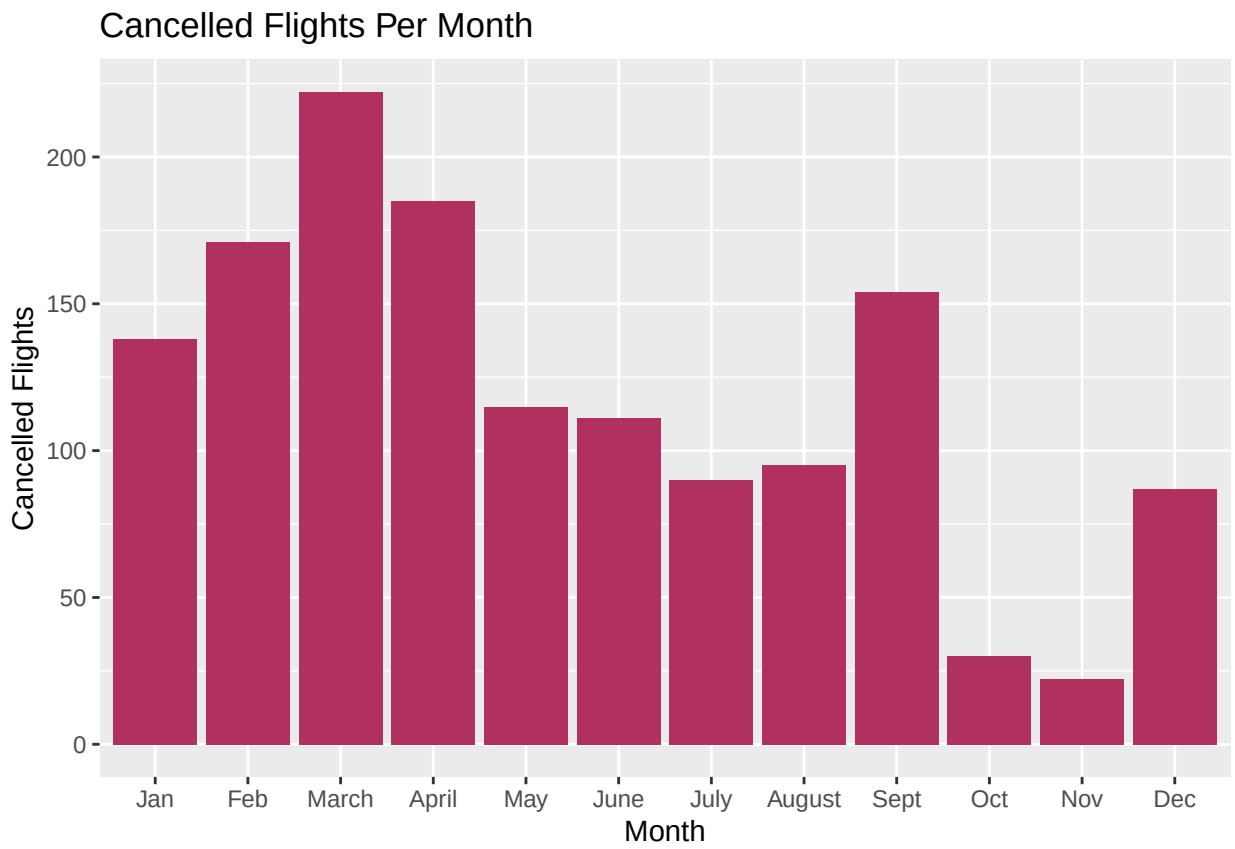


From this graph, we can see that only about 3% of American Airlines flights have been canceled. MQ, by far, has the highest percentage of cancellations, with 6.3% of flights canceled. Also of note, NW never had a cancellation but only flew 121 flights, 99% less than the most flown airline.

How many flights were canceled in each month?

```
ggplot(data=abia, aes(x=Month, y=Cancelled, fill = Month)) +
  geom_bar(stat="identity", fill = 'maroon')+
  scale_x_discrete(limit = c("Jan", "Feb", "March", "April", "May", "June", "July", "August",
                             "Sept", "Oct", "Nov", "Dec")) +
  ggtitle('Cancelled Flights Per Month') +
  xlab('Month') +
  ylab('Cancelled Flights') + guides(fill = FALSE)
```

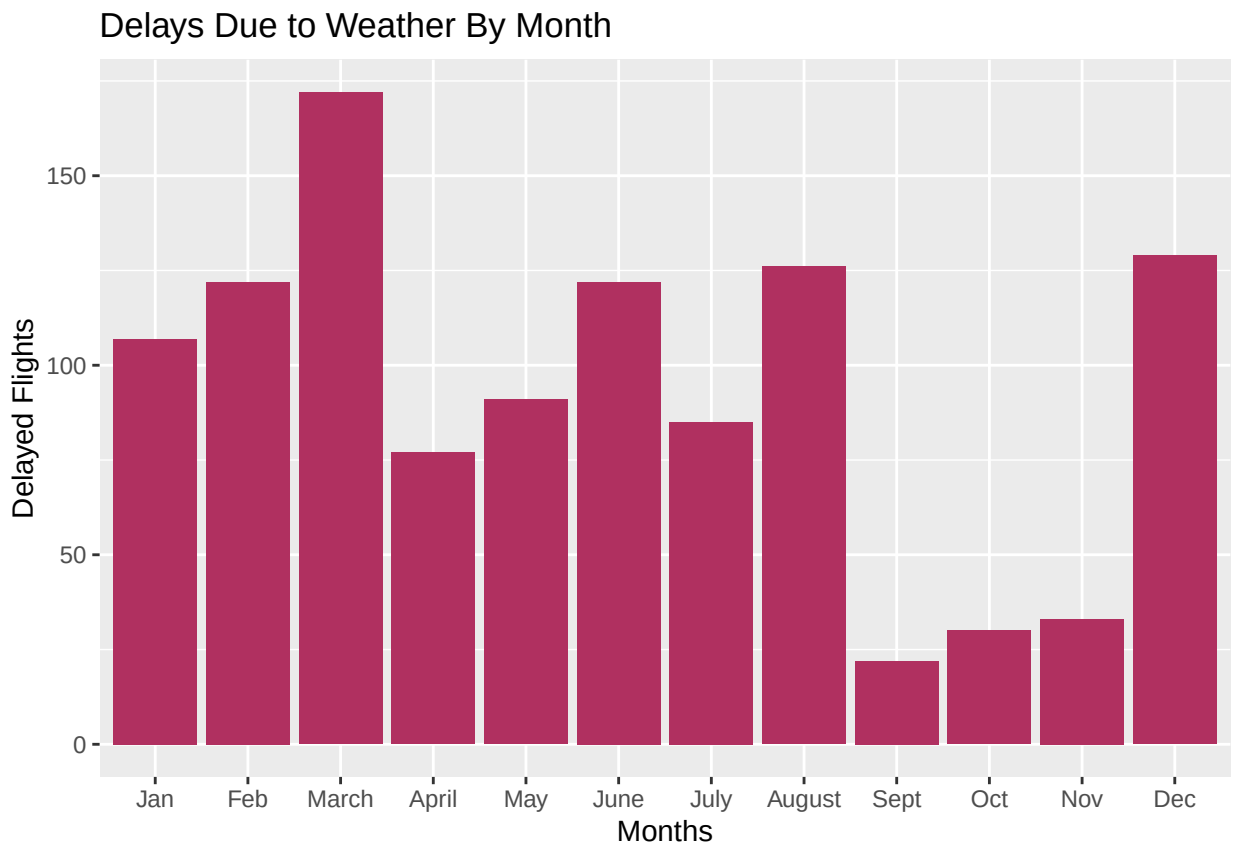
```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



Most cancellations occurred in March and April. Our team expected most cancellations would occur due to weather in December and January.

To further explore this, let's look at weather delays per month.

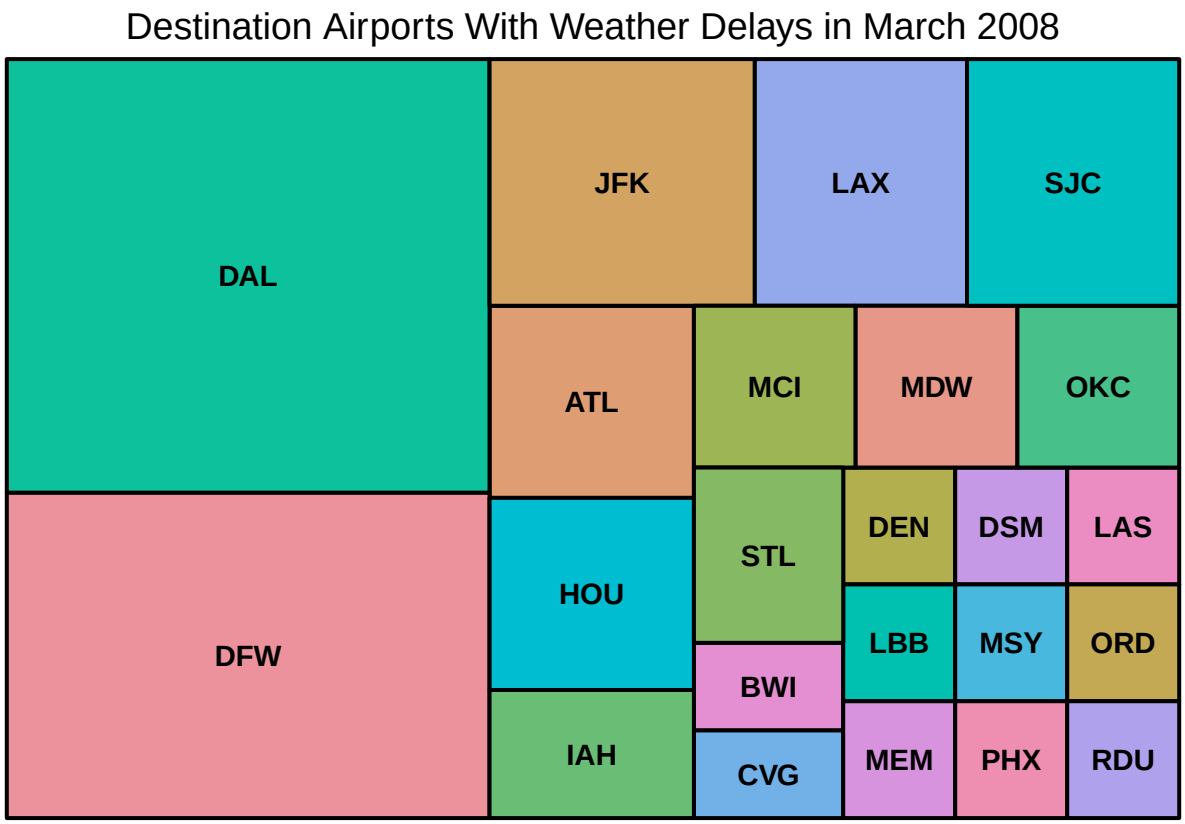
```
delayed <- abia %>% filter(abia$WeatherDelay > 0)
delayed <- aggregate(delayed$WeatherDelay, by=list(delayed$Month), FUN=length)
delayed_plot <- ggplot(delayed, aes(Group.1, x))
delayed_plot + geom_bar(stat = "identity", fill = 'maroon') +
  scale_x_discrete(limit = c("Jan", "Feb", "March", "April", "May", "June", "July", "August",
                             "Sept", "Oct", "Nov", "Dec")) +
  xlab("Months") + ylab('Delayed Flights') + ggtitle('Delays Due to Weather By Month')
```



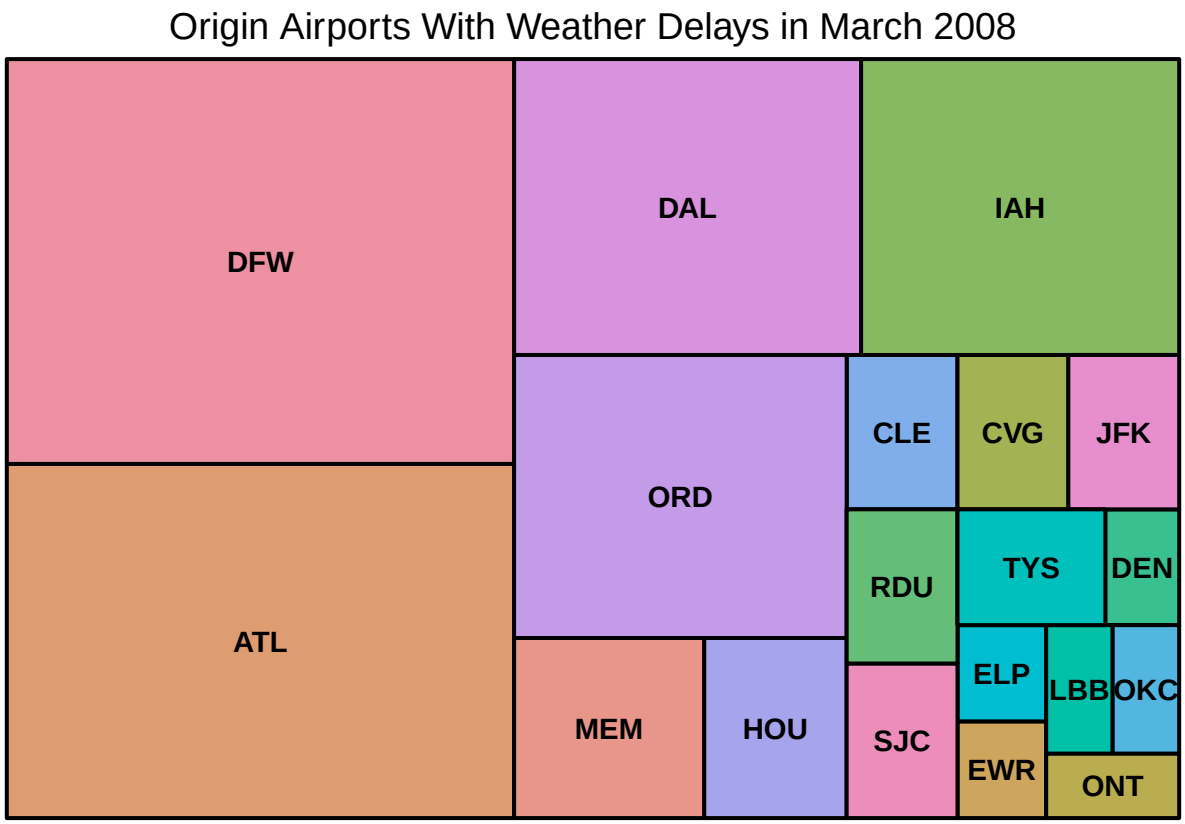
The shape of this bar chart is very similar to cancellations by month. Our team hypothesized that perhaps there was a significant weather event in Austin or a destination airport in March that would increase the weather delays and cancellations.

```
march_delays_origin <- abia %>% filter(Origin == 'AUS' & Month == 3 & WeatherDelay > 0)

march_delays_dest <- abia %>% filter(Dest == 'AUS' & Month == 3 & WeatherDelay > 0)

destinations <- aggregate(march_delays_origin$Dest, by=list(march_delays_origin$Dest), FUN=length)

origins <- aggregate(march_delays_dest$Origin, by=list(march_delays_dest$Origin), FUN=length)
```

```
treemap(destinations, index = "Group.1", vSize = "x", type = "index",
        title = 'Destination Airports With Weather Delays in March 2008')
```




```
treemap(origins, index = "Group.1", vSize = "x", type = "index",
        title = 'Origin Airports With Weather Delays in March 2008')
```



Austin is the destination airport in 69% of delays, indicating that Austin is not the cause of most weather delays. Additionally, in March, there were a high amount of delays from ATL and ORD as the origin and not many in which they were the destination, indicating that there may have been a weather event there in the month of March.

Let's look at cancellations and significant weather delays together. A significant weather delay can be defined as one that is more than 123 minutes, or above the third quartile.

```
significant <- abia %>% filter(abia$WeatherDelay> 123 | abia$Cancelled==1)
```

Here, we get the airport codes and coordinates for each destination.

```
airport_codes <- read.csv('airport-codes.csv')
airports <- airport_codes %>% filter(airport_codes$type == "large_airport")
```

First, let's look at these significant delays and cancellations when Austin is the origin airport.

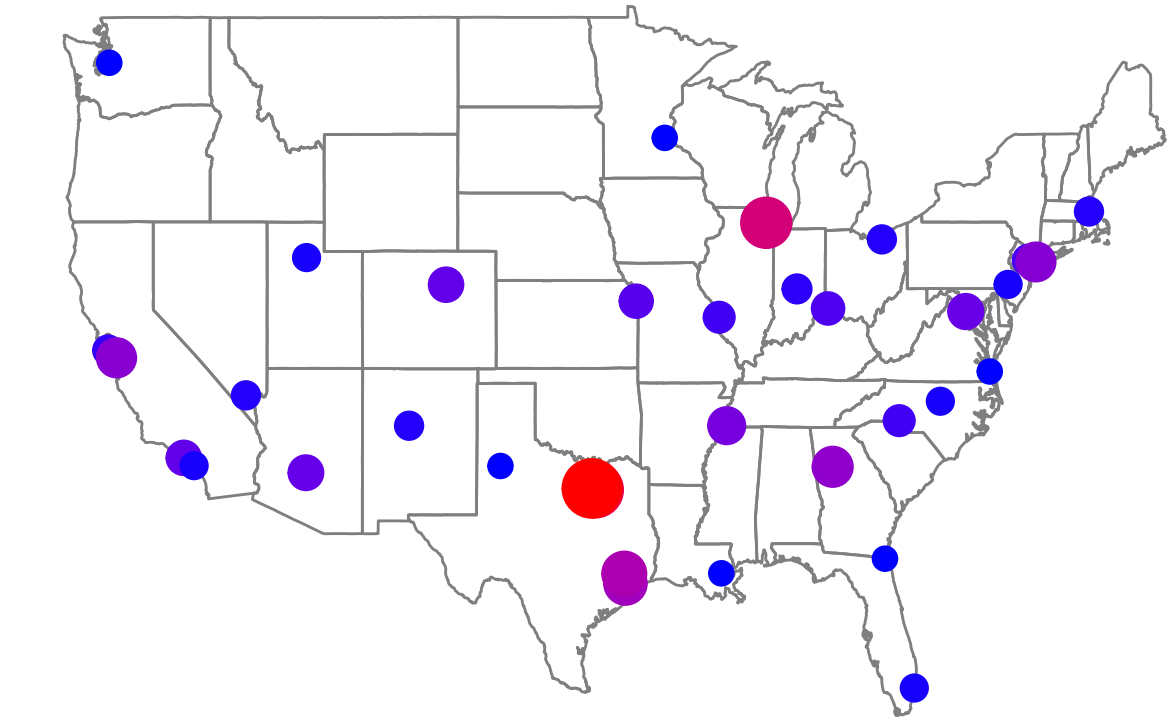
```
significant <- significant %>% filter(significant$Origin == 'AUS')

string <- significant$Origin
significant$Origin <- paste0("K", string)

string <- significant$Dest
significant$Dest <- paste0("K", string)
```

Get the latitudes and longitudes for the airports

```
merged_airports <- merge(significant, airports, by.x='Dest', by.y='ident')

merged_count <- aggregate(merged_airports$Dest, by=list(merged_airports$Dest), FUN=length)

merged_airports <- tidyr::separate(merged_airports, coordinates, into = c("long", "lat"), sep = ",")

merged_airports$long <- as.numeric(merged_airports$long)
merged_airports$lat <- as.numeric(merged_airports$lat)

map_data <- merged_airports %>% select(lat, long, Dest)
map_data <- unique(map_data)

df2 <- merge(merged_count, map_data, by.x='Group.1', by.y='Dest')
```

```
usmap <- borders("state")
ggplot() + usmap +
  geom_point(data = df2, aes(x = lat, y = long, size = x, colour =x)) + scale_size(range = c(4,10)) +
scale_color_gradient(low = "blue", high = "red") +
  theme(panel.background = element_rect(fill = "white"),
        axis.line = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position =  'none'
  ) +
  labs(title = "ABIA Departure Delays and Cancellations") + guides(fille = FALSE)
```

```
## Warning: ‘guides(<scale> = FALSE)‘ is deprecated. Please use ‘guides(<scale> =
## "none")‘ instead.
```

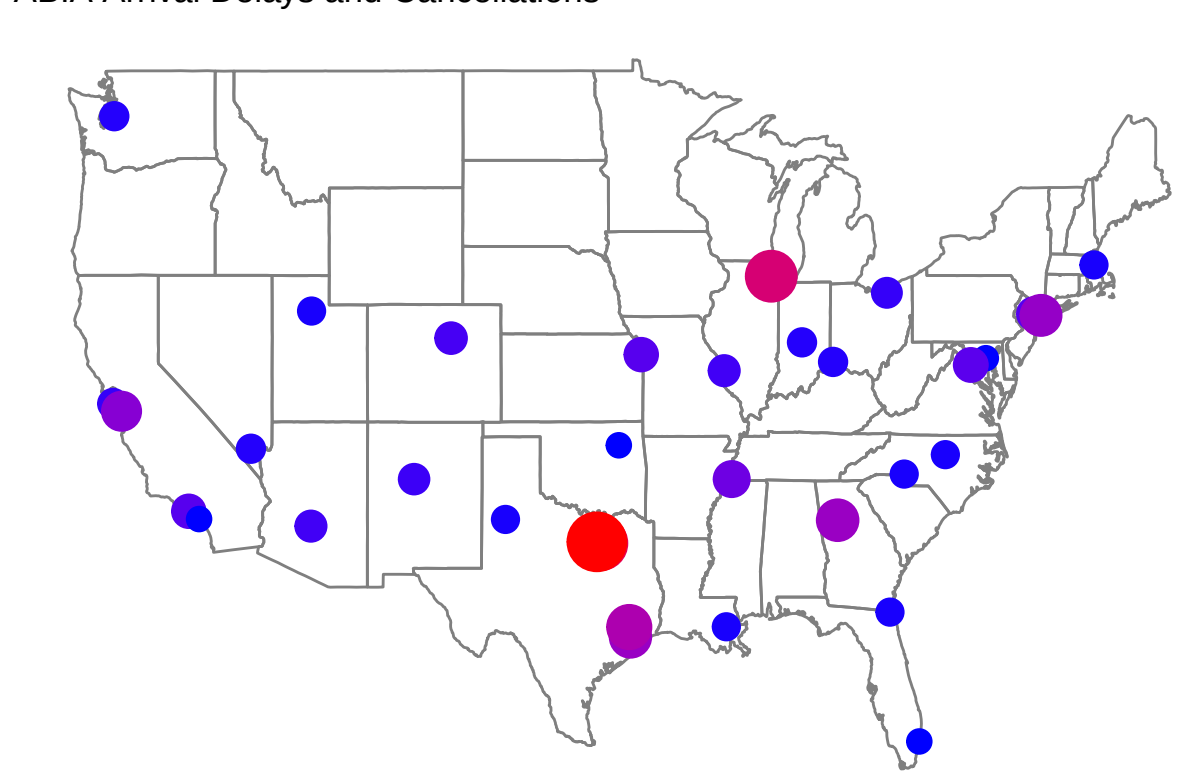


In this graph, we can visually see that the hotspots for delays out of Austin are for Dallas and secondly for Chicago. There are more delays for east coast cities compared to west coast cities.

```
significant <- abia %>% filter(abia$WeatherDelay> 123 | abia$Cancelled==1)

significant <- significant %>% filter(significant$Dest == 'AUS')

string <- significant$Origin
significant$Origin <- paste0("K", string)

string <- significant$Dest
significant$Dest <- paste0("K", string)

merged_airports <- merge(significant, airports, by.x='Origin', by.y='ident')

merged_count <- aggregate(merged_airports$Origin, by=list(merged_airports$Origin), FUN=length)

merged_airports <- tidyr::separate(merged_airports, coordinates, into = c("long", "lat"), sep = ",")

merged_airports$long <- as.numeric(merged_airports$long)
merged_airports$lat <- as.numeric(merged_airports$lat)

map_data <- merged_airports %>% select(lat, long, Origin)
map_data <- unique(map_data)

df2 <- merge(merged_count, map_data, by.x='Group.1', by.y='Origin')
```

```
usmap <- borders("state")
ggplot() + usmap +
  geom_point(data = df2, aes(x = lat, y = long, size = x, colour =x)) + scale_size(range = c(4,10)) +
scale_color_gradient(low = "blue", high = "red") +
  theme(panel.background = element_rect(fill = "white"),
        axis.line = element_blank(),
        axis.text.x = element_blank(),
        axis.text.y = element_blank(),
        axis.ticks = element_blank(),
        axis.title.x = element_blank(),
        axis.title.y = element_blank(),
        legend.position =  'none'
  ) +
  labs(title = "ABIA Arrival Delays and Cancellations") + guides(fille = FALSE)
```

```
## Warning: `guides(<scale> = FALSE)` is deprecated. Please use `guides(<scale> =
## "none")` instead.
```



Looking at the significant delays and cancellations in which Austin is the destination airport, we can see that over the year, the cities that the most significant delays and cancellations came from were the same.

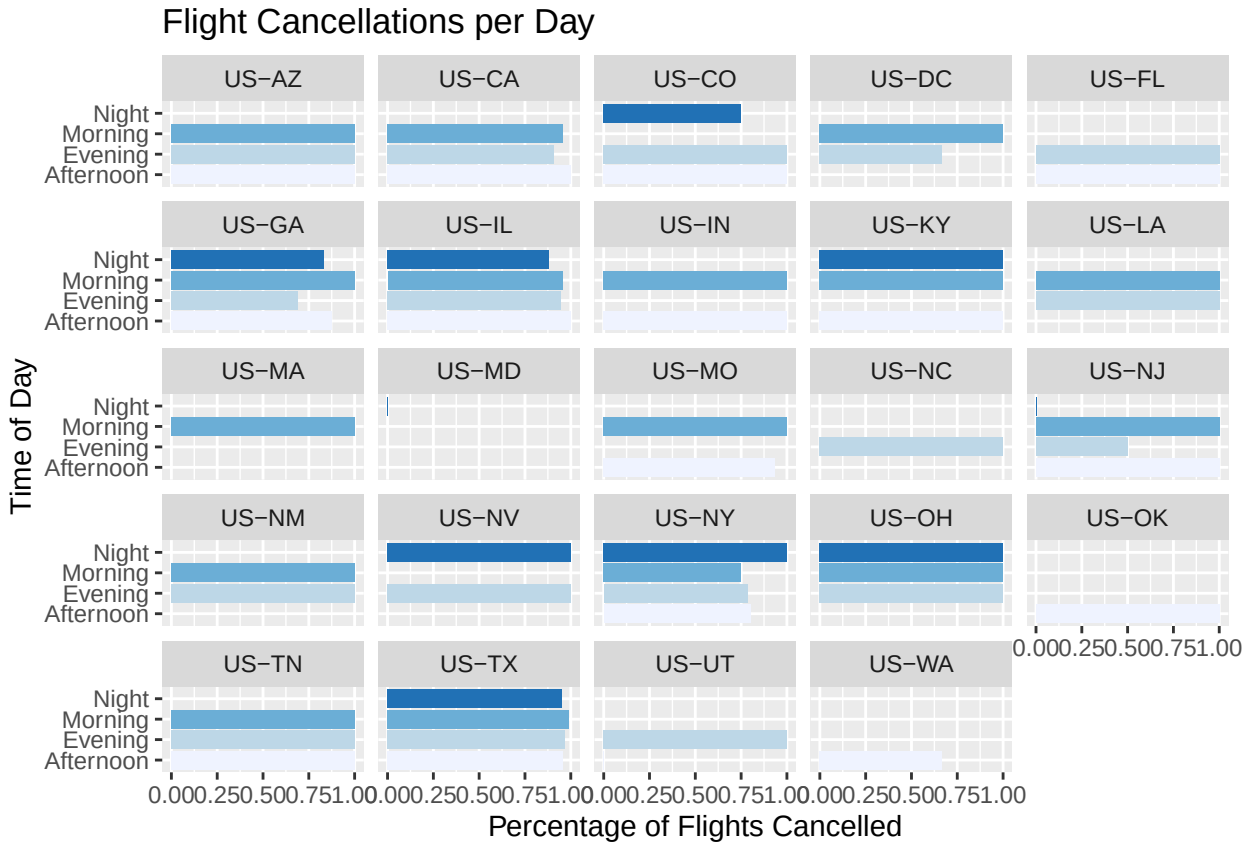
Finally, let's plot the percentage of cancellations by state and time of day.

```
merged_airports['DepTime'] = as.integer(merged_airports$CRSDepTime / 100)
merged_airports['TimeOfDay'] = ifelse(merged_airports$DepTime >=0 & merged_airports$DepTime < 12, 'Morn
                                      ifelse(merged_airports$DepTime >= 12 & merged_airports$DepTi
```

```
                                                    ifelse(merged_airports$DepTime >=14 & merged_airports$
merged_airports$DepTime= as.factor(merged_airports$DepTime)
merged_airports$TimeOfDay = as.factor(merged_airports$TimeOfDay)
group = merged_airports %>%
  group_by(TimeOfDay, iso_region)  %>%
  summarize(count = sum(Cancelled == 1) / sum(iso_region == iso_region))
```

```
## `summarise()` has grouped output by 'TimeOfDay'. You can override using the `.groups` argument.
```

```
ggplot(data = group, aes(x = TimeOfDay, y = count, fill = TimeOfDay)) +
  geom_bar(stat = 'identity') +
  facet_wrap(~iso_region) +
  scale_fill_brewer(4,palette = "Blues")+
  coord_flip()  +
  labs(title = "Flight Cancellations per Day") + xlab("Time of Day") + ylab("Percentage of Flights Canc
  theme(legend.position = "none")
```



For the primary hotspots of cancellations and delays, TX and IL, the time of day seems to not matter.

3. Portfolio Analysis

In this problem, we constructed three different portfolios of exchange-traded funds (ETFs) and used bootstrap resampling to analyze the short-term tail risk of our portfolios.

Our ETFs: (are diverse and offer differing levels of risk)

SPDR S&P 500 (SPY): A popular, safe, and diverse ETF. It is one of the largest on the market. It tracks the S&P 500 which measures the U.S. equity market and indicates the financial health and stability in the market. Average annual return of ~10%.

Vanguard Growth Index Fund (VUG): Is a growth ETF that focuses on low-risk, large-cap US-based growth stocks. It offers diversified exposure. The average 5-year return is about 22%.

Schwab US Small-Cap (SCHA): Offers exposure to small cap firms in the US equity market. Small caps historically greather annual returns and perform well with inflation. However recessions hit these small cap companies hard (or events like COVID-19 hurt).

Invesco QQQ (QQQ): A popular (non-financial company holding) ETF that tracks the NASDAQ 100 Index. About ~48% of its holdings are top technology companies. This ETF offers investors big rewards during bull markets and is good for long-term growth. The downside: it is comprised of large-cap stocks, is not diverse, and declines during bear markets and appears to be currently over valued. The avergae 5-year market return is roughly 27.62%

iShares Core MSCI Total International Stock (IXUS): Holds large, medium, and small cap non-US equities. It is a free-float adjusted market cap index that measures the equity performance of developed and emerging market countires. (generally safe and diversified)

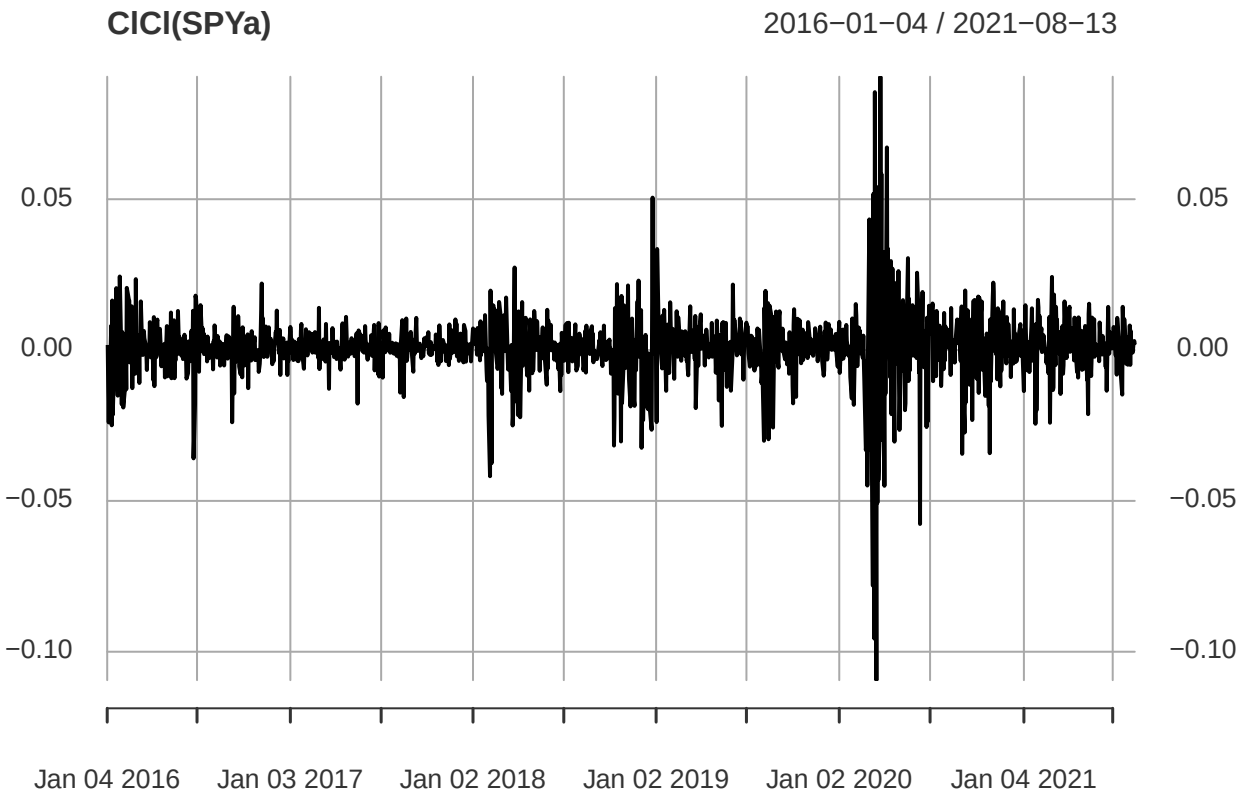
ProShares VIX ST Futures (VIXY): Offers exposure to short-term VIX futures. It is a high risk/high reward ETF because the performance is dependent on market volatility.

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

# Import a few ETFs
myetfs = c("SPY", "VUG", "SCHA", "QQQ", "IXUS", "VIXY" )

#get price data for past 5 years
getSymbols(myetfs, from = "2016-01-01")
```

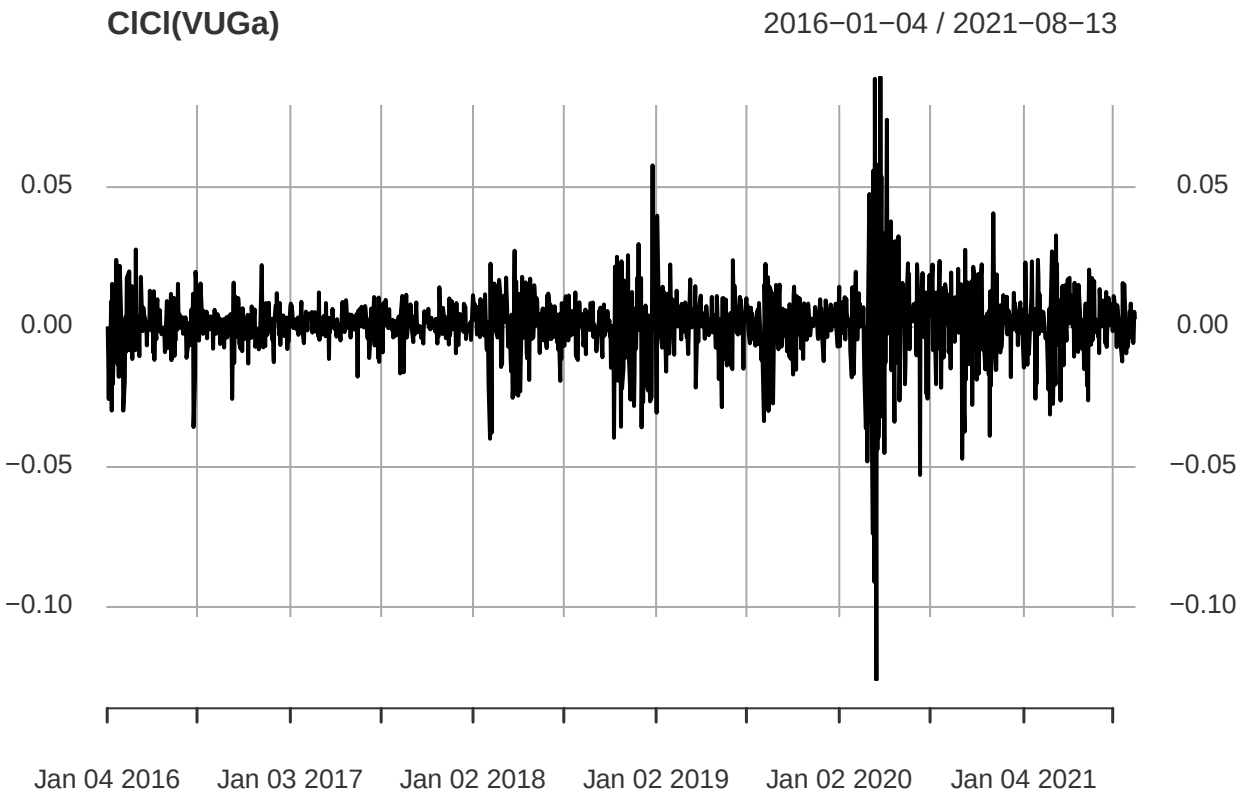
```
## [1] "SPY"  "VUG"  "SCHA" "QQQ"  "IXUS" "VIXY"
```

```
# Adjust for splits and dividends
SPYa = adjustOHLC(SPY)
VUGa = adjustOHLC(VUG)
SCHAa = adjustOHLC(SCHA)
QQQa = adjustOHLC(QQQ)
IXUSa = adjustOHLC(IXUS)
VIXYa = adjustOHLC(VIXY)
```

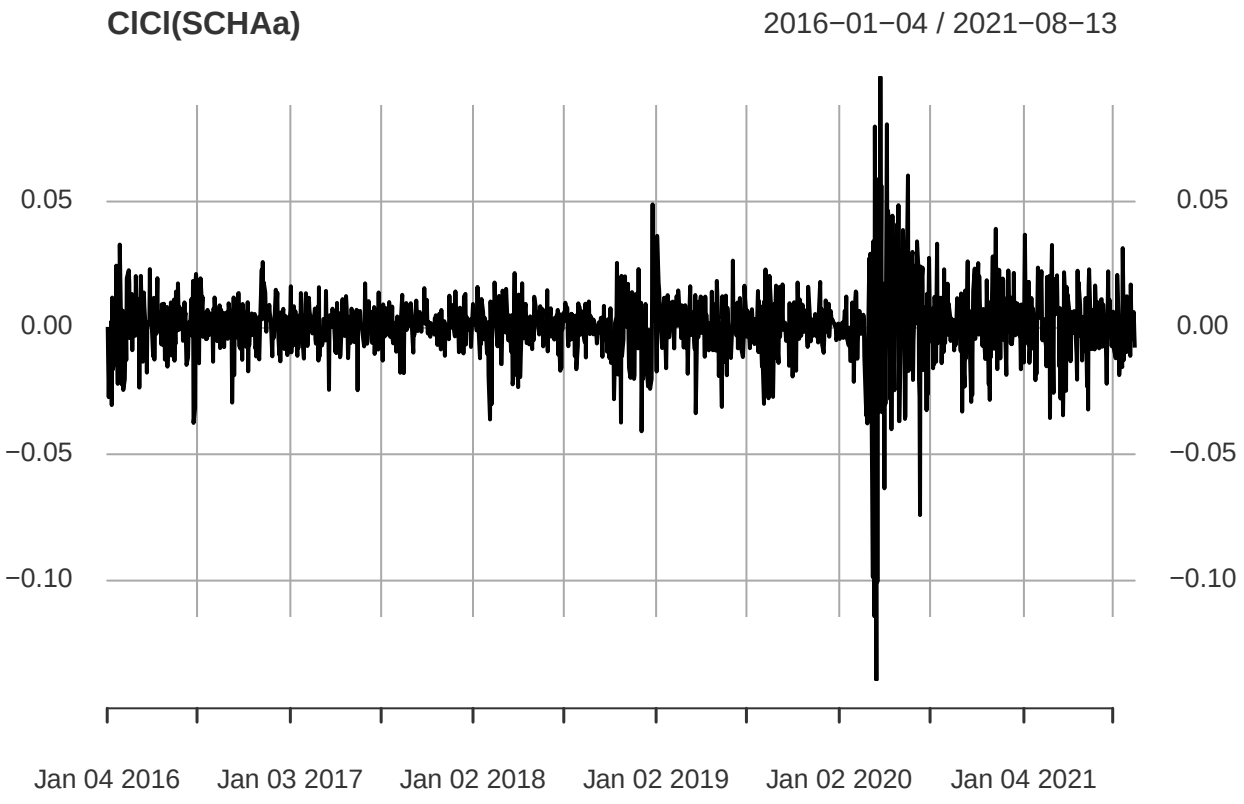
```
# Look at close-to-close changes
plot(ClCl(SPYa))
```



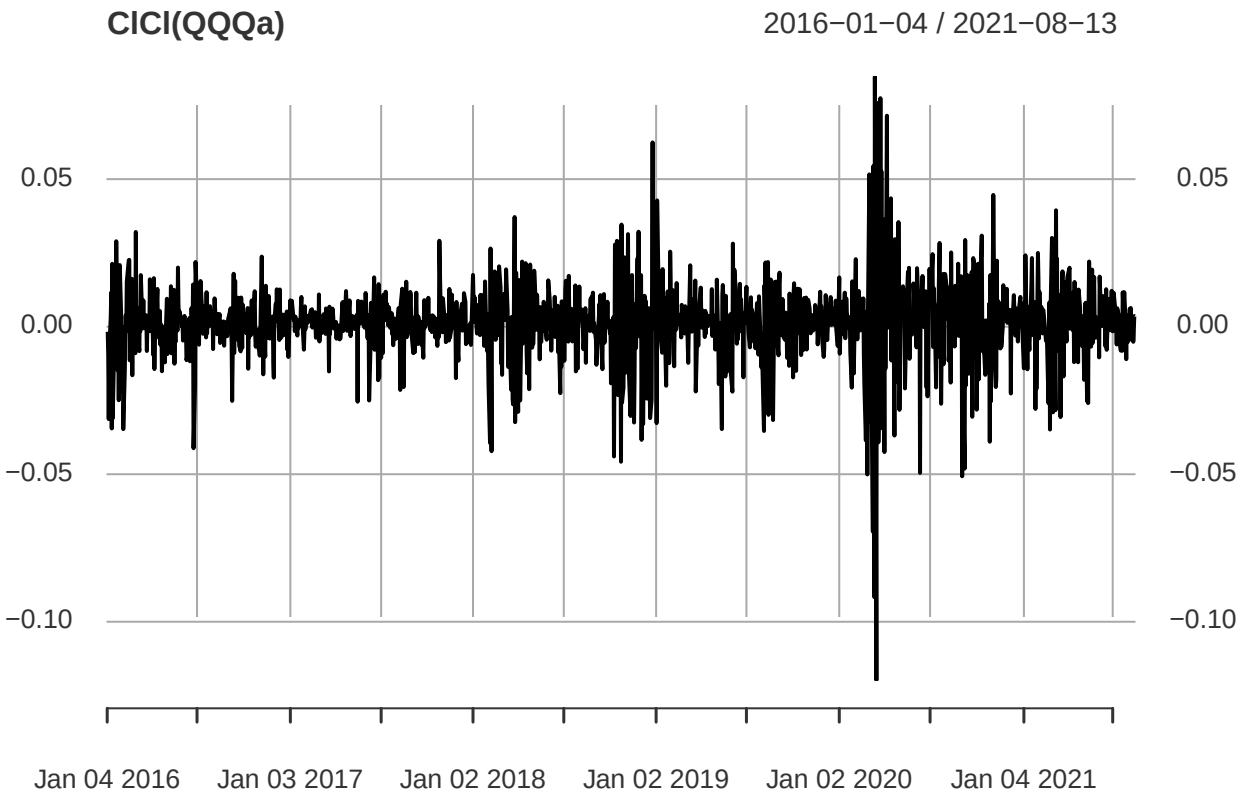

```
plot(ClCl(VUGa))
```



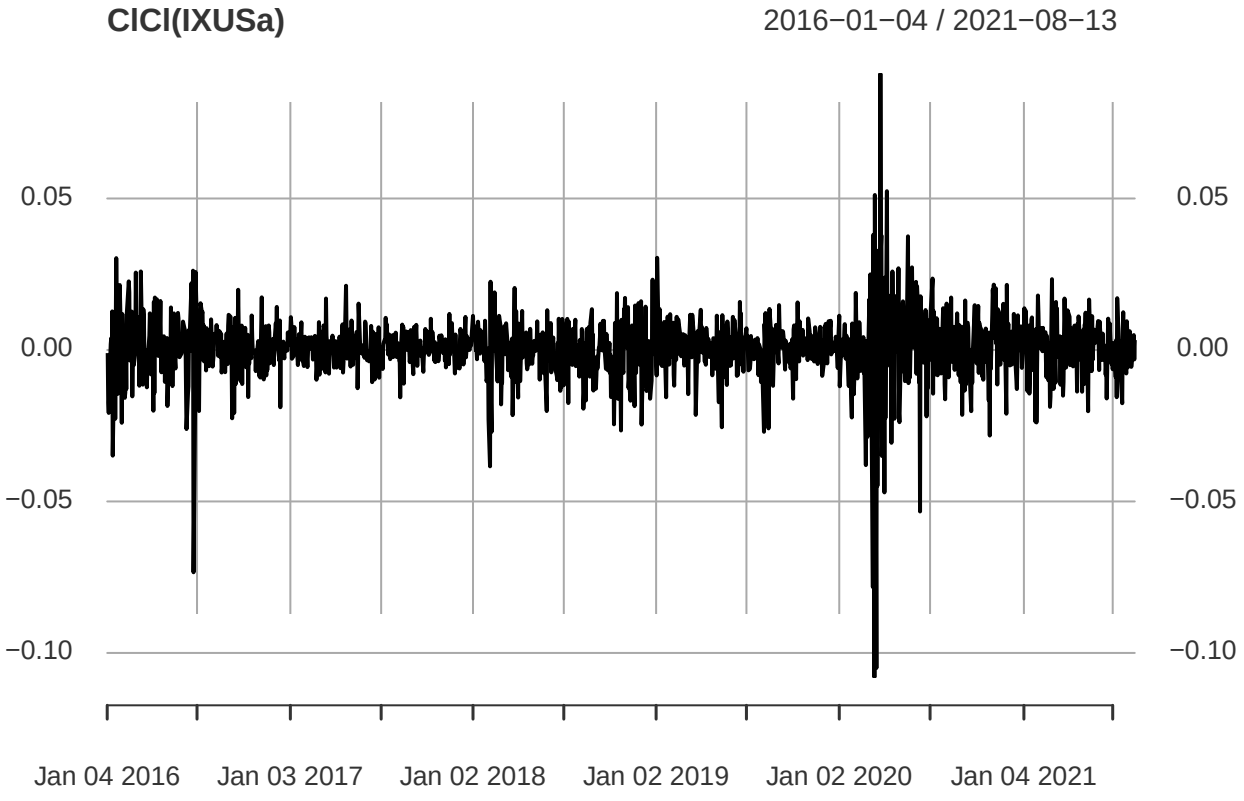

```
plot(ClCl(SCHAa))
```



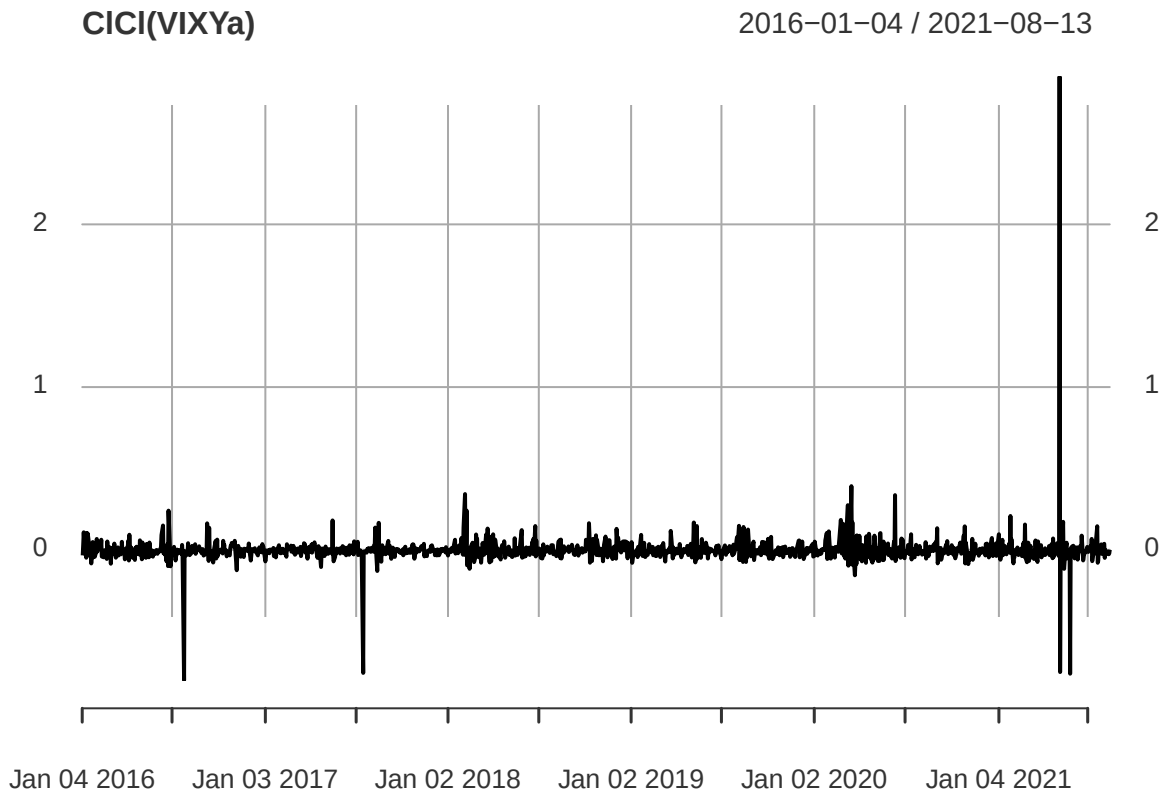

```
plot(ClCl(QQQa))
```




```
plot(ClCl(IXUSa))
```




```
plot(ClCl(VIXYa))
```



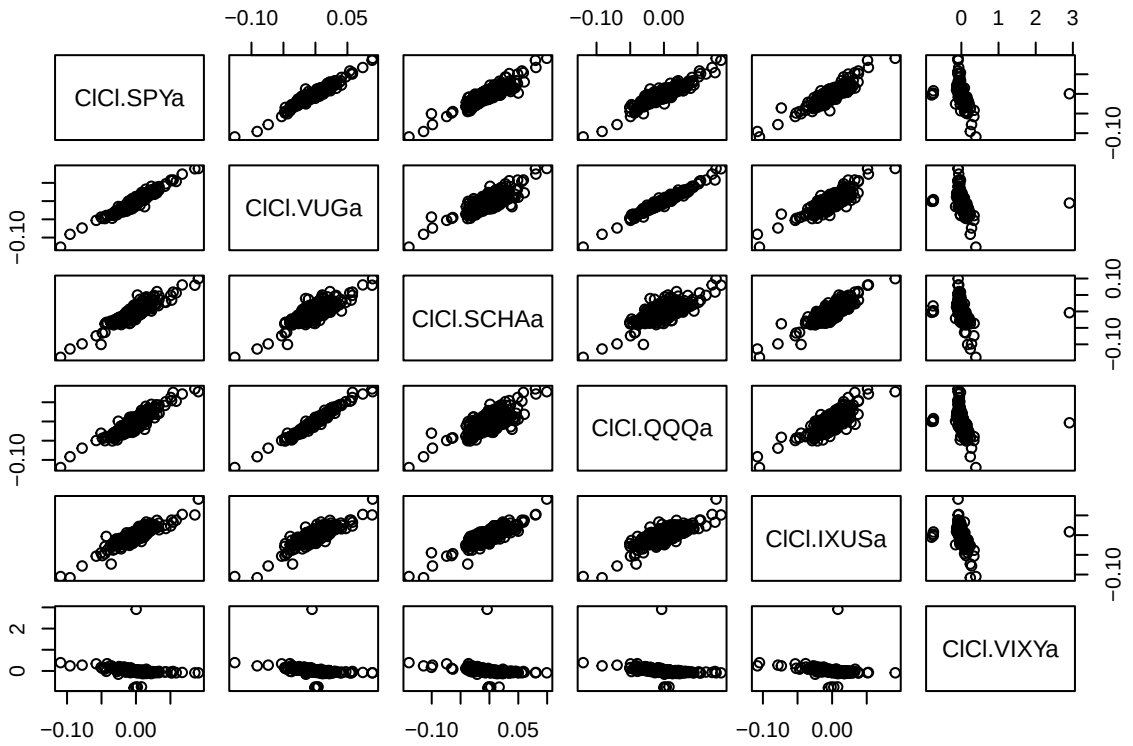
The plots shows the volatility of ETFs over the 5 year period.

```
# Combine close to close changes in a single matrix
all_returns = cbind(ClCl(SPYa),ClCl(VUGa),ClCl(SCHAa),ClCl(QQQa),
                    ClCl(IXUSa), ClCl(VIXYa))
head(all_returns)
```

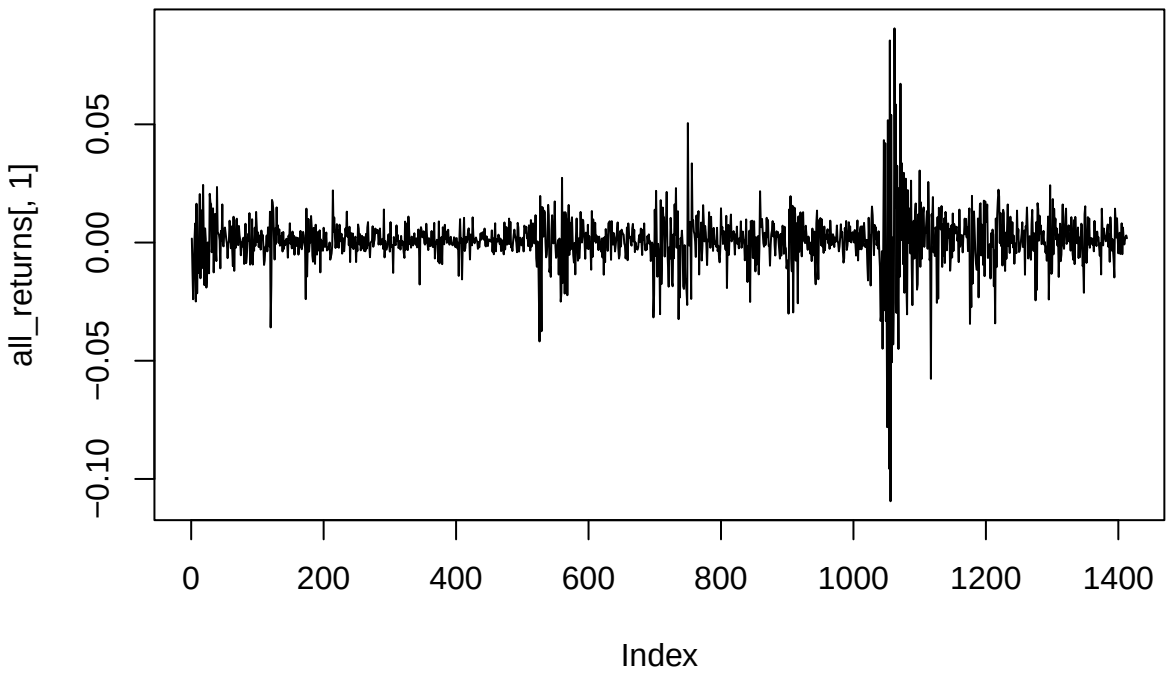
```
##                ClCl.SPYa     ClCl.VUGa    ClCl.SCHAa    ClCl.QQQa   ClCl.IXUSa
## 2016-01-04            NA            NA            NA           NA           NA
## 2016-01-05  0.0016913590  0.0001912212  0.0003914856 -0.001735178 -0.001030058
## 2016-01-06 -0.0126141934 -0.0109952581 -0.0146742318 -0.009605672 -0.017323159
## 2016-01-07 -0.0239915694 -0.0257154191 -0.0274027199 -0.031313495 -0.020776536
## 2016-01-08 -0.0109765780 -0.0093272773 -0.0149040425 -0.008200639 -0.012430304
## 2016-01-11  0.0009900115  0.0010016627 -0.0045596062  0.003076627  0.001953038
##             ClCl.VIXYa
## 2016-01-04          NA
## 2016-01-05 -0.03450700
## 2016-01-06  0.03282276
## 2016-01-07  0.10734461
## 2016-01-08  0.05165810
## 2016-01-11 -0.03153425
```

```
# first row is NA because we didn't have a "before" in our data
all_returns = as.matrix(na.omit(all_returns))
N = nrow(all_returns)
```

```
# Observe the returns correlations for the ETFs
pairs(all_returns)
```

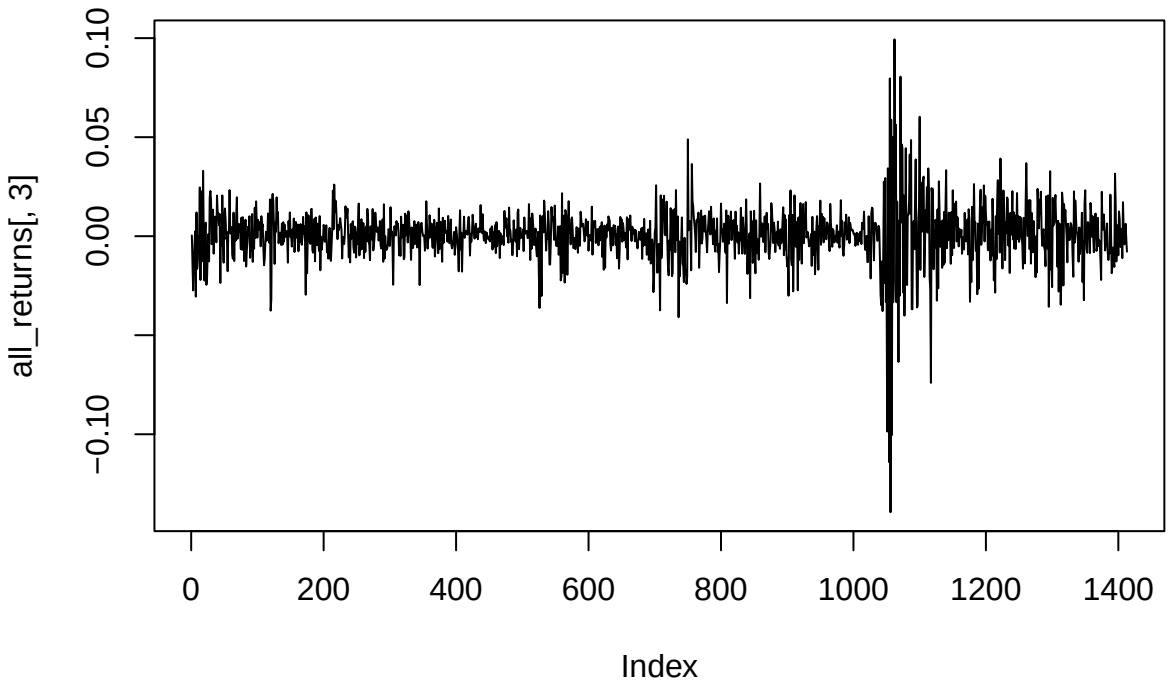



```
plot(all_returns[,1], type='l')
```

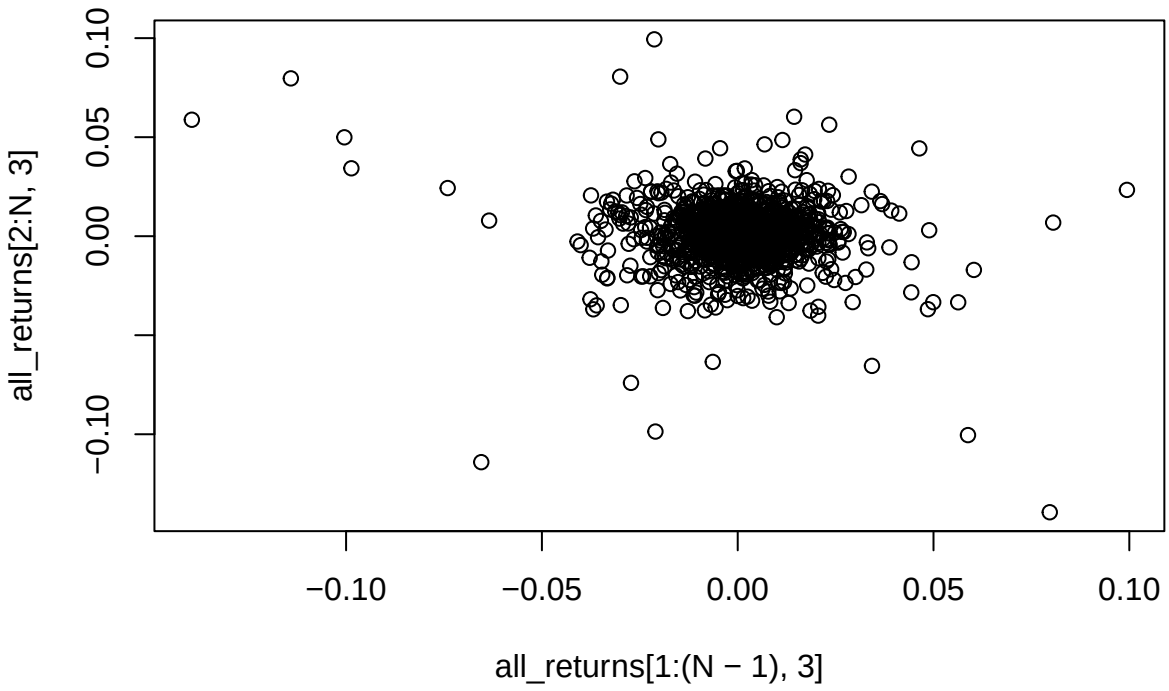


There are some strong correlations between the ETFs. Some are performing well, while others are not. Now, we look at the market returns over time.

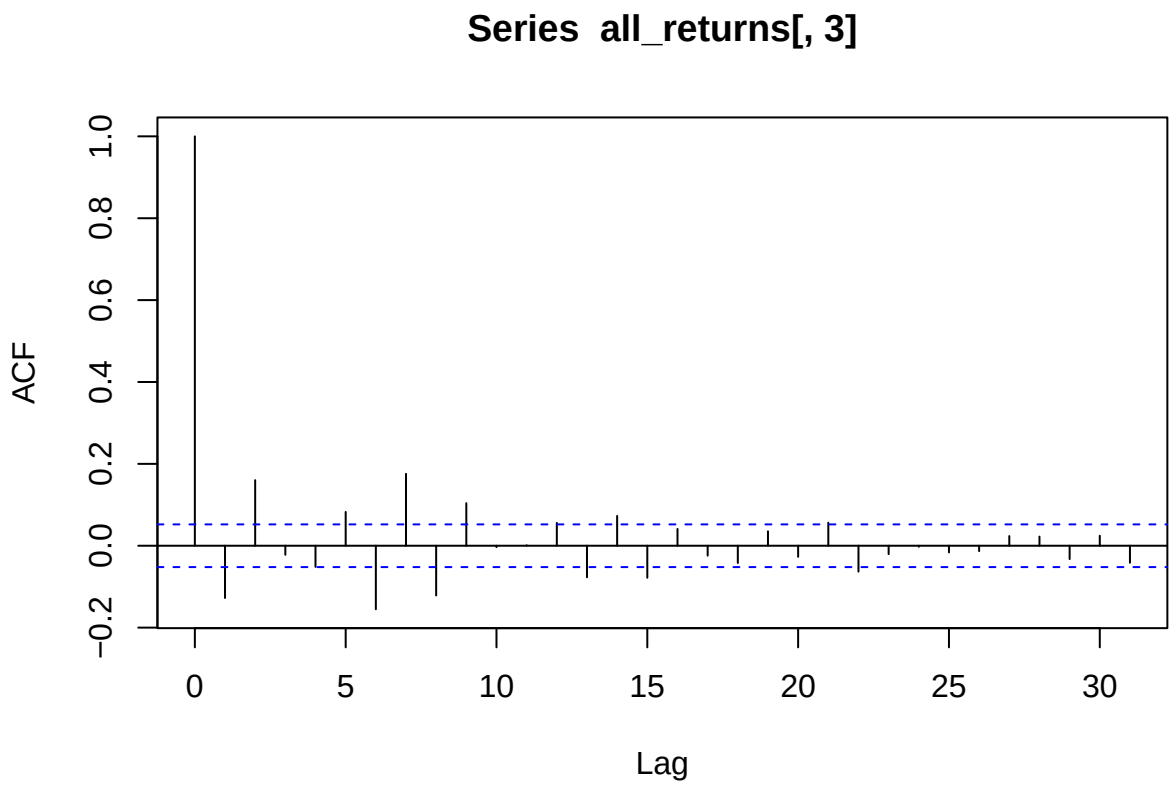
```
plot(all_returns[,3], type='l')
```




```
# are today's returns correlated with tomorrow's?
# not really!
plot(all_returns[1:(N-1),3], all_returns[2:N,3])
```




```
# An autocorrelation plot: nothing there
acf(all_returns[,3])
```



Conclusions: Efficient Market Hypothesis

Returns are uncorrelated from one day to the next which makes sense, otherwise it'd be an easy inefficiency to exploit,and market inefficiencies that are exploited tend to disappear as a result.

```
# Sample a random return from the empirical joint distribution
# This simulates a random day
return.today = resample(all_returns, 1, orig.ids=FALSE)
```

P1: Safe Portfolio

ETFs: "SPY", "VUG", "SCHA", "QQQ", "IXUS", "VXX" For a safe portfolio, we gave more weights (80%) to SPY, VUG, QQQ because they are generally considered safer and consistently high performing.

Our initial wealth is $100,000.

```
## begin block
# Now simulate many different possible futures
# just repeating the above block thousands of times
set.seed(1)
initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.3,0.3,0.07, 0.2, 0.1, 0.03)
  holdings = weights * total_wealth
  n_days = 20
```

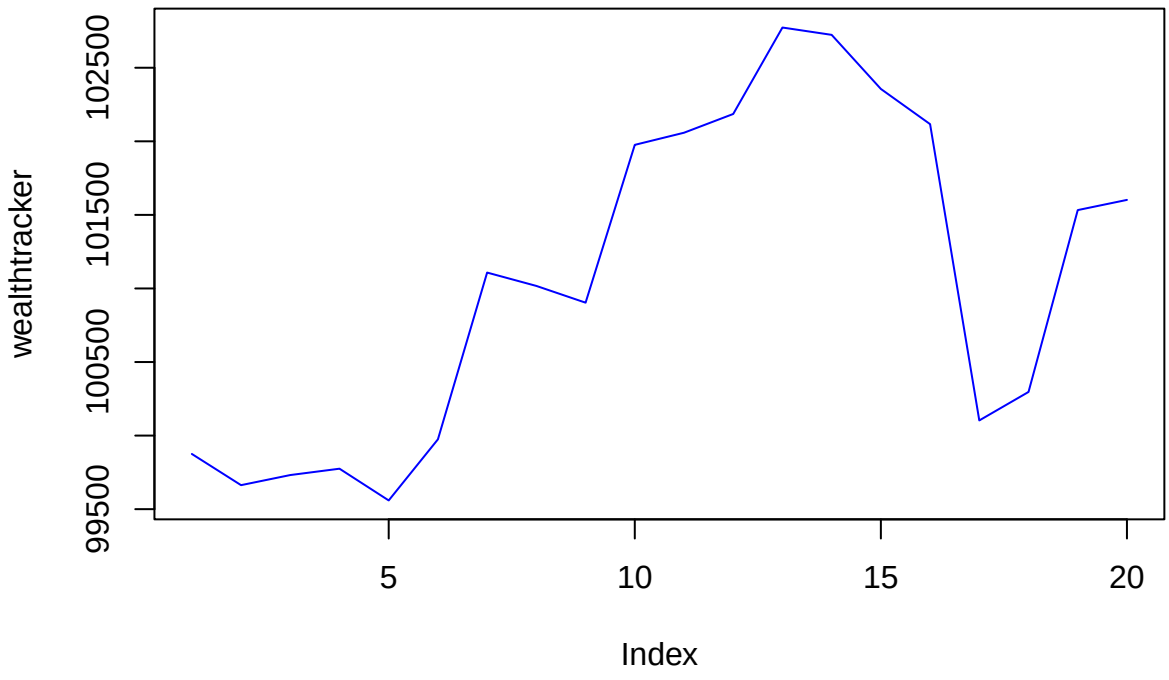
```
  wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
  }
  wealthtracker
}
```

Shows total wealth over the 20 days with this portfolio.

```
total_wealth
```

```
## [1] 101601.9
```

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

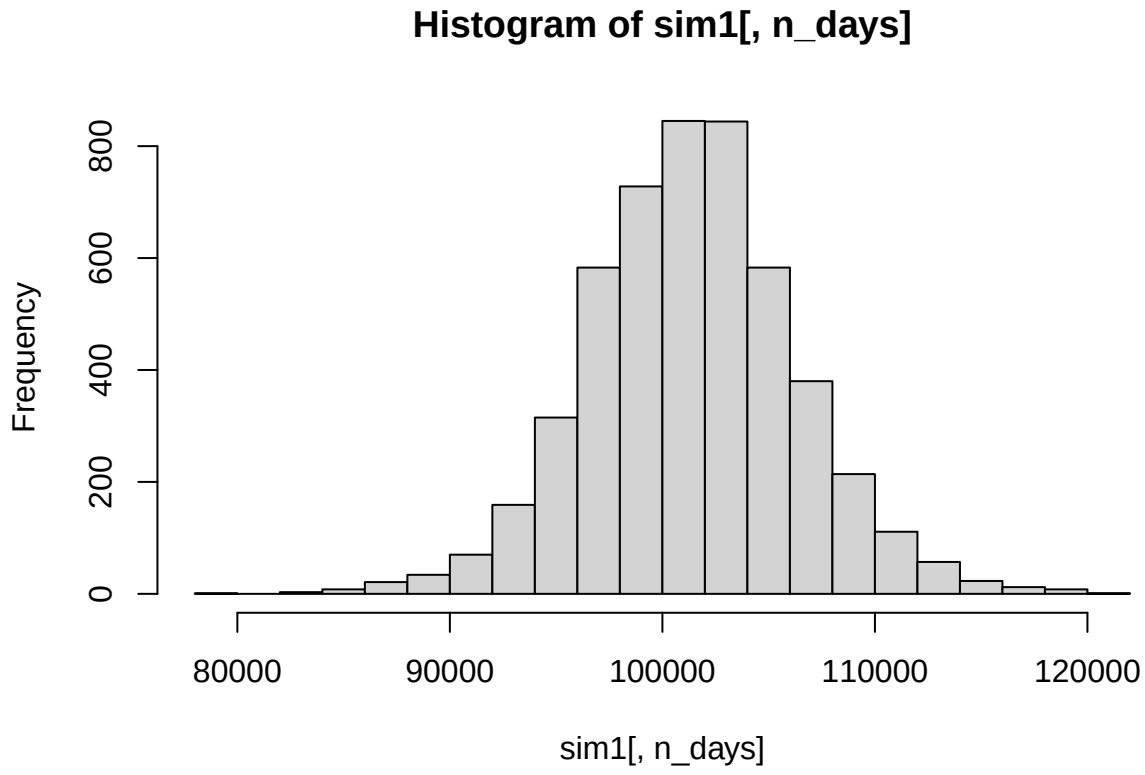



```
# each row is a simulated trajectory
# each column is a data
head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 100362.54 100969.29 102308.24 102121.01 101888.33 102045.32 101450.26
```

```
## result.2 100265.51  99121.92  98865.87  99560.62  99521.26  99446.38  99082.37
## result.3  99499.41  99464.25  99687.46  96840.81  96934.42  98568.97  98756.79
## result.4  95633.24  95235.67  95682.00  95531.30  95840.89  95343.79  95152.16
## result.5  99694.10  99303.55  99140.84  98026.72  98652.54  99942.02 100415.60
## result.6  99332.77  99850.27  98605.72  98774.34 100545.91 101615.32 102218.33
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 101845.69 101238.04 101054.88 102351.38 102305.37 102241.91 103479.15
## result.2  99356.39  99559.33  99135.28  99529.88 100696.00 101075.08 101302.99
## result.3  98410.41  99495.54  99322.62  97304.18  98648.09  99184.93  99254.84
## result.4  95104.22  95334.72  95807.43  96350.58  96617.14  96014.57  95652.57
## result.5  99815.81 100417.61 100771.03 101340.58 101493.20 101645.76 102296.90
## result.6 101992.10 101586.58 101855.13 101858.12 102148.05 101966.64 102689.52
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 102812.12 103654.88 103419.75 102232.67 102329.18 102937.82
## result.2 100995.78 100030.91 100282.61 100192.52 100523.80 102825.12
## result.3  99464.90  99440.73 100032.47 100238.64  99360.06  98781.17
## result.4  94805.45  94544.32  94748.43  94587.32  95524.06  96388.44
## result.5 103209.68 102908.44 102970.19 103085.56 103097.38 103218.49
## result.6 102195.62 105027.93 104121.43 103383.90 103643.73 104639.57
```

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




```
summary(sim1)
```

```
##        V1              V2              V3              V4
```

```
##  Min.   : 89684   Min.   : 86186   Min.   : 85021   Min.   : 85096
##  1st Qu.: 99718   1st Qu.: 99488   1st Qu.: 99304   1st Qu.: 99136
##  Median :100083   Median :100170   Median :100254   Median :100276
##  Mean   :100080   Mean   :100144   Mean   :100206   Mean   :100263
##  3rd Qu.:100529   3rd Qu.:100871   3rd Qu.:101170   3rd Qu.:101414
##  Max.   :108562   Max.   :109115   Max.   :111667   Max.   :111881
##        V5               V6               V7               V8
##  Min.   : 85034   Min.   : 82712   Min.   : 83687   Min.   : 83783
##  1st Qu.: 99108   1st Qu.: 99038   1st Qu.: 98913   1st Qu.: 98842
##  Median :100386   Median :100417   Median :100482   Median :100572
##  Mean   :100359   Mean   :100430   Mean   :100487   Mean   :100574
##  3rd Qu.:101643   3rd Qu.:101884   3rd Qu.:102075   3rd Qu.:102302
##  Max.   :113570   Max.   :114048   Max.   :114579   Max.   :116320
##        V9              V10              V11              V12
##  Min.   : 84228   Min.   : 84500   Min.   : 82334   Min.   : 82691
##  1st Qu.: 98757   1st Qu.: 98721   1st Qu.: 98598   1st Qu.: 98553
##  Median :100582   Median :100630   Median :100683   Median :100738
##  Mean   :100622   Mean   :100673   Mean   :100740   Mean   :100807
##  3rd Qu.:102478   3rd Qu.:102620   3rd Qu.:102808   3rd Qu.:102993
##  Max.   :117941   Max.   :117148   Max.   :116758   Max.   :116399
##       V13              V14              V15              V16
##  Min.   : 82542   Min.   : 82756   Min.   : 82588   Min.   : 82706
##  1st Qu.: 98494   1st Qu.: 98407   1st Qu.: 98377   1st Qu.: 98325
##  Median :100817   Median :100871   Median :100948   Median :101001
##  Mean   :100855   Mean   :100934   Mean   :100980   Mean   :101055
##  3rd Qu.:103154   3rd Qu.:103338   3rd Qu.:103527   3rd Qu.:103720
##  Max.   :116951   Max.   :118875   Max.   :118237   Max.   :119360
##       V17              V18              V19              V20
##  Min.   : 82391   Min.   : 79790   Min.   : 79712   Min.   : 79996
##  1st Qu.: 98294   1st Qu.: 98273   1st Qu.: 98296   1st Qu.: 98206
##  Median :101084   Median :101167   Median :101295   Median :101356
##  Mean   :101148   Mean   :101224   Mean   :101311   Mean   :101391
##  3rd Qu.:103883   3rd Qu.:104064   3rd Qu.:104155   3rd Qu.:104416
##  Max.   :119590   Max.   :119065   Max.   :120064   Max.   :121434
```

Profit/Loss

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

```
## [1] 101391.1
```

```
mean(sim1[,n_days] - initial_wealth) #mean profit/loss expected return
```

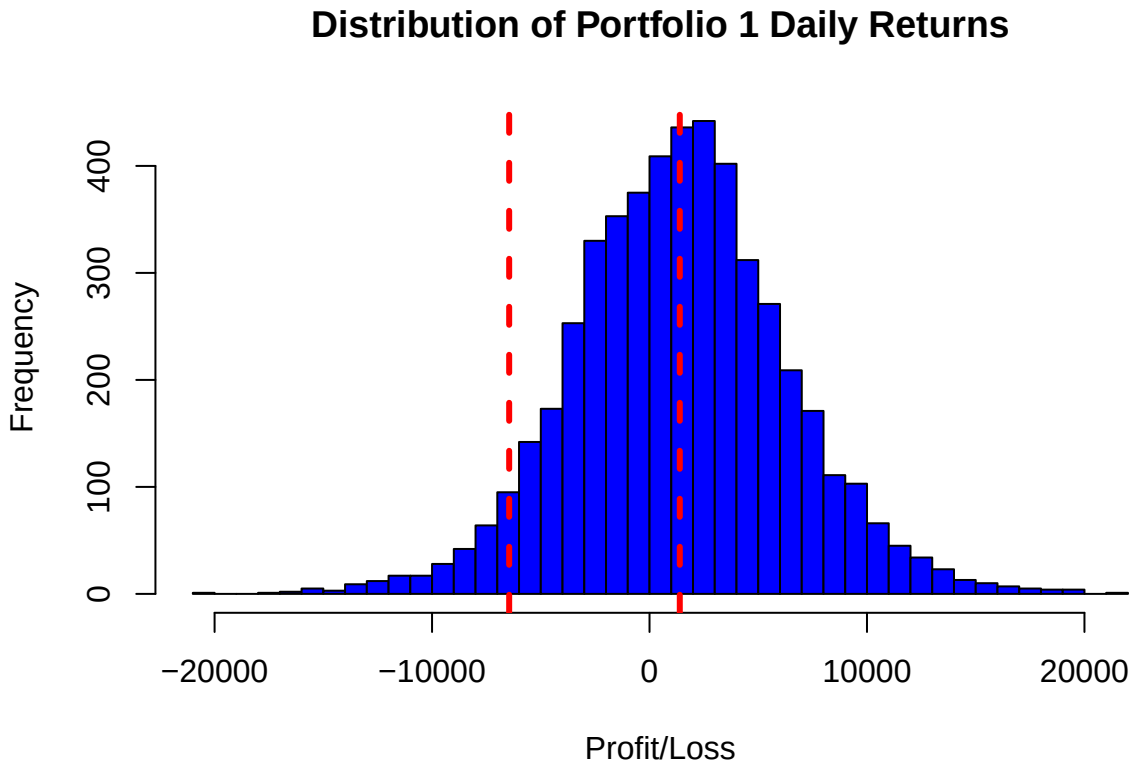
```
## [1] 1391.145
```

```
# Simple Value at Risk
# 5% value at risk:
var <- quantile(sim1[,n_days]- initial_wealth, prob=0.05)
var
```

```
##        5%
## -6453.556
```

Distribution of Portfolio 1 Daily Returns

```
hist(sim1[,n_days]- initial_wealth, breaks=30, col= 'Blue', main = 'Distribution of Portfolio 1 Daily R
     xlab = 'Profit/Loss')
abline(v = var, col="red", lwd=3, lty=2)
abline(v=mean(sim1[,n_days] - initial_wealth), col = 'red', lwd = 3, lty =2 )
```



Note: this is a negative number (a loss, e.g. -6453.556), but we conventionally express VaR as a positive number (e.g. 6453.556) – 5% chance.

Average return of investment after 20 days - $101391.1 (up 1391.145 from initial wealth)

5% Value at Risk for safe portfolio - $6453.556

P2: Aggressive Portfolio

ETFs: “SPY”, “VUG”, “SCHA”, “QQQ”, “IXUS”, “VXX” For an aggressive portfolio, we gave more weights (85%) to SCHA, VXX, QQQ because they are more high risk ETFs

Our initial wealth is $100,000.

```
set.seed(1)
initial_wealth = 100000
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.05,0.05,0.25, 0.3, 0.05, 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
}
```

Shows total wealth over the 20 days with this portfolio.

```
total_wealth
```

```
## [1] 96196.14
```

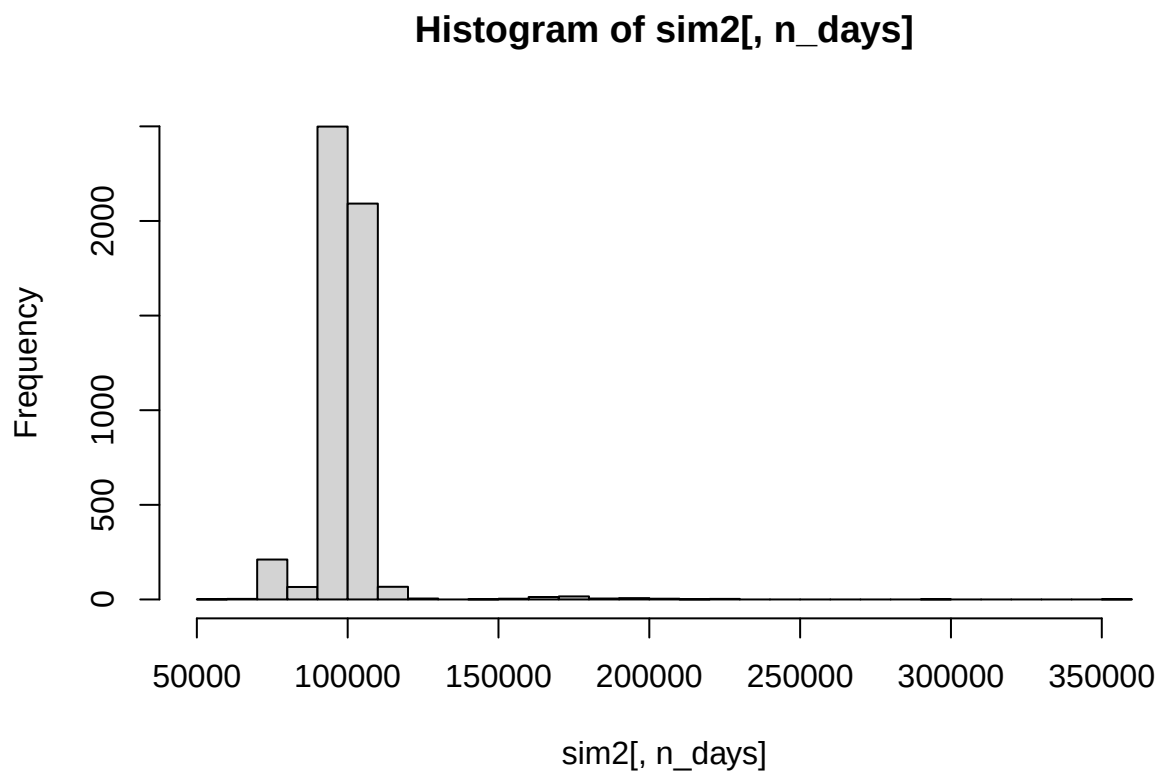
```
plot(wealthtracker, type='l', col = 'Green')
```


```
# each row is a simulated trajectory
# each column is a data
head(sim2)
```

```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
```

```
## result.1 100194.78  99764.56  99037.11  98538.14  98546.67  98177.74  98278.27
## result.2  99374.63 100183.16 100533.29 100404.79 100663.66 100190.70 104060.53
## result.3  99722.76  99804.19  99806.29 107309.55 107272.70 106191.02 106749.01
## result.4 100015.70 100396.52  98452.65  98897.00  98586.32  99071.74  98691.71
## result.5 100531.75  99423.80  99829.65  98890.37  99191.24  98324.06  97278.98
## result.6  99937.31 100391.00  98245.78  97940.15  98129.20  97945.21  96957.57
##               [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  98580.13  98894.17  98151.82  97389.19  97607.05  98125.97  98089.15
## result.2 104320.87 103734.05 104151.91 103721.58 102946.06 102820.00 102584.10
## result.3 106518.61 104777.69 104636.00 105155.70 102252.54 102573.13 100177.84
## result.4  99257.84  99348.84  99414.83  99133.69  98411.42  98492.18  97763.76
## result.5  96794.91  98081.72  98285.42  98973.23  99989.76  99667.87 100117.46
## result.6  96950.31  95726.82  95732.35  95766.20  95017.64  94965.33  95126.60
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  97893.22  97128.98  96754.73  96523.82  96618.82  96052.59
## result.2 102851.21 103671.16 103420.77 103134.91 103307.49 102396.23
## result.3 100090.28  99972.94  99364.81  98944.62  98967.33  98847.88
## result.4  97313.39  98658.78  98947.71 100396.43  99271.22  98743.11
## result.5 100722.41 101739.20 101674.79 101414.19 101351.06 102071.55
## result.6  95093.97  96865.42  96418.64  96635.55  96330.16  96673.08
```

```
hist(sim2[,n_days], 25)
```




```
head_sim2 <-summary(head(sim2))
```

Profit/Loss

```
# Profit/loss
mean(sim2[,n_days])
```

```
## [1] 99799.76
```

```
mean(sim2[,n_days] - initial_wealth) #mean profit/loss expected return
```

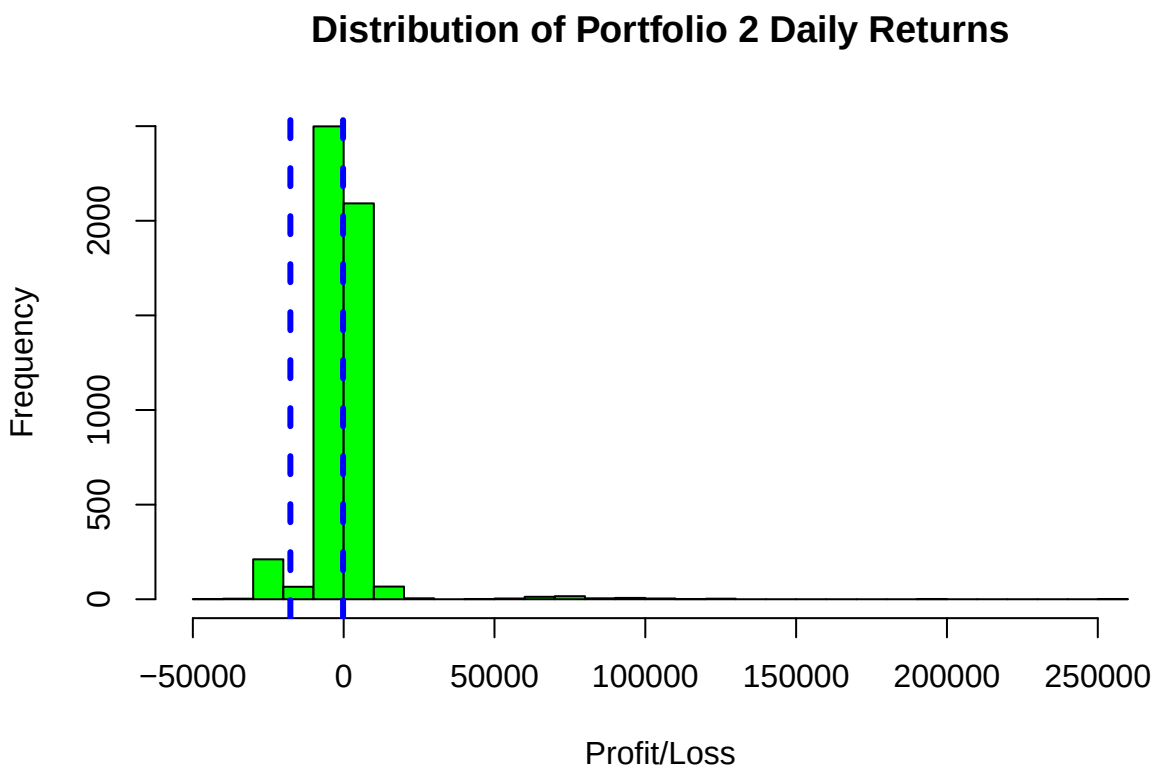
```
## [1] -200.2402
```

```
# Simple Value at Risk
# 5% value at risk:
var2 <- quantile(sim2[,n_days]- initial_wealth, prob=0.05)
var2
```

```
##        5%
## -17667.12
```

Distribution of Portfolio 2 Daily Returns

```
hist(sim2[,n_days]- initial_wealth, breaks=30, col= 'green', main = 'Distribution of Portfolio 2 Daily R
     xlab = 'Profit/Loss')
abline(v = var2, col="blue", lwd=3, lty=2)
abline(v=mean(sim2[,n_days] - initial_wealth), col = 'blue', lwd = 3, lty =2 )
```



Note: this is a negative number (a loss, e.g. -17667.12), but we conventionally express VaR as a positive number (e.g. 17667.12) – 5% chance.

Average return of investment after 20 days - $99799.76 (down -200.2402 from initial wealth)

5% Value at Risk for safe portfolio - $17667.12

P3: DIVERSE PORTFOLIO

ETFs: “SPY”, “VUG”, “SCHA”, “QQQ”, “IXUS”, “VXX” For a diverse portfolio, we gave equal weights to all of the ETFs.

Our initial wealth is $100,000.

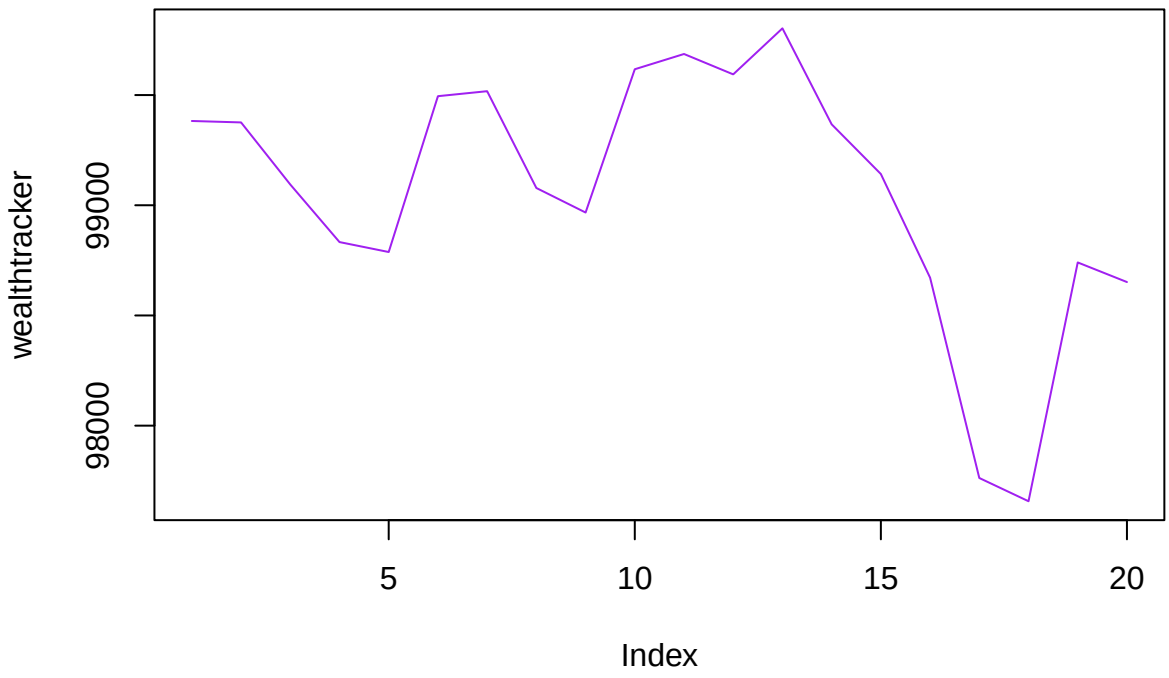
```
set.seed(1)
initial_wealth = 100000
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
  total_wealth = initial_wealth
  weights = c(0.167,0.167,0.167, 0.167, 0.167, 0.165)
  holdings = weights * total_wealth
  n_days = 20
  wealthtracker = rep(0, n_days)
  for(today in 1:n_days) {
    return.today = resample(all_returns, 1, orig.ids=FALSE)
    holdings = holdings + holdings*return.today
    total_wealth = sum(holdings)
    wealthtracker[today] = total_wealth
  }
  wealthtracker
}
```

Shows total wealth over the 20 days with this portfolio.

```
total_wealth
```

```
## [1] 98651.57
```

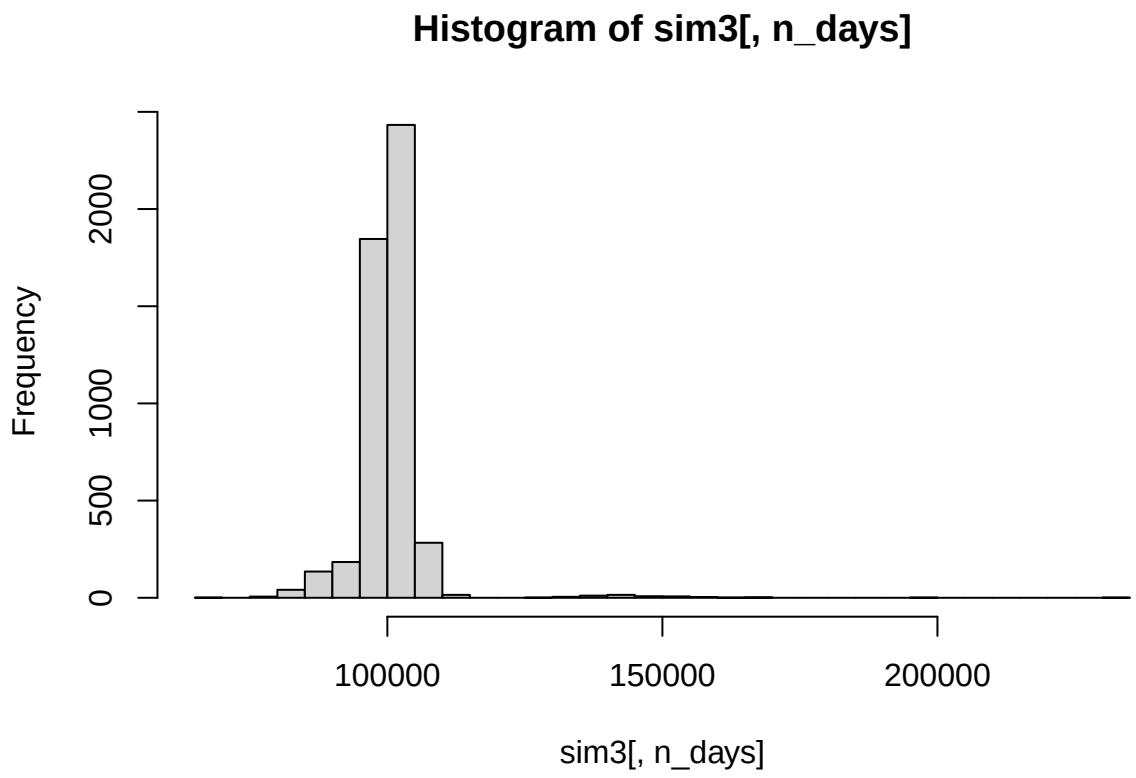
```
plot(wealthtracker, type='l', col = 'purple')
```




```
# each row is a simulated trajectory
# each column is a data
head(sim3)
```

```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 100266.55 100335.97 100644.16 100376.03 100274.02 100216.92  99986.46
## result.2  99782.70  99644.06  99674.62  99961.27 100059.48  99801.77 101498.38
## result.3  99640.03  99685.15  99848.31 102186.57 102174.67 102475.11 102884.93
## result.4  97739.25  97733.93  96959.33  97091.51  97054.66  97176.77  96965.41
## result.5 100091.33  99190.44  99296.90  98316.62  98584.02  98792.73  98520.44
## result.6  99665.07 100073.34  98464.11  98329.72  99317.40  99727.43  99551.20
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 100371.26 100275.00  99826.69 100025.64 100099.17 100297.42 100877.74
## result.2 101719.22 101521.17 101547.90 101554.33 101621.34 101709.95 101781.36
## result.3 102562.33 102200.52 102089.86 101435.96 100619.49 101018.54  99894.28
## result.4  97260.29  97339.27  97575.24  97696.65  97466.80  97244.21  96700.19
## result.5  98056.70  98983.17  99166.64  99791.82 100453.32 100389.73 100956.08
## result.6  99364.95  98580.14  98634.80  98606.28  98344.95  98225.02  98721.18
##             [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 100446.80 100427.13 100099.81  99485.77  99589.33  99611.62
## result.2 101764.69 101766.57 101721.21 101916.60 102077.02 102718.79
## result.3  99956.03  99871.23  99888.47  99765.05  99263.41  98860.46
## result.4  96282.26  96765.74  97046.83  97690.68  97542.79  97716.34
## result.5 101573.39 101820.29 101798.79 101711.59 101684.02 101955.64
## result.6  98482.40 100707.55 100231.59  99898.65  99810.13 100453.83
```

```
hist(sim3[,n_days], 25)
```




```
head_sim3 <-summary(head(sim2))
```

Profit/Loss

```
# Profit/loss
mean(sim3[,n_days])
```

```
## [1] 100499.5
```

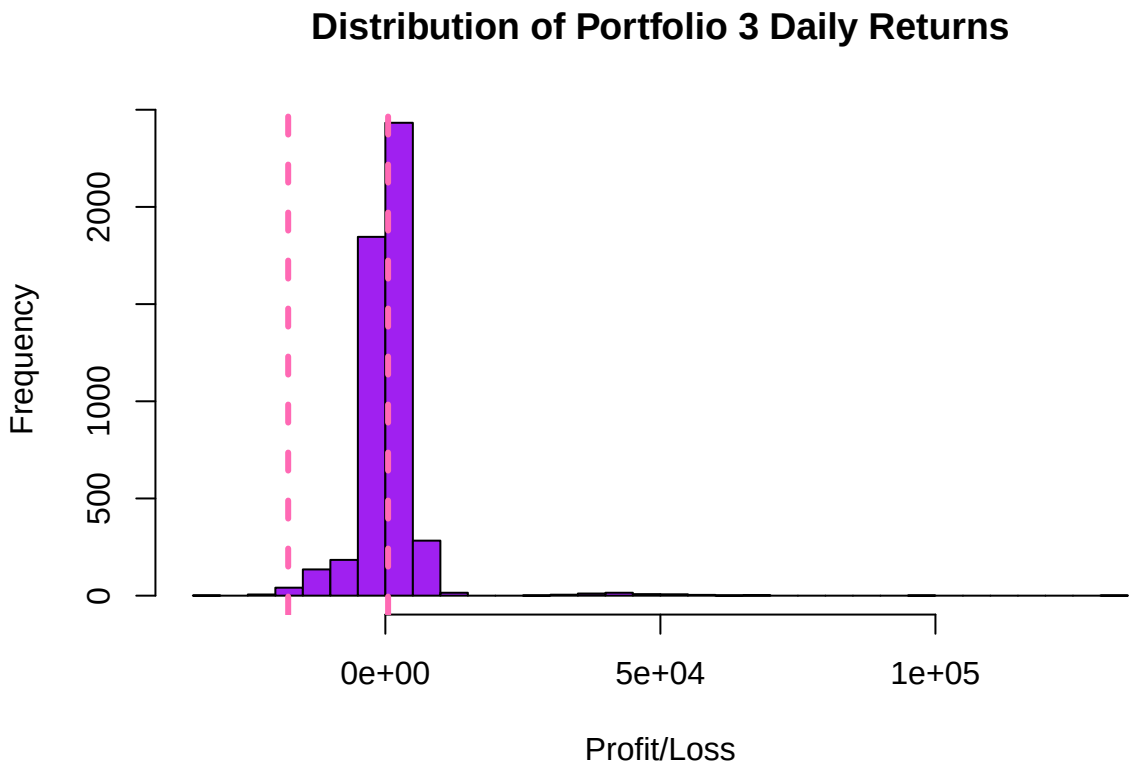
```
mean(sim3[,n_days] - initial_wealth) #mean profit/loss expected return
```

```
## [1] 499.4893
```

```
# Simple Value at Risk
# 5% value at risk:
var3 <- quantile(sim2[,n_days]- initial_wealth, prob=0.05)
var3
```

```
##        5%
## -17667.12
```

```
hist(sim3[,n_days]- initial_wealth, breaks=30, col= 'purple', main = 'Distribution of Portfolio 3 Daily
     xlab = 'Profit/Loss')
abline(v = var3, col="hot pink", lwd=3, lty=2)
abline(v=mean(sim3[,n_days] - initial_wealth), col = 'hot pink', lwd = 3, lty =2 )
```



Note: this is a negative number (a loss, e.g. -17667.12), but we conventionally express VaR as a positive number (e.g. 18972.43) – 5% chance.

Average return of investment after 20 days - $100499.5 (up 499.4893 dollars from initial wealth)

5% Value at Risk for safe portfolio - $17667.12

4. Market Segmentation

Load necessary libraries.

```
library(LICORS)
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
```

Load data.

```
mktg <- read.csv('social_marketing.csv')

mktg <- mktg[,-36] #remove spam
mktg <- mktg[,-36] #remove adult
mktg <- mktg[,-2] #remove chatter
mktg <- mktg[,-5] #remove uncategorized
```

Center and scale the data.

```
X = mktg[, -1]
X = scale(X, center=TRUE, scale=TRUE)
```

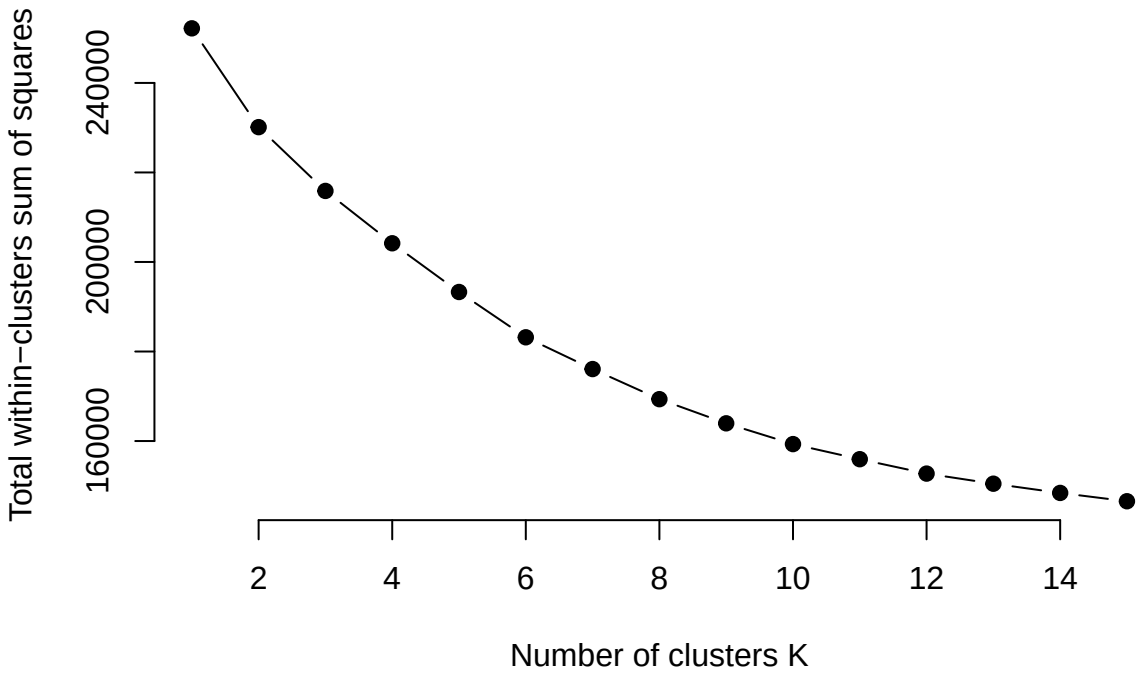
```
#Extract the centers and scales from the new rescaled data
mu = attr(X,"scaled:center")
sigma = attr(X,"scaled:scale")
```

Finding the optimal number of clusters using the Elbow Method.

```
set.seed(1)
# Compute and plot wss for k = 2 to k = 15.
k.max <- 15
data <- X
wss <- sapply(1:k.max,
              function(k){kmeans(data, k, nstart=50,iter.max = 15 )$tot.withinss})
wss
```

```
##  [1] 252192.0 230121.4 215873.7 204176.7 193297.4 183176.1 176078.5 169350.0
##  [9] 163954.3 159314.6 155944.8 152723.2 150466.9 148404.3 146550.0
```

```
plot(1:k.max, wss,
     type="b", pch = 19, frame = FALSE,
     xlab="Number of clusters K",
     ylab="Total within-clusters sum of squares")
```



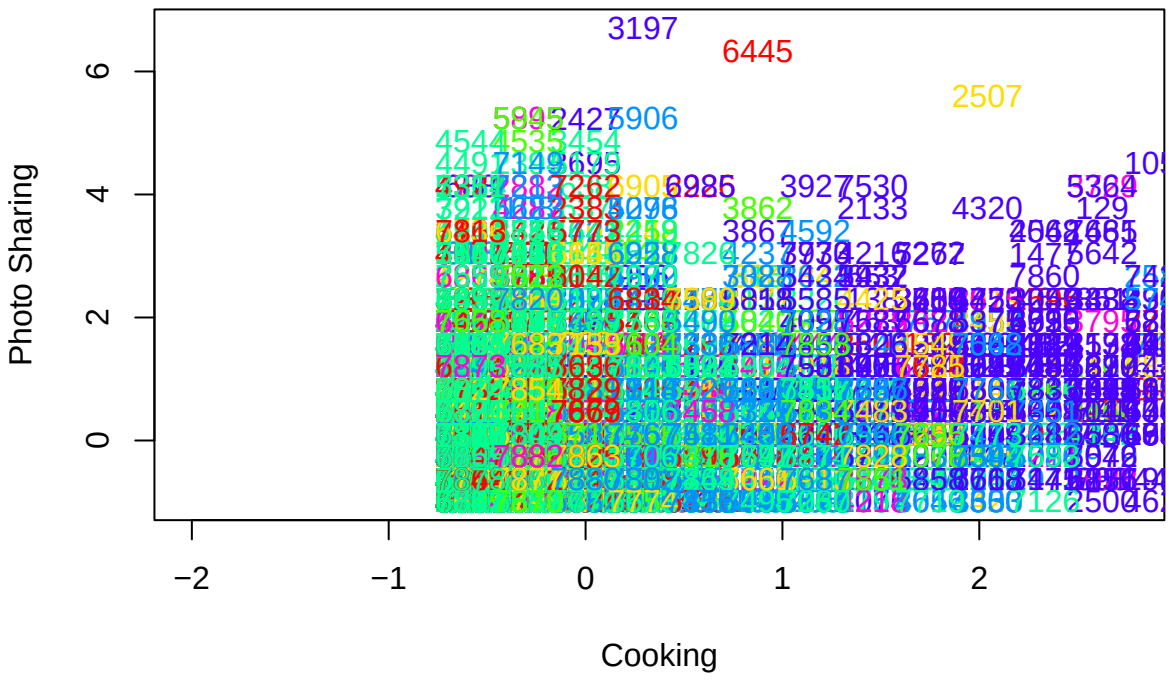
It is hard to find the number of clusters from the plot as the within sum of squares decreases as number of clusters increases. We chose 7 clusters because it is easier to look at and interpret and it will be easy to identify target market segments for NutrientH20 .

We ran a K-means with 7 clusters and 50 starts.

Cluster Visualization

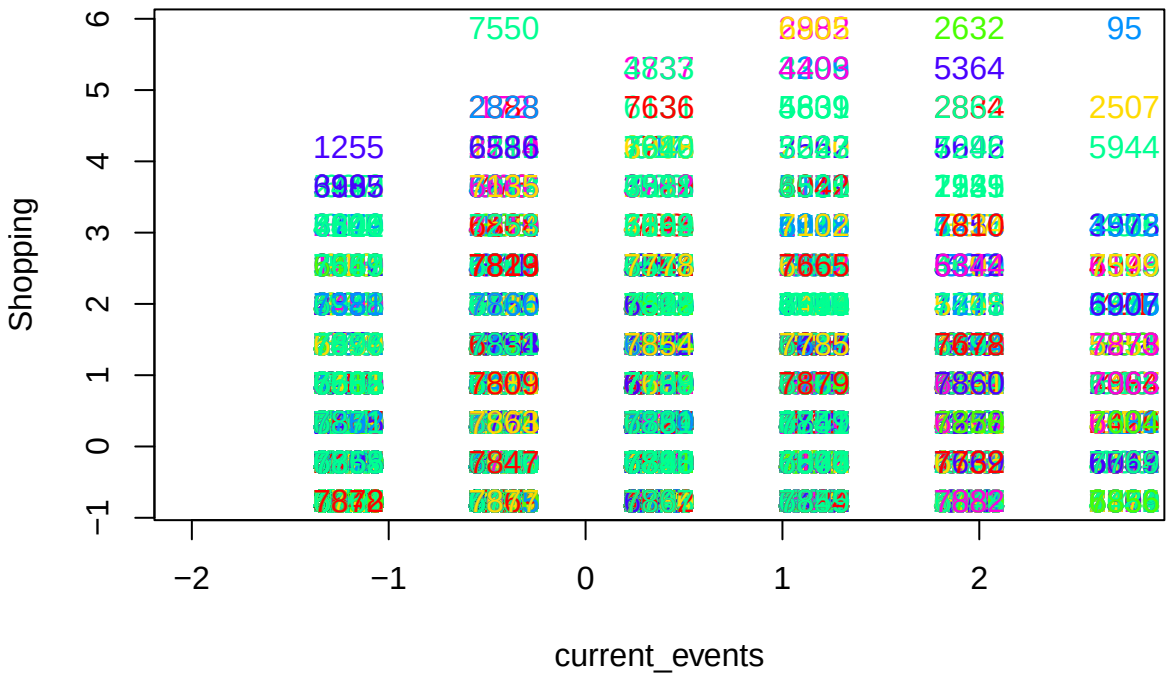
Cluster 1: Influencers

```
plot(X[,"cooking"], X[,"photo_sharing"], xlim=c(-2,2.75),
     type="n", ylab="Photo Sharing", xlab="Cooking")
text(X[,"cooking"], X[,"photo_sharing"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



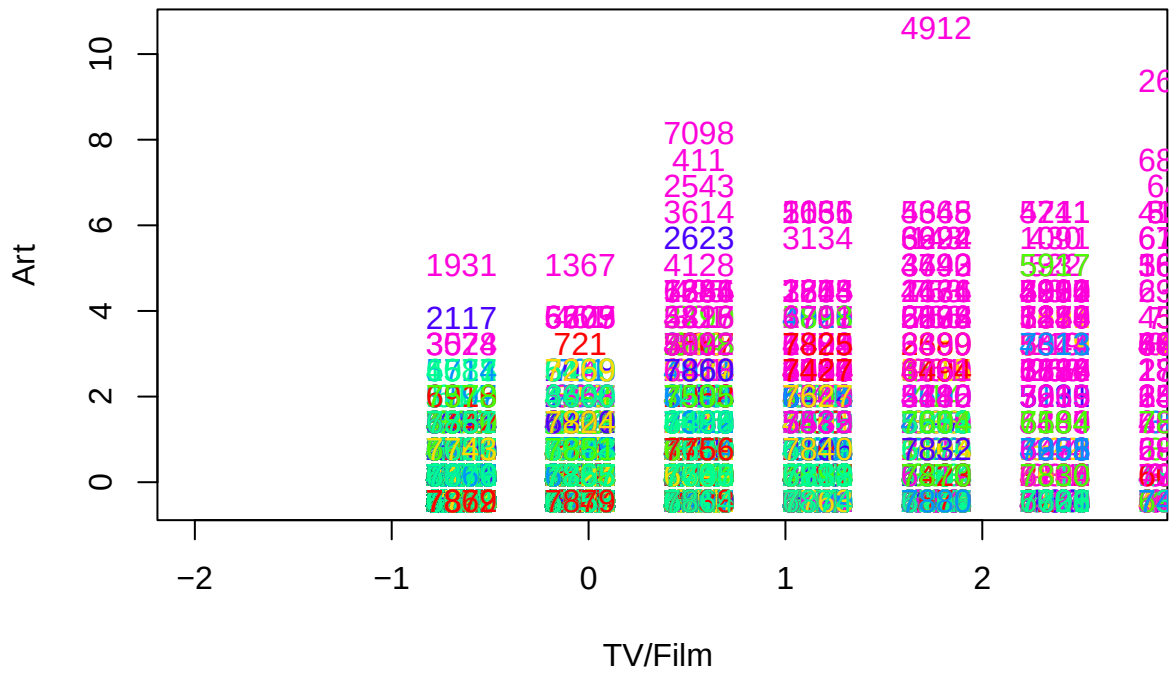
Cluster 2: Lifestyle Influencers

```
plot(X[,"current_events"], X[,"shopping"], xlim=c(-2,2.75),
     type="n", ylab="Shopping", xlab="current_events")
text(X[,"current_events"], X[,"shopping"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



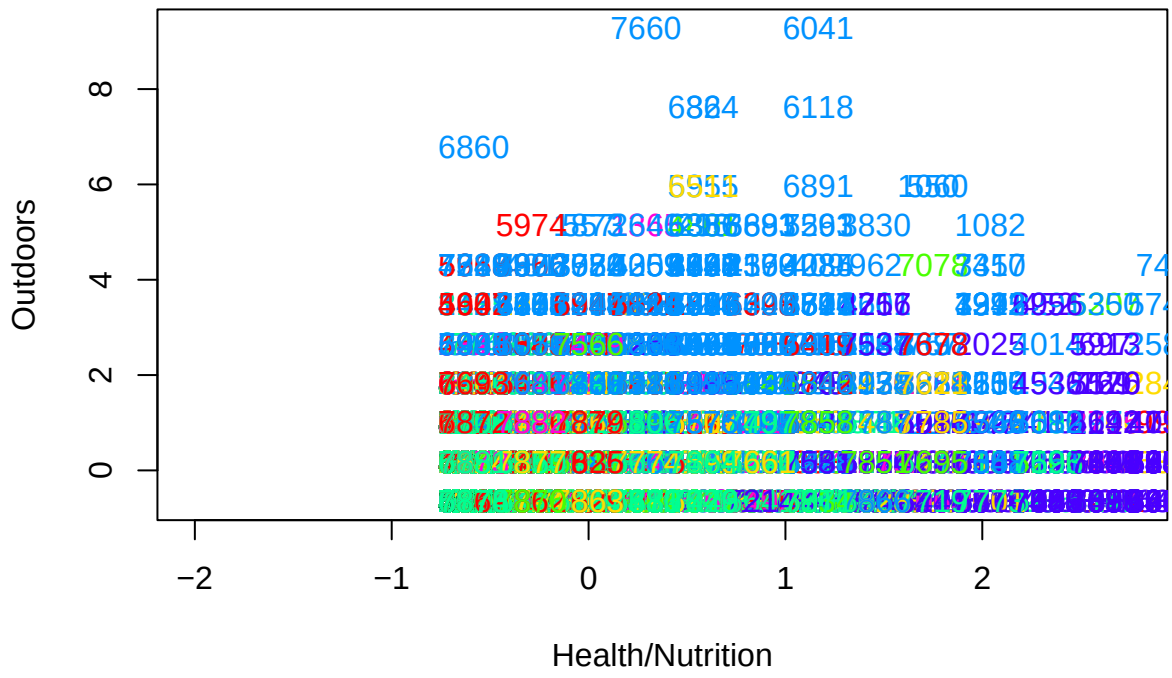
Cluster 3: Film Buffs

```
plot(X[,"tv_film"], X[,"art"], xlim=c(-2,2.75),
     type="n", ylab="Art", xlab="TV/Film ")
text(X[,"tv_film"], X[,"art"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



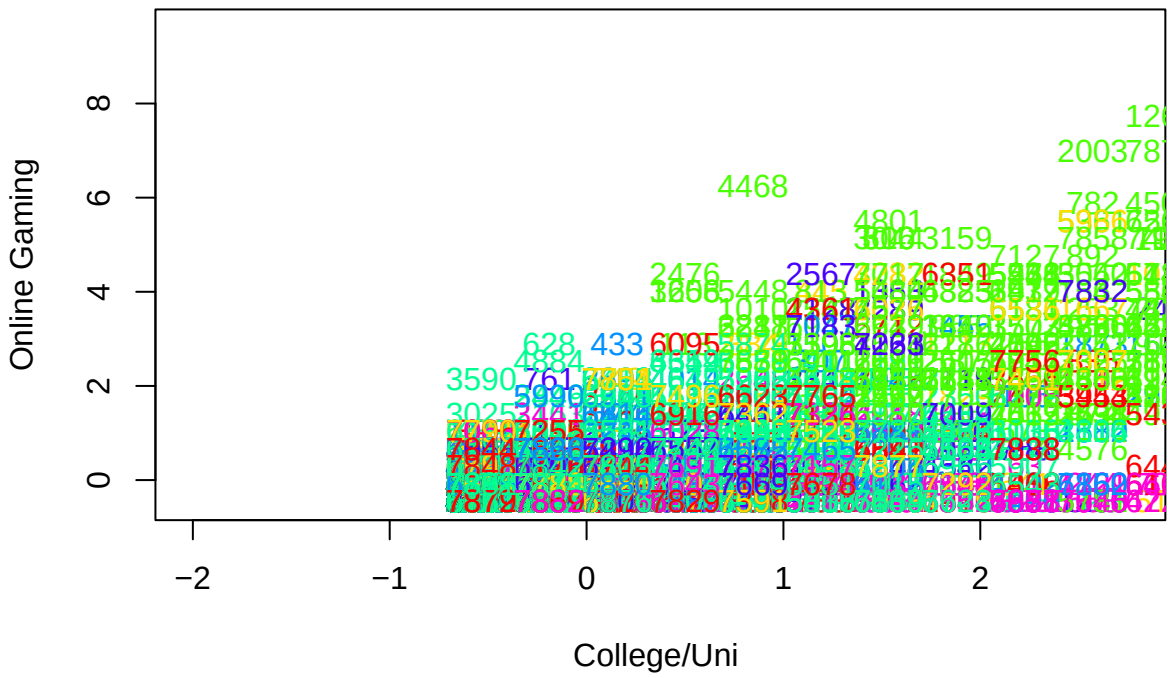
Cluster 4: Fitness/Healthy People

```
plot(X[,"health_nutrition"], X[,"outdoors"], xlim=c(-2,2.75),
     type="n", ylab="Outdoors", xlab="Health/Nutrition")
text(X[,"cooking"], X[,"outdoors"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



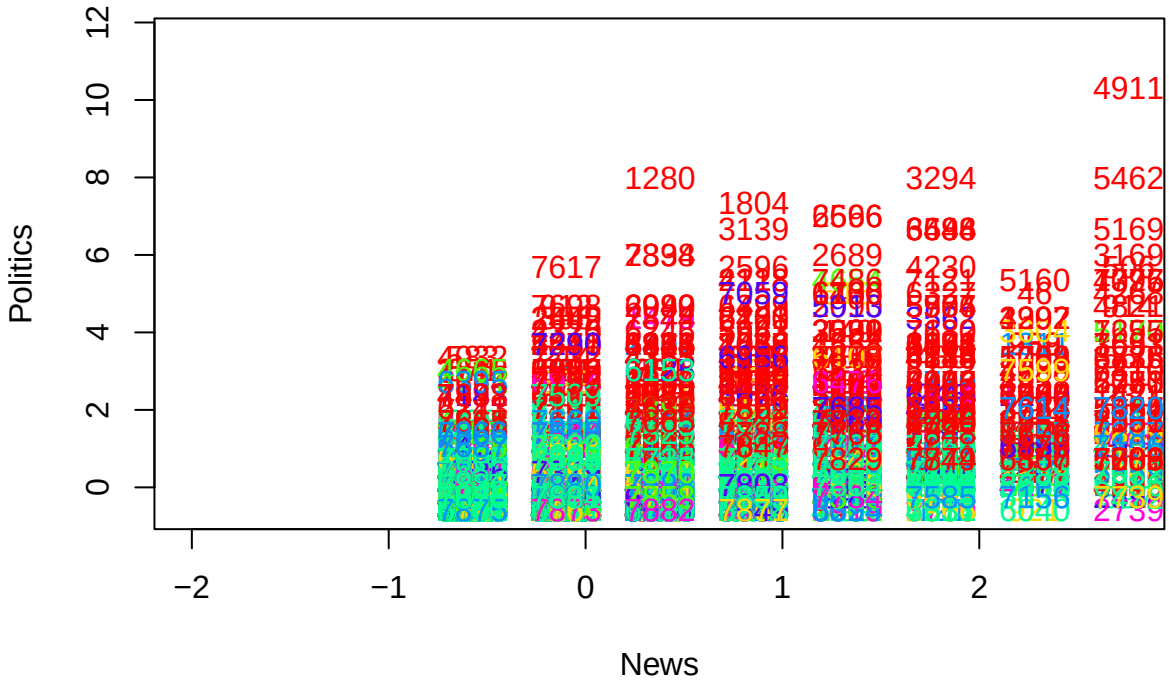
Cluster 5: College Gamers

```
plot(X[,"college_uni"], X[,"online_gaming"], xlim=c(-2,2.75),
     type="n", ylab="Online Gaming", xlab="College/Uni ")
text(X[,"college_uni"], X[,"online_gaming"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



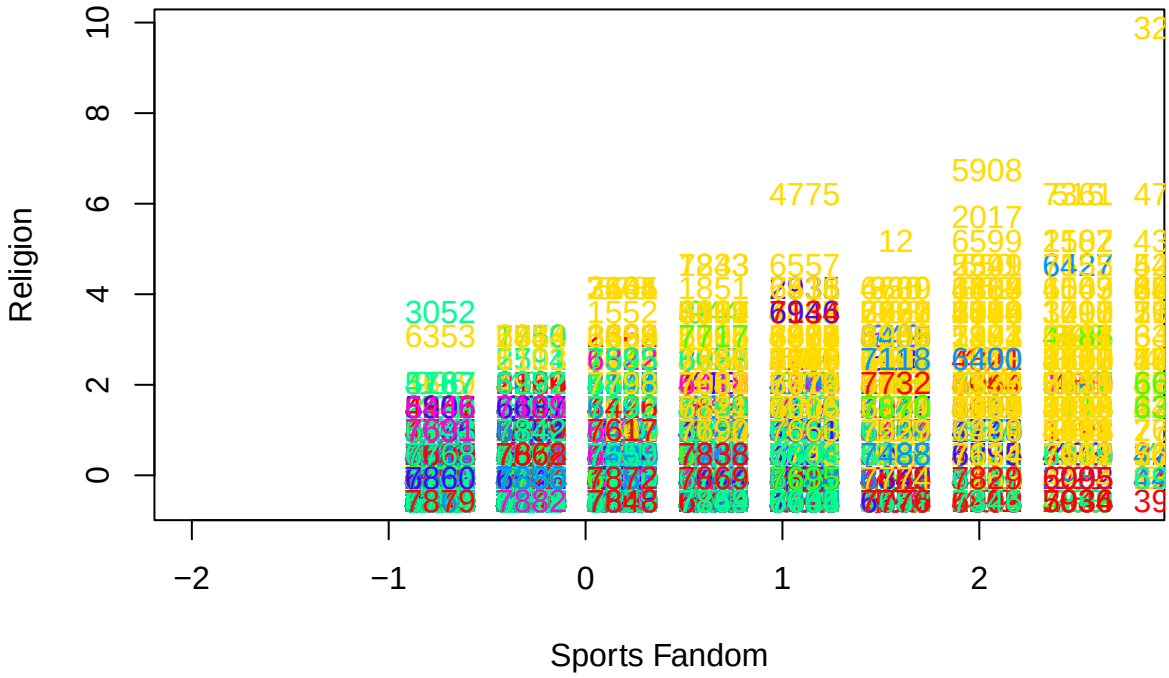
Cluster 6: Wall Street/NY/Bankers & Lawyers

```
plot(X[,"news"], X[,"politics"], xlim=c(-2,2.75),
     type="n", ylab="Politics", xlab="News")
text(X[,"news"], X[,"politics"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



Cluster 7: Suburban Dads

```
plot(X[,"sports_fandom"], X[,"religion"], xlim=c(-2,2.75),
     type="n", ylab="Religion", xlab="Sports Fandom")
text(X[,"sports_fandom"], X[,"religion"], labels=rownames(mktg),
     col=rainbow(7)[cluster_all$cluster])
```



Market Segments: 1. Cooking, Photo Sharing, Beauty, Fashion - Social Media Influencers 2. Travel, Current Events, Shopping - Lifestyle Influencers/Trendy People 3. TV Film, Art - Film Buffs/Artistic Critics 4. Personal fitness, Nutrition, Outdoors, Cooking - Fit/Healthy People 5. Online Gaming, College/Uni - College Gamers 6. Politics, Travel, News - Wall Street/NY/Bankers & Lawyers 7. Sports, Parenting, Food, Family, Religion - Suburban Dads

Based on the K++-Means clustering, we can identify seven distinct market segments that NutrientH20 can target for their marketing campaigns.

For example, Cluster 4 - which we dubbed Fit/Healthy People and Cluster 5 - which we call College Gamers are very different. Cluster 5 consists mainly of (we can assume young) people who are in school/university and enjoy playing video games; meanwhile, Cluster 4 are more focused on being active and healthy.

Cluster in 1 is primarily people who appear to like cooking, photos, beauty and fashion, which we can identify as our social media influencers. In contrast, Cluster 2 has people that focus on all parts of life and seem to keep up with the times whether that be fashion, news, or travel.

Cluster 3 seems like people with artistic opinions (because art and tv/film were high) - they seem to like talking about tv shows, movies, and art (probably ranging from theater to music to physical art).

Cluster 6 and 7 were also very different. Cluster 6 we classified as Wall Street/New Yorkers/Politicians because this audiences' interests were politics,travel, and news. On the other hand, Cluster 7 is primarily people who like sports, parenting, food, family and religion; we classified them as a "suburban dad" type because those interest align with something our dads like.

5. Author Attribution

Importing the correct libraries

1. setting up the training set setting up reader plain function

```
readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }
```

preprocessing the training data and tokenization

```
author_dirs = Sys.glob('../data/ReutersC50/C50train/*')

file_list = NULL
labels = NULL

for(author in author_dirs) {
  author_name = substring(author, first=29)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  labels = append(labels, rep(author_name, length(files_to_add)))
}

readerPlain = function(fname){
  readPlain(elem=list(content=readLines(fname)),
            id=fname, language='en') }

# Need a more clever regex to get better names here
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

authors = lapply(file_list, readerPlain)

mynames = file_list %>%
  { strsplit(., '/', fixed=TRUE) } %>%
  { lapply(., tail, n=2) } %>%
  { lapply(., paste0, collapse = '') } %>%
  unlist
```

Putting authors and text into a corpus and dropping lowercases, numbers, punctuation, white space, and stop-words.

```
names(authors) = mynames

documents_raw = Corpus(VectorSource(authors))

my_documents = documents_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

my_documents = tm_map(my_documents, content_transformer(removeWords), stopwords("en"))

DTM = DocumentTermMatrix(my_documents)
DTM
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 32570)>>
## Non-/sparse entries: 537861/80887139
## Sparsity           : 99%
## Maximal term length: 44
## Weighting          : term frequency (tf)
```

Removing sparse items and storing training set into variable

```
DTM = removeSparseTerms(DTM, 0.99)

tfidf = weightTfIdf(DTM)

#creating the training set
X_train = as.matrix(tfidf)
tfidf
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3393)>>
## Non-/sparse entries: 382971/8099529
## Sparsity           : 95%
## Maximal term length: 44
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

2. Creating the test set Reading in files and pre-processing the data with tokenization

```
author_dirs2 = Sys.glob('../data/ReutersC50/C50test/*')
file_list2 = NULL
labels2 = NULL
for(author in author_dirs2) {
  author_name2 = substring(author, first = 28)
  files_to_add2 = Sys.glob(paste0(author, '/*.txt'))
  file_list2 = append(file_list2, files_to_add2)
  labels2 = append(labels2, rep(author_name2, length(files_to_add2)))
}

# Need a more clever regex to get better names here
all_docs2 = lapply(file_list2, readerPlain)
names(all_docs2) = file_list2

names(all_docs2) = sub('.txt', '', names(all_docs2))

authors2 = lapply(file_list2, readerPlain)

mynames2 = file_list2 %>%
  { strsplit(., '/', fixed=TRUE) } %>%
  { lapply(., tail, n=2) } %>%
  { lapply(., paste0, collapse = '') } %>%
  unlist

names(authors2) = mynames2


documents_raw2 = Corpus(VectorSource(authors2))
```

```
my_documents2 = documents_raw2 %>%
  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

my_documents2 = tm_map(my_documents2, content_transformer(removeWords), stopwords("en"))

DTM2 = DocumentTermMatrix(my_documents2, list(dictionary = colnames(DTM)))
DTM2
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3393)>>
## Non-/sparse entries: 419314/8063186
## Sparsity           : 95%
## Maximal term length: 44
## Weighting          : term frequency (tf)
```

Creating the test set

```
tfidf2 = weightTfIdf(DTM2)

#Creating the test set
X_test = as.matrix(tfidf2)
tfidf2
```

```
## <<DocumentTermMatrix (documents: 2500, terms: 3393)>>
## Non-/sparse entries: 379314/8103186
## Sparsity           : 96%
## Maximal term length: 44
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

3. Applying PCA for dimensionality reduction Prepping data for reduction eliminating columns with no values

```
X_train1 = X_train[,which(colSums(X_train)!= 0)]
X_test1 <- X_test[,which(colSums(X_test)!= 0)]
```

getting the intersecting columns

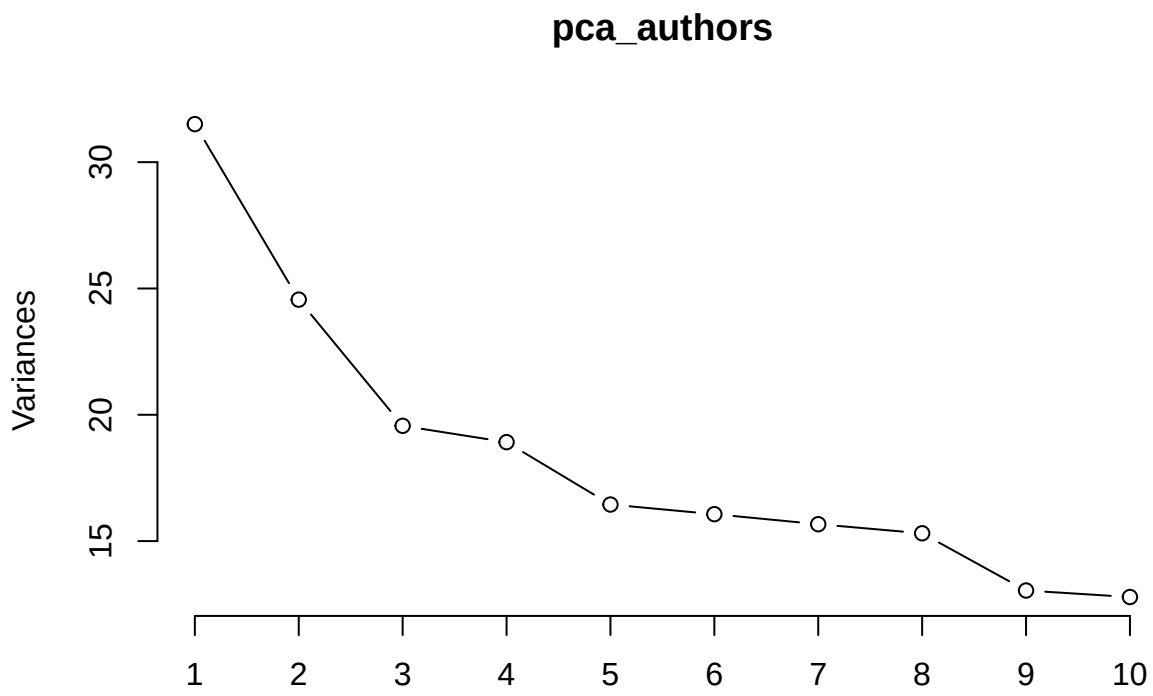
```
X_train1 = X_train[,intersect(colnames(X_test1),colnames(X_train1))]
X_test1 = X_test[,intersect(colnames(X_test1),colnames(X_train1))]
```

getting the principal components

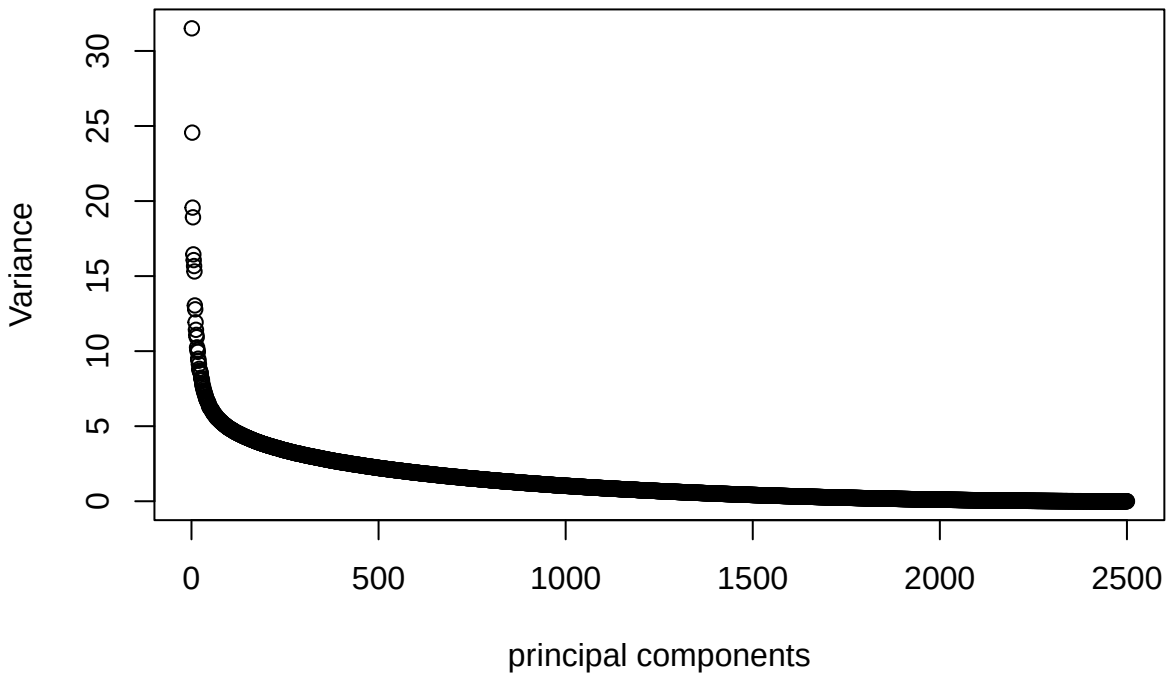
```
pca_authors = prcomp(X_train1, scale=TRUE)
pca_pred = predict(pca_authors, newdata = X_test1)
```

getting the right number of principal components to use for modeling

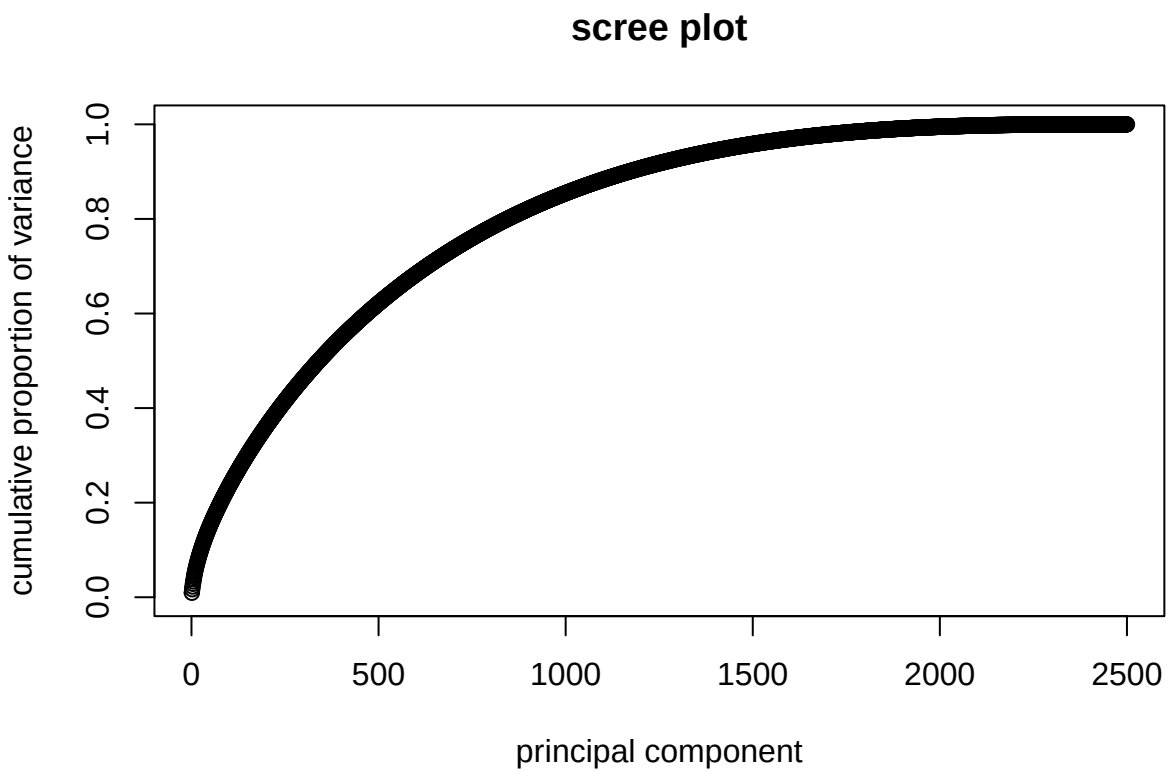
```
plot(pca_authors, type = 'line')
```




```
variance <- apply(pca_authors$x, 2, var)
proportion_var <- variance / sum(variance)
cumulative_proportion <- cumsum(proportion_var)
y = variance
x = c(1:2500)
plot1 <- plot(x, y, xlab = 'principal components', ylab = 'Variance')
```




```
y1 <- cumsum(pca_authors$sdev^2/sum(pca_authors$sdev^2))
x1 <- c(1:2500)
plot(x1, y1,
     xlab = 'principal component',
     ylab = 'cumulative proportion of variance',
     ylim = c(0,1),
     type = 'b',
     main = 'scree plot')
```



Based on our principal component analysis, at around principal component 729, 75% of the variance is explained, so 729 principal components will be used.

Prepping dataset for various models to be applied

```
final_train = data.frame(pca_authors$x[,1:729])
final_train['author'] = labels
final_load = pca_authors$rotation[,1:729]

final_test_pre <- scale(X_test1) %*% final_load
final_test <- as.data.frame(final_test_pre)
final_test['author'] = labels2
```

4. Creating a KNN model Preparing Data

```
train <- subset(final_train, select = -c(author))
test <- subset(final_test, select = -c(author))
train_author <- as.factor(final_train$author)
test_author <- as.factor(final_test$author)
```

Applying KNN function and calculating test accuracy

```
set.seed(123)
knn_prediction <- knn(train, test, train_author, k = 12)
knn_mat <- as.data.frame(cbind(knn_prediction, test_author))
knn_mat_val <- ifelse(as.integer(knn_prediction)==as.integer(test_author),1,0)
sum(knn_mat_val)
```

```
## [1] 870
```

```
sum(knn_mat_val*100/nrow(knn_mat))
```

```
## [1] 34.8
```

When using the 12 nearest neighbors, the test accuracy is about 34.8%. This questions the effectiveness of using KNN to model this data.

5. Creating a Random Forest decision tree applying random forest function on previously prepared data

```
set.seed(123)
forest_authors <- randomForest(as.factor(author)~., data = final_train, mtry = 96, importance = TRUE)
forest_pred <- predict(forest_authors, data = final_test)
```

Calculating test accuracy

```
pred_mat <- as.data.frame(table(forest_pred, as.factor(final_test$author)))
prediction <- forest_pred
actual_value <- as.factor(final_test$author)
prediction_actual <- as.data.frame(cbind(actual_value, prediction))
prediction_actual$flag <- ifelse(prediction_actual$actual_value == prediction_actual$prediction,1,0)
sum(prediction_actual$flag)
```

```
## [1] 2013
```

```
sum(prediction_actual$flag)*100/nrow(prediction_actual)
```

```
## [1] 80.52
```

Using a random forest model with 96 trees will yield a test accuracy of about 80.52%, which is much better than the nearest neighbors model created above. KNN and random forest were the models that we chose to create for these data. One thing to consider when running Random Forest is the number of tress to implement. We chose to implement 96 trees for the optimal test accuracy, but this may increase run time in the code. Other models such as Naive Bayes or multiple regression could also be used as well, with varying degrees of test accuracy.

6. Association Rule Mining

```
#load packages
library(arules)
library(arulesViz)
library(tidyverse)
#detach(package:tm, unload=TRUE)
```

```
groceries <- read_csv("groceries.txt", col_names = FALSE)
```

```
##
## -- Column specification --------------------------------------------------------
## cols(
##   X1 = col_character(),
##   X2 = col_character(),
##   X3 = col_character(),
##   X4 = col_character()
## )

## Warning: 8830 parsing failures.
## row col  expected    actual           file
##   2  -- 4 columns 3 columns 'groceries.txt'
##   3  -- 4 columns 1 columns 'groceries.txt'
##   6  -- 4 columns 5 columns 'groceries.txt'
##   7  -- 4 columns 1 columns 'groceries.txt'
##   8  -- 4 columns 5 columns 'groceries.txt'
## ... ... ......... ......... ..............
## See problems(...) for more details.
```

```
head(groceries)
```

```
## # A tibble: 6 x 4
##   X1               X2                 X3             X4
##   <chr>            <chr>              <chr>          <chr>
## 1 citrus fruit     semi-finished bread margarine      ready soups
## 2 tropical fruit   yogurt             coffee         <NA>
## 3 whole milk       <NA>               <NA>           <NA>
## 4 pip fruit        yogurt             cream cheese   meat spreads
## 5 other vegetables whole milk         condensed milk long life bakery product
## 6 whole milk       butter             yogurt         rice
```

```
summary(groceries)
```

```
##       X1                 X2                 X3                 X4
##  Length:9835        Length:9835        Length:9835        Length:9835
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
```

Turn user into a factor.

```
groceries$X1 <- as.factor(groceries$X1)
groceries$X2 <- as.factor(groceries$X2)
groceries$X3 <- as.factor(groceries$X3)
groceries$X4 <- as.factor(groceries$X4)
```

Cast this variable as a special arules "transactions" class.

```
str(groceries)
```

```
## spec_tbl_df [9,835 x 4] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ X1: Factor w/ 158 levels "abrasive cleaner",..: 29 147 156 108 101 156 117 101 111 156 ...
```

```
##  $ X2: Factor w/ 151 levels "abrasive cleaner",..: 118 150 NA 150 149 12 NA 143 NA 22 ...
##  $ X3: Factor w/ 155 levels "abrasive cleaner",..: 82 32 NA 37 33 154 NA 114 NA NA ...
##  $ X4: Factor w/ 153 levels "abrasive cleaner",..: 106 NA NA 80 76 108 NA 8 NA NA ...
##  - attr(*, "problems")= tibble [8,830 x 5] (S3: tbl_df/tbl/data.frame)
##   ..$ row     : int [1:8830] 2 3 6 7 8 9 10 11 12 13 ...
##   ..$ col     : chr [1:8830] NA NA NA NA ...
##   ..$ expected: chr [1:8830] "4 columns" "4 columns" "4 columns" "4 columns" ...
##   ..$ actual  : chr [1:8830] "3 columns" "1 columns" "5 columns" "1 columns" ...
##   ..$ file    : chr [1:8830] "'groceries.txt'" "'groceries.txt'" "'groceries.txt'" "'groceries.txt'"
##  - attr(*, "spec")=
##   .. cols(
##   ..   X1 = col_character(),
##   ..   X2 = col_character(),
##   ..   X3 = col_character(),
##   ..   X4 = col_character()
##   .. )
```

```
summary(groceries)
```

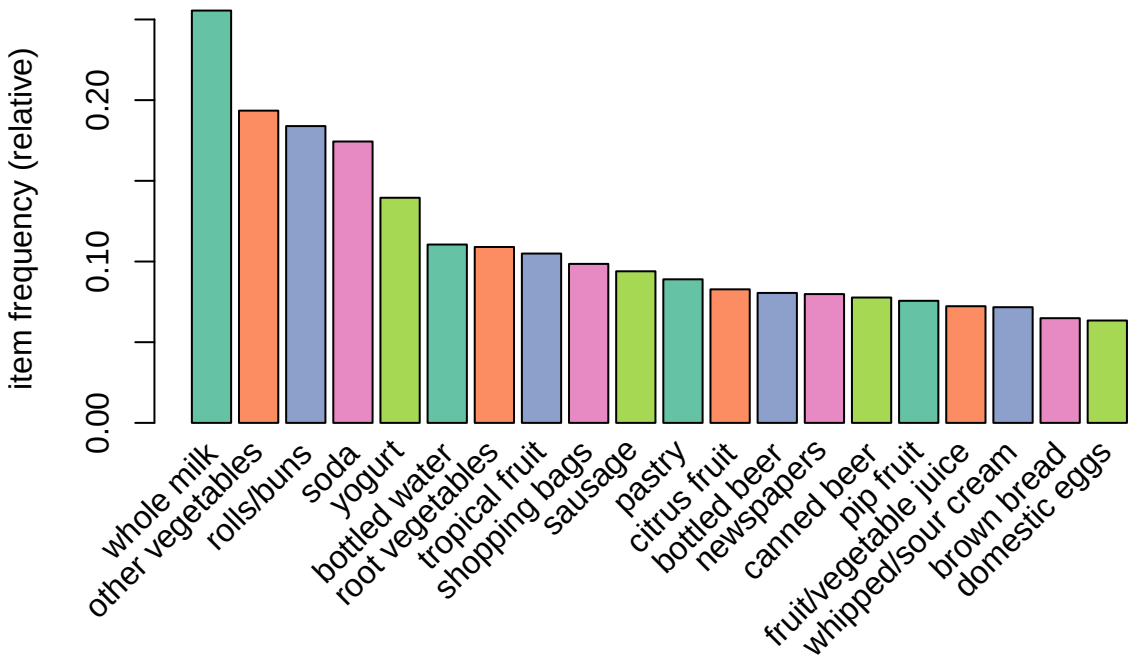
```
##                X1                     X2                     X3
##  sausage         : 825   whole milk      : 654   whole milk      : 506
##  whole milk      : 717   other vegetables: 550   other vegetables: 415
##  frankfurter     : 580   root vegetables : 383   rolls/buns      : 293
##  tropical fruit  : 482   rolls/buns      : 378   yogurt          : 289
##  other vegetables: 460   tropical fruit  : 355   soda            : 229
##  citrus fruit    : 453   (Other)         :5356   (Other)         :4301
##  (Other)         :6318   NA's            :2159   NA's            :3802
##                X4
##  whole milk      : 315
##  other vegetables: 254
##  rolls/buns      : 238
##  soda            : 211
##  yogurt          : 202
##  (Other)         :3514
##  NA's            :5101
```

```
groceries <- read.transactions(file="groceries.txt",sep = ',',format="basket",rm.duplicates=TRUE)
groc_trans = as(groceries, "transactions")
summary(groc_trans)
```

```
## 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
```

```
library(RColorBrewer)
coul <- brewer.pal(5, "Set2")
itemFrequencyPlot(groc_trans, topN = 20, col=coul)
```



We transform the data into a "transactions" class before applying the apriori algorithm. There are total of 9835 transactions in our dataset.

Whole milk is the most frequent item bought by shoppers, followed by other vegetables, then rolls & buns.

```
# apriori algorithm expects a list of baskets in a special format
# In this case, one "basket" of groceries per customer
# First split data into a list of artists for each customer
groceries <- read.transactions(file="groceries.txt",sep = ',',format="basket",rm.duplicates=TRUE)
summary(groceries)
```

```
## 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
```

```
groc_rules <- apriori(groceries,
                      parameter=list(support=.001, confidence=0.9, maxlen=50)) #128 rules
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.9    0.1    1 none FALSE            TRUE       5   0.001      1
##  maxlen target  ext
##      50  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [129 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
arules::inspect(groc_rules)
```

```
##       lhs                            rhs                    support confidence    coverage      lift c
## [1]   {liquor,
##        red/blush wine}          => {bottled beer}     0.001931876  0.9047619 0.002135231 11.235269
## [2]   {cereals,
##        curd}                    => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [3]   {bottled beer,
##        soups}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [4]   {house keeping products,
##        whipped/sour cream}      => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [5]   {pastry,
##        sweet spreads}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [6]   {rice,
##        sugar}                   => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [7]   {bottled water,
##        rice}                    => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [8]   {canned fish,
##        hygiene articles}        => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [9]   {grapes,
##        onions}                  => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [10]  {hard cheese,
##        oil}                     => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [11]  {butter,
##        rice,
##        root vegetables}         => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [12]  {fruit/vegetable juice,
##        herbs,
##        whole milk}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [13]  {citrus fruit,
##        herbs,
##        tropical fruit}          => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [14]  {flour,
##        root vegetables,
##        whipped/sour cream}      => {whole milk}       0.001728521  1.0000000 0.001728521  3.913649
## [15]  {butter,
##        domestic eggs,
##        soft cheese}             => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [16]  {soft cheese,
##        tropical fruit,
##        whipped/sour cream}      => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [17]  {root vegetables,
##        soft cheese,
##        whipped/sour cream}      => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [18]  {citrus fruit,
##        root vegetables,
##        soft cheese}             => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [19]  {frankfurter,
##        frozen meals,
##        tropical fruit}          => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [20]  {frankfurter,
##        frozen meals,
##        tropical fruit}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [21]  {butter,
##        frozen meals,
##        tropical fruit}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
```

```
## [22]  {hard cheese,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [23]  {butter milk,
##        pork,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [24]  {butter milk,
##        fruit/vegetable juice,
##        pip fruit}                => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [25]  {frankfurter,
##        root vegetables,
##        sliced cheese}            => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [26]  {butter,
##        sliced cheese,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [27]  {coffee,
##        oil,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [28]  {napkins,
##        onions,
##        root vegetables}          => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [29]  {berries,
##        butter,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [30]  {hamburger meat,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [31]  {butter,
##        hygiene articles,
##        napkins}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [32]  {butter,
##        hygiene articles,
##        pip fruit}                => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [33]  {butter,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [34]  {domestic eggs,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [35]  {hygiene articles,
##        root vegetables,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [36]  {hygiene articles,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [37]  {cream cheese,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [38]  {cream cheese,
##        other vegetables,
##        sugar}                    => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [39]  {curd,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
```

```
## [40]  {citrus fruit,
##         domestic eggs,
##         sugar}                    => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [41]  {domestic eggs,
##         sugar,
##         tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [42]  {domestic eggs,
##         sugar,
##         yogurt}                   => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [43]  {root vegetables,
##         sugar,
##         whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [44]  {pork,
##         rolls/buns,
##         waffles}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [45]  {long life bakery product,
##         napkins,
##         whipped/sour cream}       => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [46]  {long life bakery product,
##         napkins,
##         tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [47]  {butter,
##         long life bakery product,
##         sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [48]  {dessert,
##         tropical fruit,
##         whipped/sour cream}       => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [49]  {cream cheese,
##         domestic eggs,
##         napkins}                  => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [50]  {butter,
##         cream cheese,
##         root vegetables}          => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [51]  {butter,
##         cream cheese,
##         root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [52]  {cream cheese,
##         pip fruit,
##         whipped/sour cream}       => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [53]  {cream cheese,
##         pip fruit,
##         sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [54]  {citrus fruit,
##         cream cheese,
##         root vegetables}          => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [55]  {butter,
##         root vegetables,
##         white bread}              => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [56]  {butter,
##         coffee,
##         whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [57]  {coffee,
##         domestic eggs,
##         root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
```

```
## [58]  {butter,
##        curd,
##        domestic eggs}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [59]  {butter,
##        citrus fruit,
##        curd}                    => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [60]  {bottled beer,
##        domestic eggs,
##        margarine}               => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [61]  {brown bread,
##        pip fruit,
##        whipped/sour cream}      => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [62]  {domestic eggs,
##        fruit/vegetable juice,
##        margarine}               => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [63]  {butter,
##        pip fruit,
##        whipped/sour cream}      => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [64]  {butter,
##        soda,
##        whipped/sour cream}      => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [65]  {butter,
##        pastry,
##        pip fruit}               => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [66]  {domestic eggs,
##        tropical fruit,
##        whipped/sour cream}      => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [67]  {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}      => {other vegetables} 0.001931876  0.9047619 0.002135231  4.675950
## [68]  {other vegetables,
##        rice,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [69]  {rice,
##        root vegetables,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [70]  {herbs,
##        other vegetables,
##        root vegetables,
##        tropical fruit}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [71]  {grapes,
##        tropical fruit,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [72]  {frozen meals,
##        pip fruit,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [73]  {hard cheese,
##        other vegetables,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
```

```
## [74]  {ham,
##        pip fruit,
##        tropical fruit,
##        yogurt}                 => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [75]  {ham,
##        pip fruit,
##        tropical fruit,
##        whole milk}             => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [76]  {butter,
##        sliced cheese,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [77]  {butter,
##        sliced cheese,
##        tropical fruit,
##        whole milk}             => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [78]  {oil,
##        root vegetables,
##        tropical fruit,
##        yogurt}                 => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [79]  {oil,
##        root vegetables,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [80]  {oil,
##        other vegetables,
##        root vegetables,
##        yogurt}                 => {whole milk}       0.001423488  1.0000000 0.001423488  3.913649
## [81]  {oil,
##        root vegetables,
##        whole milk,
##        yogurt}                 => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
## [82]  {other vegetables,
##        root vegetables,
##        waffles,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [83]  {cream cheese,
##        curd,
##        other vegetables,
##        whipped/sour cream}     => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [84]  {citrus fruit,
##        cream cheese,
##        whipped/sour cream,
##        whole milk}             => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [85]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        root vegetables}        => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [86]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        yogurt}                 => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [87]  {butter,
##        tropical fruit,
```

```
##        white bread,
##        yogurt}                  => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [88]  {butter,
##        other vegetables,
##        tropical fruit,
##        white bread}             => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [89]  {butter,
##        other vegetables,
##        root vegetables,
##        white bread}             => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [90]  {butter,
##        root vegetables,
##        white bread,
##        whole milk}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [91]  {citrus fruit,
##        frozen vegetables,
##        other vegetables,
##        yogurt}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [92]  {beef,
##        rolls/buns,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [93]  {curd,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [94]  {citrus fruit,
##        curd,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [95]  {butter,
##        napkins,
##        other vegetables,
##        whipped/sour cream}      => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [96]  {butter,
##        other vegetables,
##        pork,
##        whipped/sour cream}      => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [97]  {butter,
##        other vegetables,
##        pork,
##        root vegetables}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [98]  {frankfurter,
##        root vegetables,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [99]  {brown bread,
##        other vegetables,
##        pip fruit,
##        root vegetables}         => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [100] {brown bread,
##        other vegetables,
##        rolls/buns,
##        root vegetables}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
```

```
## [101] {butter,
##        domestic eggs,
##        other vegetables,
##        whipped/sour cream}       => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [102] {butter,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [103] {butter,
##        domestic eggs,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [104] {butter,
##        fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [105] {butter,
##        soda,
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [106] {bottled water,
##        butter,
##        citrus fruit,
##        other vegetables}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [107] {newspapers,
##        rolls/buns,
##        soda,
##        whole milk}               => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [108] {domestic eggs,
##        other vegetables,
##        pip fruit,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [109] {citrus fruit,
##        domestic eggs,
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [110] {domestic eggs,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [111] {domestic eggs,
##        other vegetables,
##        tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [112] {citrus fruit,
##        domestic eggs,
##        other vegetables,
##        tropical fruit}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [113] {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                   => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [114] {fruit/vegetable juice,
##        tropical fruit,
```

```
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [115] {fruit/vegetable juice,
##        pip fruit,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [116] {citrus fruit,
##        fruit/vegetable juice,
##        other vegetables,
##        soda}                     => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [117] {citrus fruit,
##        pastry,
##        rolls/buns,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [118] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001220132  1.0000000 0.001220132  5.168156
## [119] {bottled water,
##        other vegetables,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [120] {pastry,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [121] {root vegetables,
##        sausage,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [122] {rolls/buns,
##        root vegetables,
##        sausage,
##        tropical fruit}           => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [123] {bottled water,
##        rolls/buns,
##        root vegetables,
##        tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [124] {oil,
##        other vegetables,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [125] {oil,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [126] {oil,
##        other vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [127] {butter,
```

```
##        domestic eggs,
##        other vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [128] {citrus fruit,
##        root vegetables,
##        whipped/sour cream,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [129] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
```

```
arules::inspect(subset(groc_rules, subset=lift > 5))
```

```
##      lhs                       rhs                    support confidence    coverage      lift coun
## [1]  {liquor,
##       red/blush wine}       => {bottled beer}     0.001931876  0.9047619 0.002135231 11.235269    19
## [2]  {citrus fruit,
##       root vegetables,
##       soft cheese}          => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156    10
## [3]  {butter,
##       cream cheese,
##       root vegetables}      => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698    10
## [4]  {brown bread,
##       pip fruit,
##       whipped/sour cream}   => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156    11
## [5]  {grapes,
##       tropical fruit,
##       whole milk,
##       yogurt}               => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156    10
## [6]  {ham,
##       pip fruit,
##       tropical fruit,
##       yogurt}               => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156    10
## [7]  {ham,
##       pip fruit,
##       tropical fruit,
##       whole milk}           => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156    11
## [8]  {butter,
##       sliced cheese,
##       tropical fruit,
##       whole milk}           => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698    10
## [9]  {cream cheese,
##       curd,
##       other vegetables,
##       whipped/sour cream}   => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698    10
## [10] {butter,
##       other vegetables,
##       tropical fruit,
##       white bread}          => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698    10
## [11] {butter,
```

```
##       fruit/vegetable juice,
##       tropical fruit,
##       whipped/sour cream}    => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156    10
## [12] {newspapers,
##       rolls/buns,
##       soda,
##       whole milk}            => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156    10
## [13] {citrus fruit,
##       fruit/vegetable juice,
##       other vegetables,
##       soda}                  => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400    10
## [14] {citrus fruit,
##       root vegetables,
##       tropical fruit,
##       whipped/sour cream}    => {other vegetables} 0.001220132  1.0000000 0.001220132  5.168156    12
## [15] {oil,
##       other vegetables,
##       tropical fruit,
##       whole milk,
##       yogurt}                => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400    10
```

```
arules::inspect(subset(groc_rules, subset=confidence > 0.6))
```

```
##       lhs                           rhs                    support confidence    coverage      lift c
## [1]   {liquor,
##        red/blush wine}          => {bottled beer}     0.001931876  0.9047619 0.002135231 11.235269
## [2]   {cereals,
##        curd}                    => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [3]   {bottled beer,
##        soups}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [4]   {house keeping products,
##        whipped/sour cream}      => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [5]   {pastry,
##        sweet spreads}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [6]   {rice,
##        sugar}                   => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [7]   {bottled water,
##        rice}                    => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [8]   {canned fish,
##        hygiene articles}        => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [9]   {grapes,
##        onions}                  => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [10]  {hard cheese,
##        oil}                     => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [11]  {butter,
##        rice,
##        root vegetables}         => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [12]  {fruit/vegetable juice,
##        herbs,
##        whole milk}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [13]  {citrus fruit,
##        herbs,
##        tropical fruit}          => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [14]  {flour,
```

```
##        root vegetables,
##        whipped/sour cream}       => {whole milk}       0.001728521  1.0000000 0.001728521  3.913649
## [15]  {butter,
##        domestic eggs,
##        soft cheese}              => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [16]  {soft cheese,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [17]  {root vegetables,
##        soft cheese,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [18]  {citrus fruit,
##        root vegetables,
##        soft cheese}              => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [19]  {frankfurter,
##        frozen meals,
##        tropical fruit}           => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [20]  {frankfurter,
##        frozen meals,
##        tropical fruit}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [21]  {butter,
##        frozen meals,
##        tropical fruit}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [22]  {hard cheese,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [23]  {butter milk,
##        pork,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [24]  {butter milk,
##        fruit/vegetable juice,
##        pip fruit}                => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [25]  {frankfurter,
##        root vegetables,
##        sliced cheese}            => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [26]  {butter,
##        sliced cheese,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [27]  {coffee,
##        oil,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [28]  {napkins,
##        onions,
##        root vegetables}          => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [29]  {berries,
##        butter,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [30]  {hamburger meat,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [31]  {butter,
##        hygiene articles,
##        napkins}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [32]  {butter,
```

```
##        hygiene articles,
##        pip fruit}                => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [33]  {butter,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [34]  {domestic eggs,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [35]  {hygiene articles,
##        root vegetables,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [36]  {hygiene articles,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [37]  {cream cheese,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [38]  {cream cheese,
##        other vegetables,
##        sugar}                    => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [39]  {curd,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [40]  {citrus fruit,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [41]  {domestic eggs,
##        sugar,
##        tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [42]  {domestic eggs,
##        sugar,
##        yogurt}                   => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [43]  {root vegetables,
##        sugar,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [44]  {pork,
##        rolls/buns,
##        waffles}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [45]  {long life bakery product,
##        napkins,
##        whipped/sour cream}       => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [46]  {long life bakery product,
##        napkins,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [47]  {butter,
##        long life bakery product,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [48]  {dessert,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [49]  {cream cheese,
##        domestic eggs,
##        napkins}                  => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [50]  {butter,
```

```
##        cream cheese,
##        root vegetables}          => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [51]  {butter,
##        cream cheese,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [52]  {cream cheese,
##        pip fruit,
##        whipped/sour cream}       => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [53]  {cream cheese,
##        pip fruit,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [54]  {citrus fruit,
##        cream cheese,
##        root vegetables}          => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [55]  {butter,
##        root vegetables,
##        white bread}              => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [56]  {butter,
##        coffee,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [57]  {coffee,
##        domestic eggs,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [58]  {butter,
##        curd,
##        domestic eggs}            => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [59]  {butter,
##        citrus fruit,
##        curd}                     => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [60]  {bottled beer,
##        domestic eggs,
##        margarine}                => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [61]  {brown bread,
##        pip fruit,
##        whipped/sour cream}       => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [62]  {domestic eggs,
##        fruit/vegetable juice,
##        margarine}                => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [63]  {butter,
##        pip fruit,
##        whipped/sour cream}       => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [64]  {butter,
##        soda,
##        whipped/sour cream}       => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [65]  {butter,
##        pastry,
##        pip fruit}                => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [66]  {domestic eggs,
##        tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [67]  {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001931876  0.9047619 0.002135231  4.675950
## [68]  {other vegetables,
```

```
##        rice,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [69]  {rice,
##        root vegetables,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [70]  {herbs,
##        other vegetables,
##        root vegetables,
##        tropical fruit}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [71]  {grapes,
##        tropical fruit,
##        whole milk,
##        yogurt}                  => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [72]  {frozen meals,
##        pip fruit,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [73]  {hard cheese,
##        other vegetables,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [74]  {ham,
##        pip fruit,
##        tropical fruit,
##        yogurt}                  => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [75]  {ham,
##        pip fruit,
##        tropical fruit,
##        whole milk}              => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [76]  {butter,
##        sliced cheese,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [77]  {butter,
##        sliced cheese,
##        tropical fruit,
##        whole milk}              => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [78]  {oil,
##        root vegetables,
##        tropical fruit,
##        yogurt}                  => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [79]  {oil,
##        root vegetables,
##        tropical fruit,
##        yogurt}                  => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [80]  {oil,
##        other vegetables,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001423488  1.0000000 0.001423488  3.913649
## [81]  {oil,
##        root vegetables,
##        whole milk,
```

```
##        yogurt}                 => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
## [82]  {other vegetables,
##        root vegetables,
##        waffles,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [83]  {cream cheese,
##        curd,
##        other vegetables,
##        whipped/sour cream}     => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [84]  {citrus fruit,
##        cream cheese,
##        whipped/sour cream,
##        whole milk}             => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [85]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        root vegetables}        => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [86]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        yogurt}                 => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [87]  {butter,
##        tropical fruit,
##        white bread,
##        yogurt}                 => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [88]  {butter,
##        other vegetables,
##        tropical fruit,
##        white bread}            => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [89]  {butter,
##        other vegetables,
##        root vegetables,
##        white bread}            => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [90]  {butter,
##        root vegetables,
##        white bread,
##        whole milk}             => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [91]  {citrus fruit,
##        frozen vegetables,
##        other vegetables,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [92]  {beef,
##        rolls/buns,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [93]  {curd,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [94]  {citrus fruit,
##        curd,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [95]  {butter,
```

```
##        napkins,
##        other vegetables,
##        whipped/sour cream}       => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [96]  {butter,
##        other vegetables,
##        pork,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [97]  {butter,
##        other vegetables,
##        pork,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [98]  {frankfurter,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [99]  {brown bread,
##        other vegetables,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [100] {brown bread,
##        other vegetables,
##        rolls/buns,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [101] {butter,
##        domestic eggs,
##        other vegetables,
##        whipped/sour cream}       => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [102] {butter,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [103] {butter,
##        domestic eggs,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [104] {butter,
##        fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [105] {butter,
##        soda,
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [106] {bottled water,
##        butter,
##        citrus fruit,
##        other vegetables}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [107] {newspapers,
##        rolls/buns,
##        soda,
##        whole milk}               => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [108] {domestic eggs,
##        other vegetables,
##        pip fruit,
```

```
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [109] {citrus fruit,
##        domestic eggs,
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [110] {domestic eggs,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [111] {domestic eggs,
##        other vegetables,
##        tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [112] {citrus fruit,
##        domestic eggs,
##        other vegetables,
##        tropical fruit}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [113] {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                   => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [114] {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream,
##        whole milk}               => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [115] {fruit/vegetable juice,
##        pip fruit,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [116] {citrus fruit,
##        fruit/vegetable juice,
##        other vegetables,
##        soda}                     => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [117] {citrus fruit,
##        pastry,
##        rolls/buns,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [118] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001220132  1.0000000 0.001220132  5.168156
## [119] {bottled water,
##        other vegetables,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [120] {pastry,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [121] {root vegetables,
##        sausage,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [122] {rolls/buns,
```

```
##        root vegetables,
##        sausage,
##        tropical fruit}           => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [123] {bottled water,
##        rolls/buns,
##        root vegetables,
##        tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [124] {oil,
##        other vegetables,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [125] {oil,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [126] {oil,
##        other vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [127] {butter,
##        domestic eggs,
##        other vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [128] {citrus fruit,
##        root vegetables,
##        whipped/sour cream,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [129] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
```

```
arules::inspect(subset(groc_rules, subset=lift > 10 & confidence > 0.5))
```

```
##     lhs                          rhs            support     confidence
## [1] {liquor,red/blush wine} => {bottled beer} 0.001931876 0.9047619
##     coverage    lift     count
## [1] 0.002135231 11.23527 19
```

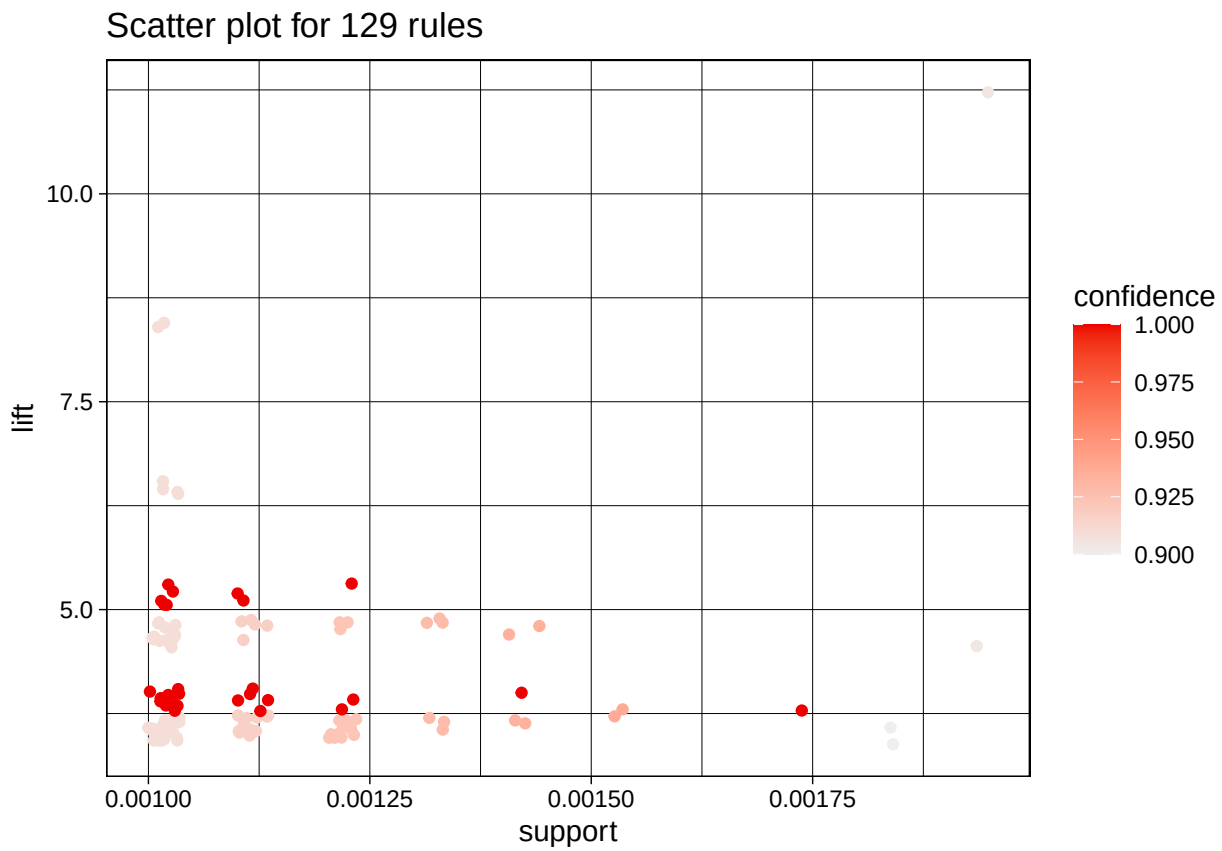
```
plot(groc_rules, method='graph')
```

```
## Warning: Too many rules supplied. Only plotting the best 100 rules using lift
## (change control parameter max if needed)
```

Look at the output... so many rules! There are 128 rules generated with support at .001, confidence at .9, and max length at 50. Choose a subset.

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

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




```
plot(groc_rules, method='two-key plot')
```

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

Plot all the rules in (support, confidence) space; notice that high lift rules tend to have low support.

Look at subsets driven by the plot.

```
inspect(subset(groc_rules, support > 0.035))
inspect(subset(groc_rules, confidence > 0.7))
```

```
##       lhs                           rhs                    support confidence    coverage      lift c
## [1]   {liquor,
##        red/blush wine}          => {bottled beer}     0.001931876  0.9047619 0.002135231 11.235269
## [2]   {cereals,
##        curd}                    => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [3]   {bottled beer,
##        soups}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [4]   {house keeping products,
##        whipped/sour cream}      => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [5]   {pastry,
##        sweet spreads}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [6]   {rice,
##        sugar}                   => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [7]   {bottled water,
##        rice}                    => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [8]   {canned fish,
##        hygiene articles}        => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [9]   {grapes,
##        onions}                  => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
```

```
## [10]  {hard cheese,
##        oil}                    => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [11]  {butter,
##        rice,
##        root vegetables}        => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [12]  {fruit/vegetable juice,
##        herbs,
##        whole milk}             => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [13]  {citrus fruit,
##        herbs,
##        tropical fruit}         => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [14]  {flour,
##        root vegetables,
##        whipped/sour cream}     => {whole milk}       0.001728521  1.0000000 0.001728521  3.913649
## [15]  {butter,
##        domestic eggs,
##        soft cheese}            => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [16]  {soft cheese,
##        tropical fruit,
##        whipped/sour cream}     => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [17]  {root vegetables,
##        soft cheese,
##        whipped/sour cream}     => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [18]  {citrus fruit,
##        root vegetables,
##        soft cheese}            => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [19]  {frankfurter,
##        frozen meals,
##        tropical fruit}         => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [20]  {frankfurter,
##        frozen meals,
##        tropical fruit}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [21]  {butter,
##        frozen meals,
##        tropical fruit}         => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [22]  {hard cheese,
##        tropical fruit,
##        whipped/sour cream}     => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [23]  {butter milk,
##        pork,
##        whole milk}             => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [24]  {butter milk,
##        fruit/vegetable juice,
##        pip fruit}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [25]  {frankfurter,
##        root vegetables,
##        sliced cheese}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [26]  {butter,
##        sliced cheese,
##        whipped/sour cream}     => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [27]  {coffee,
##        oil,
##        yogurt}                 => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [28]  {napkins,
```

```
##        onions,
##        root vegetables}          => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [29]  {berries,
##        butter,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [30]  {hamburger meat,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [31]  {butter,
##        hygiene articles,
##        napkins}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [32]  {butter,
##        hygiene articles,
##        pip fruit}                => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [33]  {butter,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [34]  {domestic eggs,
##        hygiene articles,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [35]  {hygiene articles,
##        root vegetables,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [36]  {hygiene articles,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [37]  {cream cheese,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [38]  {cream cheese,
##        other vegetables,
##        sugar}                    => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [39]  {curd,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [40]  {citrus fruit,
##        domestic eggs,
##        sugar}                    => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [41]  {domestic eggs,
##        sugar,
##        tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [42]  {domestic eggs,
##        sugar,
##        yogurt}                   => {whole milk}       0.001423488  0.9333333 0.001525165  3.652739
## [43]  {root vegetables,
##        sugar,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [44]  {pork,
##        rolls/buns,
##        waffles}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [45]  {long life bakery product,
##        napkins,
##        whipped/sour cream}       => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [46]  {long life bakery product,
```

```
##        napkins,
##        tropical fruit}           => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [47]  {butter,
##        long life bakery product,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [48]  {dessert,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [49]  {cream cheese,
##        domestic eggs,
##        napkins}                  => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [50]  {butter,
##        cream cheese,
##        root vegetables}          => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [51]  {butter,
##        cream cheese,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [52]  {cream cheese,
##        pip fruit,
##        whipped/sour cream}       => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [53]  {cream cheese,
##        pip fruit,
##        sausage}                  => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [54]  {citrus fruit,
##        cream cheese,
##        root vegetables}          => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [55]  {butter,
##        root vegetables,
##        white bread}              => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [56]  {butter,
##        coffee,
##        whipped/sour cream}       => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [57]  {coffee,
##        domestic eggs,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [58]  {butter,
##        curd,
##        domestic eggs}            => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [59]  {butter,
##        citrus fruit,
##        curd}                     => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [60]  {bottled beer,
##        domestic eggs,
##        margarine}                => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [61]  {brown bread,
##        pip fruit,
##        whipped/sour cream}       => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [62]  {domestic eggs,
##        fruit/vegetable juice,
##        margarine}                => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [63]  {butter,
##        pip fruit,
##        whipped/sour cream}       => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [64]  {butter,
```

```
##        soda,
##        whipped/sour cream}       => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [65]  {butter,
##        pastry,
##        pip fruit}                => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [66]  {domestic eggs,
##        tropical fruit,
##        whipped/sour cream}       => {whole milk}       0.001830198  0.9000000 0.002033554  3.522284
## [67]  {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001931876  0.9047619 0.002135231  4.675950
## [68]  {other vegetables,
##        rice,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [69]  {rice,
##        root vegetables,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001321810  0.9285714 0.001423488  4.799002
## [70]  {herbs,
##        other vegetables,
##        root vegetables,
##        tropical fruit}           => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [71]  {grapes,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [72]  {frozen meals,
##        pip fruit,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [73]  {hard cheese,
##        other vegetables,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [74]  {ham,
##        pip fruit,
##        tropical fruit,
##        yogurt}                   => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [75]  {ham,
##        pip fruit,
##        tropical fruit,
##        whole milk}               => {other vegetables} 0.001118454  1.0000000 0.001118454  5.168156
## [76]  {butter,
##        sliced cheese,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [77]  {butter,
##        sliced cheese,
##        tropical fruit,
##        whole milk}               => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [78]  {oil,
##        root vegetables,
##        tropical fruit,
```

```
##        yogurt}                 => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [79]  {oil,
##        root vegetables,
##        tropical fruit,
##        yogurt}                 => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [80]  {oil,
##        other vegetables,
##        root vegetables,
##        yogurt}                 => {whole milk}       0.001423488  1.0000000 0.001423488  3.913649
## [81]  {oil,
##        root vegetables,
##        whole milk,
##        yogurt}                 => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
## [82]  {other vegetables,
##        root vegetables,
##        waffles,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [83]  {cream cheese,
##        curd,
##        other vegetables,
##        whipped/sour cream}     => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [84]  {citrus fruit,
##        cream cheese,
##        whipped/sour cream,
##        whole milk}             => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [85]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        root vegetables}        => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [86]  {cream cheese,
##        other vegetables,
##        pip fruit,
##        yogurt}                 => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [87]  {butter,
##        tropical fruit,
##        white bread,
##        yogurt}                 => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [88]  {butter,
##        other vegetables,
##        tropical fruit,
##        white bread}            => {yogurt}           0.001016777  0.9090909 0.001118454  6.516698
## [89]  {butter,
##        other vegetables,
##        root vegetables,
##        white bread}            => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [90]  {butter,
##        root vegetables,
##        white bread,
##        whole milk}             => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [91]  {citrus fruit,
##        frozen vegetables,
##        other vegetables,
##        yogurt}                 => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [92]  {beef,
```

```
##        rolls/buns,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001321810  0.9285714 0.001423488  3.634103
## [93]  {curd,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [94]  {citrus fruit,
##        curd,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [95]  {butter,
##        napkins,
##        other vegetables,
##        whipped/sour cream}       => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [96]  {butter,
##        other vegetables,
##        pork,
##        whipped/sour cream}       => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [97]  {butter,
##        other vegetables,
##        pork,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [98]  {frankfurter,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [99]  {brown bread,
##        other vegetables,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [100] {brown bread,
##        other vegetables,
##        rolls/buns,
##        root vegetables}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [101] {butter,
##        domestic eggs,
##        other vegetables,
##        whipped/sour cream}       => {whole milk}       0.001220132  1.0000000 0.001220132  3.913649
## [102] {butter,
##        domestic eggs,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [103] {butter,
##        domestic eggs,
##        root vegetables,
##        yogurt}                   => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [104] {butter,
##        fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream}       => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [105] {butter,
##        soda,
##        whipped/sour cream,
```

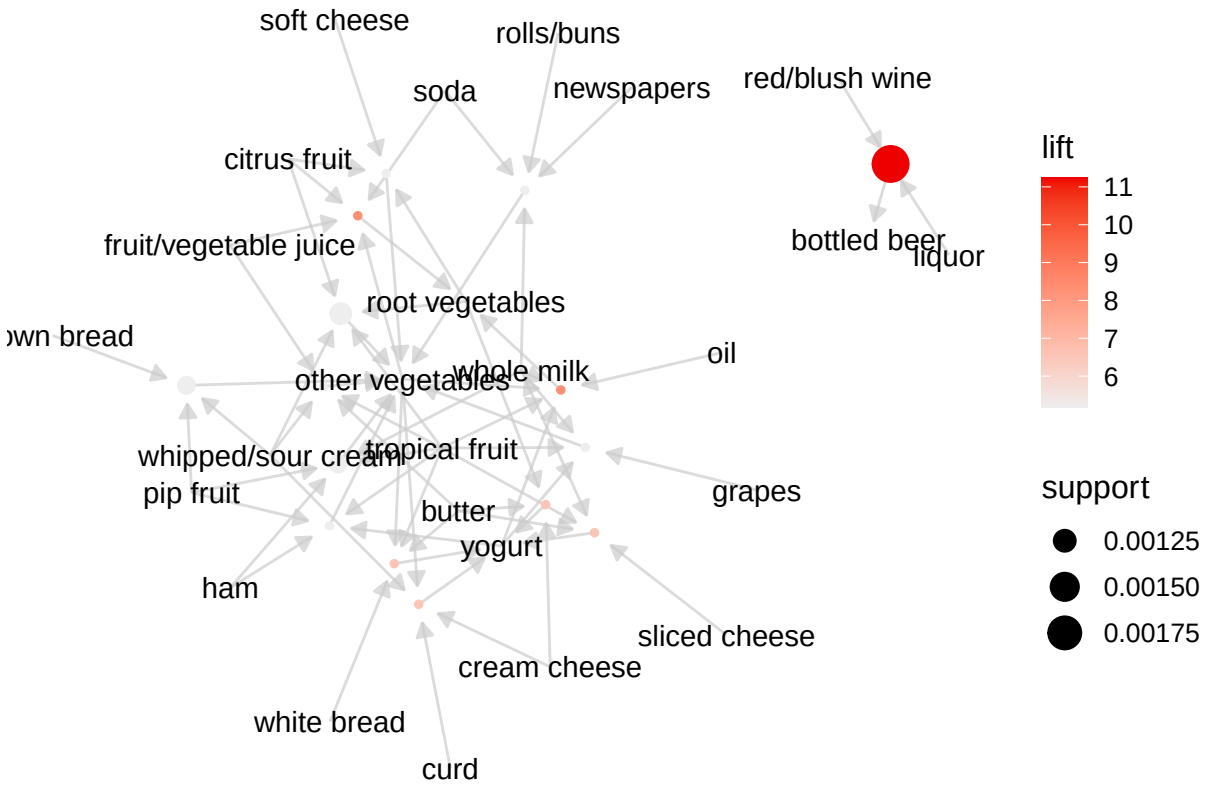
```
##        whole milk}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [106] {bottled water,
##        butter,
##        citrus fruit,
##        other vegetables}        => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [107] {newspapers,
##        rolls/buns,
##        soda,
##        whole milk}              => {other vegetables} 0.001016777  1.0000000 0.001016777  5.168156
## [108] {domestic eggs,
##        other vegetables,
##        pip fruit,
##        whipped/sour cream}      => {whole milk}       0.001220132  0.9230769 0.001321810  3.612599
## [109] {citrus fruit,
##        domestic eggs,
##        whipped/sour cream,
##        whole milk}              => {other vegetables} 0.001220132  0.9230769 0.001321810  4.770605
## [110] {domestic eggs,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                  => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [111] {domestic eggs,
##        other vegetables,
##        tropical fruit,
##        whipped/sour cream}      => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [112] {citrus fruit,
##        domestic eggs,
##        other vegetables,
##        tropical fruit}          => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [113] {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream,
##        yogurt}                  => {other vegetables} 0.001118454  0.9166667 0.001220132  4.737476
## [114] {fruit/vegetable juice,
##        tropical fruit,
##        whipped/sour cream,
##        whole milk}              => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [115] {fruit/vegetable juice,
##        pip fruit,
##        root vegetables,
##        yogurt}                  => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [116] {citrus fruit,
##        fruit/vegetable juice,
##        other vegetables,
##        soda}                    => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [117] {citrus fruit,
##        pastry,
##        rolls/buns,
##        whipped/sour cream}      => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [118] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whipped/sour cream}      => {other vegetables} 0.001220132  1.0000000 0.001220132  5.168156
## [119] {bottled water,
```

```
##        other vegetables,
##        pip fruit,
##        root vegetables}          => {whole milk}       0.001118454  1.0000000 0.001118454  3.913649
## [120] {pastry,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [121] {root vegetables,
##        sausage,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001525165  0.9375000 0.001626843  3.669046
## [122] {rolls/buns,
##        root vegetables,
##        sausage,
##        tropical fruit}           => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [123] {bottled water,
##        rolls/buns,
##        root vegetables,
##        tropical fruit}           => {whole milk}       0.001118454  0.9166667 0.001220132  3.587512
## [124] {oil,
##        other vegetables,
##        root vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  1.0000000 0.001016777  3.913649
## [125] {oil,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [126] {oil,
##        other vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {root vegetables}  0.001016777  0.9090909 0.001118454  8.340400
## [127] {butter,
##        domestic eggs,
##        other vegetables,
##        tropical fruit,
##        yogurt}                   => {whole milk}       0.001016777  0.9090909 0.001118454  3.557863
## [128] {citrus fruit,
##        root vegetables,
##        whipped/sour cream,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001016777  0.9090909 0.001118454  4.698323
## [129] {citrus fruit,
##        root vegetables,
##        tropical fruit,
##        whole milk,
##        yogurt}                   => {other vegetables} 0.001423488  0.9333333 0.001525165  4.823612
```

```
plot(head(sort(groc_rules, by="lift"), 15),
     method="graph", control=list(cex=.9))
```

```
## Warning: Unknown control parameters: cex
```

```
## Available control parameters (with default values):
## layout    =  list(fun = function (graph, dim = 2, ...)  {     if ("layout" %in% graph_attr_names(gra
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
## verbose   =  FALSE
```



This plot shows the mapping from 15 different rules picked up from groc_rules.

The plot shows the following generalizations and associations:

Whole milk (most frequently bought item), yogurt, other vegetables, and root vegetables are big centers of the plot; therefore, they are very prevalent in association rule mappings. It means that these products (dairy and vegetables) are put in the basket often in combination with of other different products.

People are also more likely to buy bottled beer if they purchased red/blush wine or liquor; which (practically) makes sense. (Association between alcoholic beverages)

Newspapers, sodas, cur, and ham (for instance) are just some items that can be seen on the outer part of the plot. This means that these items are not very well associated with the other items in the groceries basket.