Stats Takehome project

Lai Jiang

8/17/2020

##Visual Story Telling Part 1: Green Building

```
library(corrplot)
```

```
## corrplot 0.84 loaded
```

```
library(LICORS)
library(RColorBrewer)
library(mosaic)
```

```
## Loading required package: dplyr
```

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

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

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

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

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

```
## Loading required package: ggplot2
```

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

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

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

```
##
## New to ggformula?  Try the tutorials:
##  learnr::run_tutorial("introduction", package = "ggformula")
##  learnr::run_tutorial("refining", package = "ggformula")

## Loading required package: mosaicData

## Loading required package: Matrix

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

##
## The 'mosaic' package masks several functions from core packages in order to add
## additional features.  The original behavior of these functions should not be affected by this.
##
## Note: If you use the Matrix package, be sure to load it BEFORE loading mosaic.
##
## Have you tried the ggformula package for your plots?

##
## Attaching package: 'mosaic'

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

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

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

## The following objects are masked from 'package:stats':
##
##     binom.test, cor, cor.test, cov, fivenum, IQR, median, prop.test,
##     quantile, sd, t.test, var

## The following objects are masked from 'package:base':
##
##     max, mean, min, prod, range, sample, sum
```

```
library(tidyverse)
```

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

## v tibble  3.0.1     v purrr   0.3.4
## v tidyr   1.1.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.5.0
```

```
## -- Conflicts ---------------------------------------- tidyverse_conflicts() --
## x mosaic::count()            masks dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x mosaic::do()               masks dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x dplyr::filter()            masks stats::filter()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x dplyr::lag()               masks stats::lag()
## x tidyr::pack()              masks Matrix::pack()
## x mosaic::stat()             masks ggplot2::stat()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
```

```
urlfile="https://raw.githubusercontent.com/jgscott/STA380/master/data/greenbuildings.csv"
greenbuildings = read.csv(url(urlfile))
```

```
greenbuildings$green_rating<-as.factor(greenbuildings$green_rating)
##median rent for green building
median(greenbuildings$Rent[greenbuildings$green_rating==1])
```

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

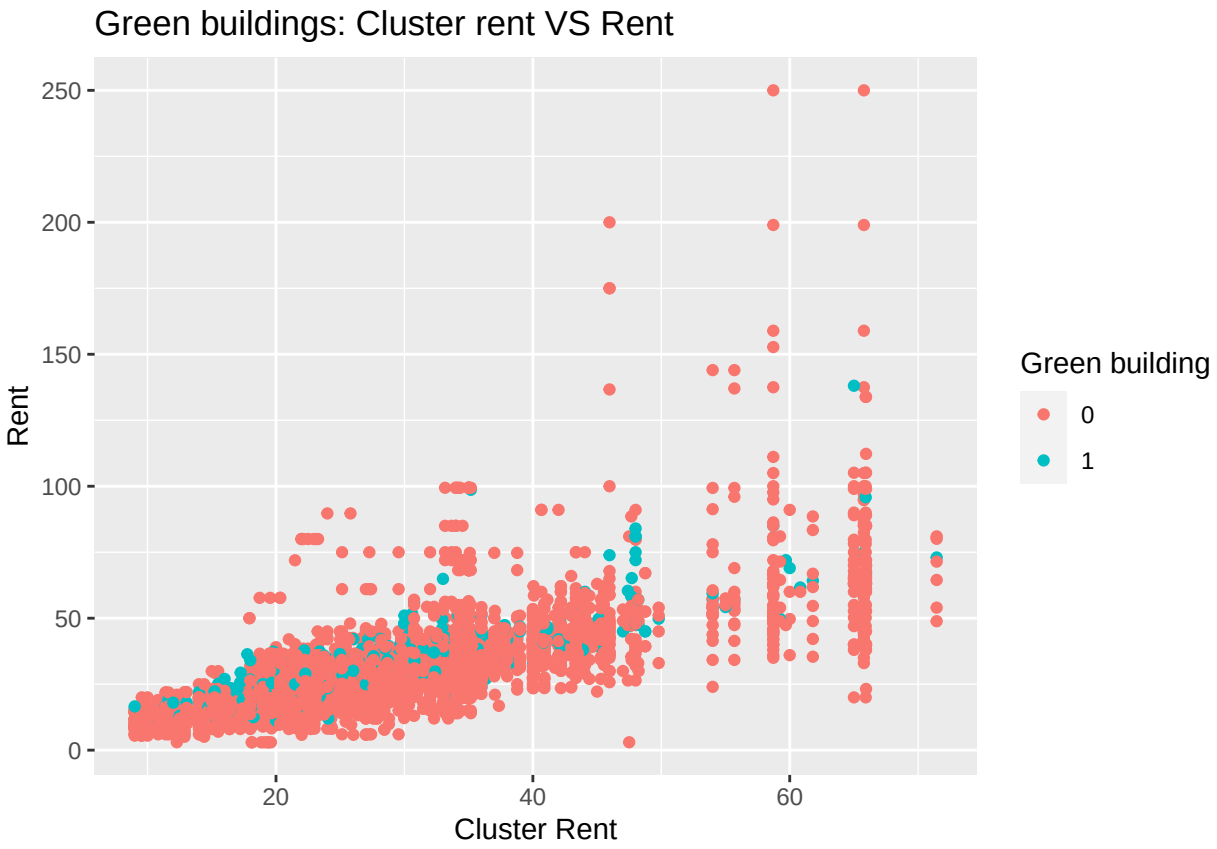
```
##median rent for non-green building
median(greenbuildings$Rent[greenbuildings$green_rating==0])
```

```
## [1] 25
```

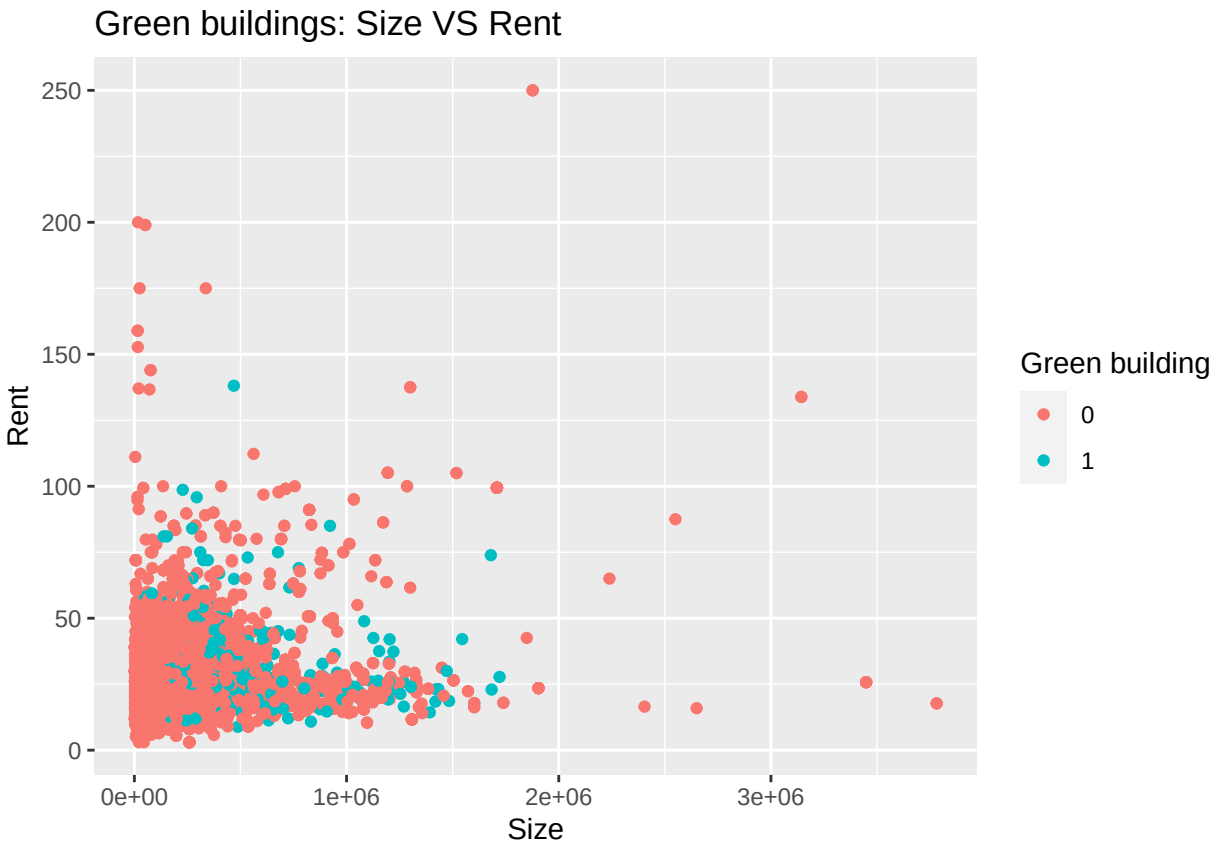
##indeed 2.6 dollar more on average per square ft

visualizations

```
ggplot(data=greenbuildings) +
  geom_point(mapping=aes(x=cluster_rent, y=Rent, color=green_rating)) +
  labs(x="Cluster Rent", y='Rent', title = 'Green buildings: Cluster rent VS Rent',
       color='Green building')
```

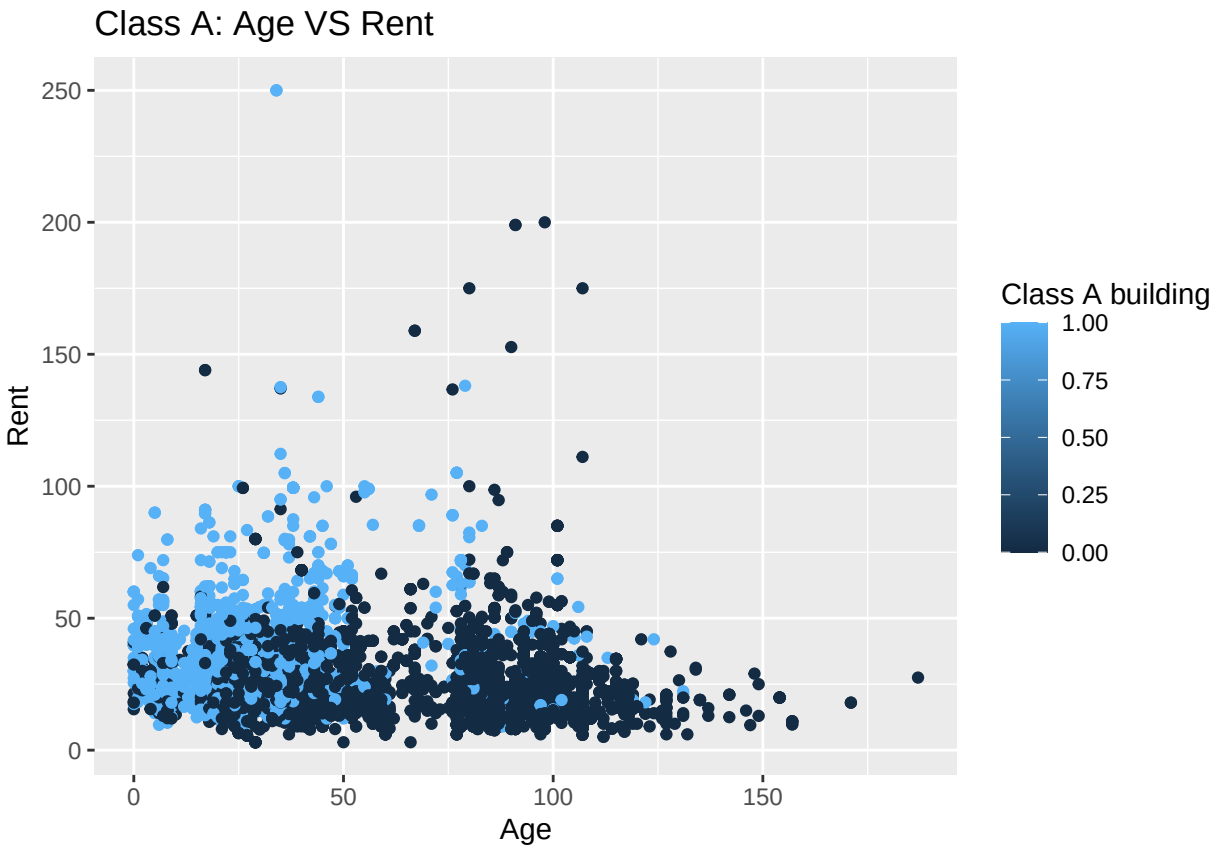



```
ggplot(data=greenbuildings) +
  geom_point(mapping=aes(x=size, y=Rent, color=green_rating)) +
  labs(x="Size", y='Rent', title = 'Green buildings: Size VS Rent',
       color='Green building')
```




```
ggplot(data=greenbuildings) +
  geom_point(mapping=aes(x=leasing_rate, y=Rent, color=green_rating)) +
  labs(x="leasing_rate", y='Rent', title = 'Green buildings: leasing rate VS Rent',
       color='Green building')
```


```
ggplot(data=greenbuildings) +
  geom_point(mapping=aes(x=age, y=Rent, colour=class_a))+
  labs(x="Age", y='Rent', title = 'Class A: Age VS Rent',
       color='Class A building')
```



Observaton:

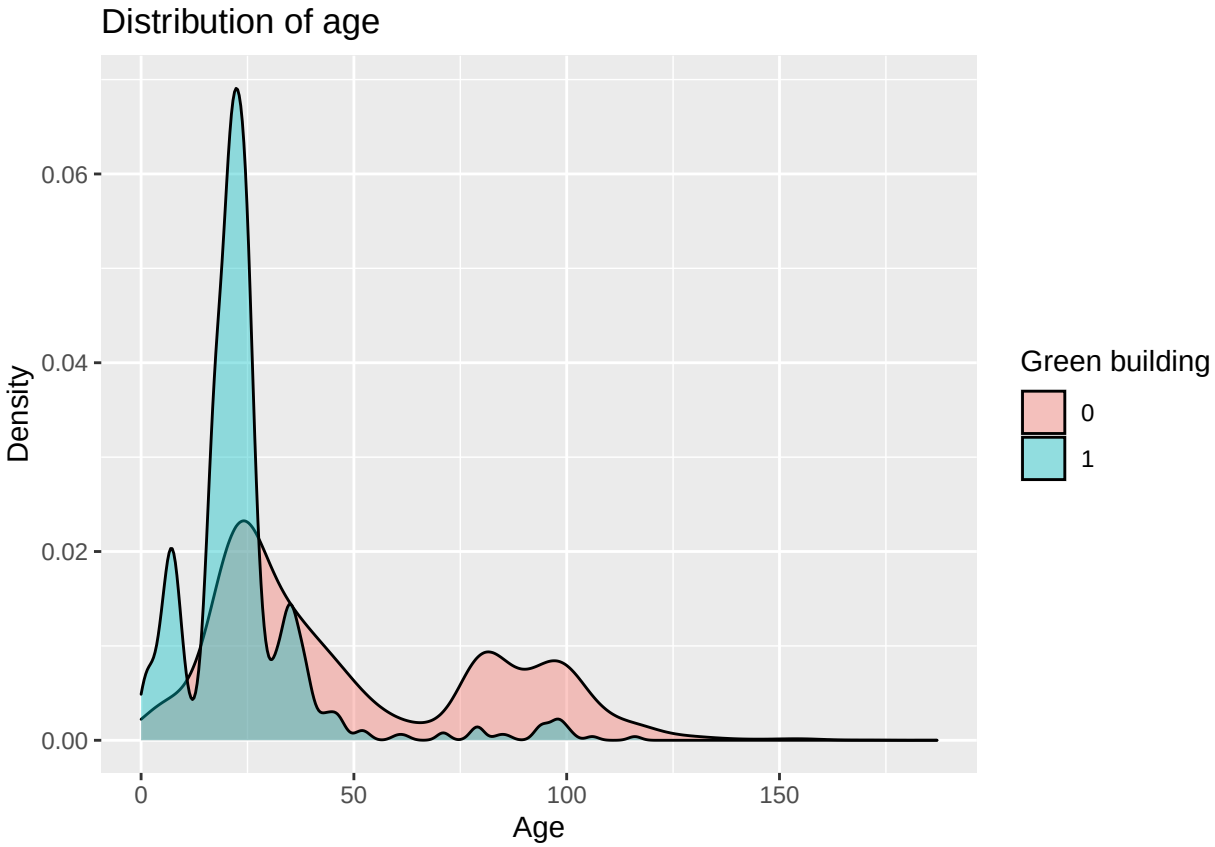
Class A building for the same age old are charged a higher rent than Non-class A buildings

Rent and Cluster rent is postively correlated.

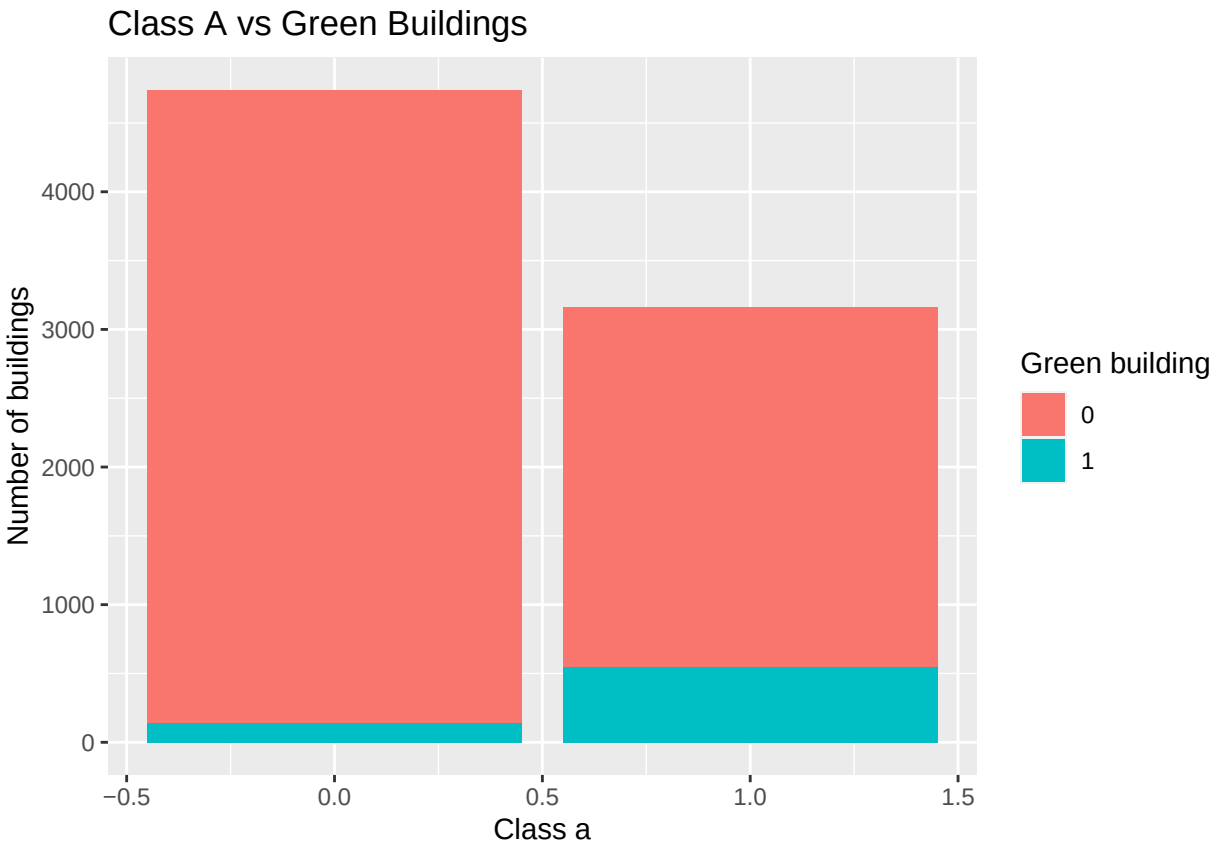
Lease rate is poitvely correlated with rent.

Size is correlated with Rent as expected

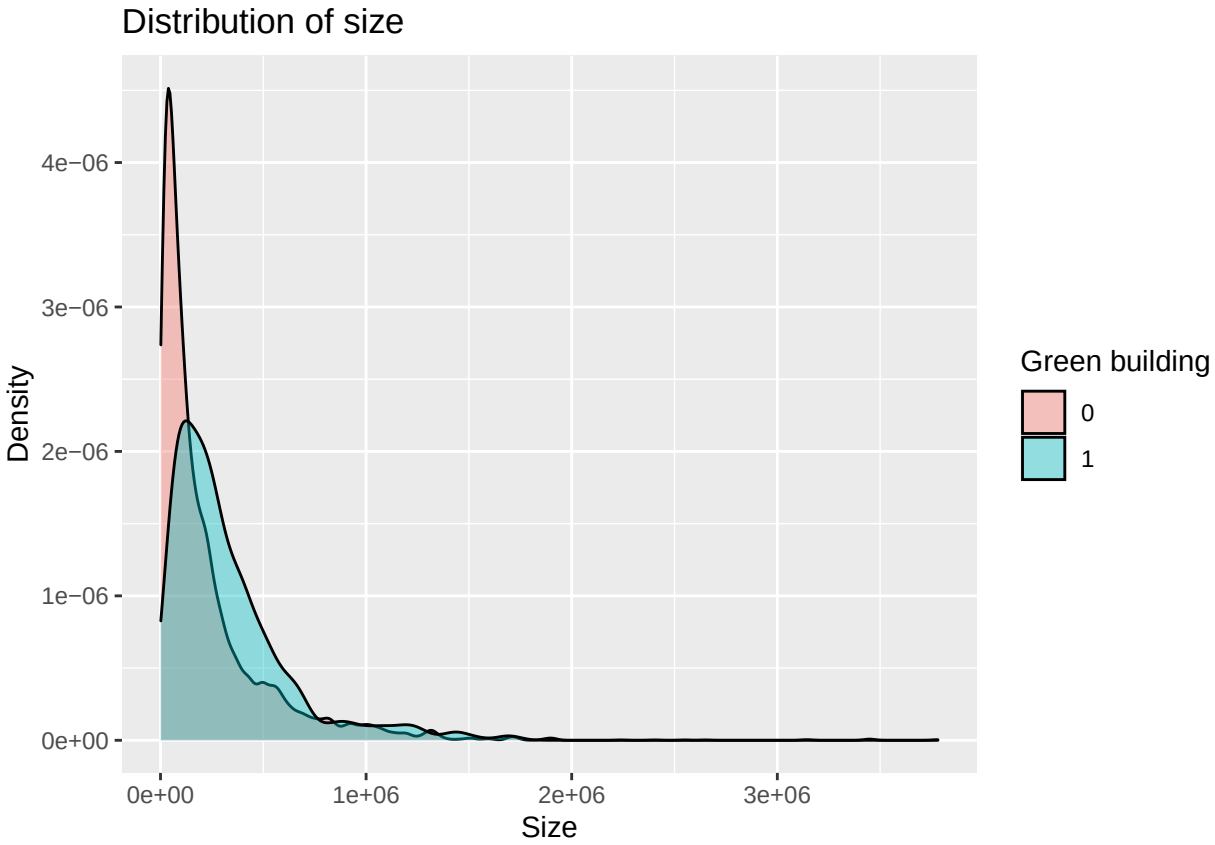
```
g = ggplot(greenbuildings, aes(x=age))
g + geom_density(aes(fill=factor(green_rating)), alpha=0.4)+
  labs(x="Age", y='Density', title = 'Distribution of age',
       fill='Green building')
```




```
ggplot(greenbuildings, aes(class_a, ..count..)) + geom_bar(aes(fill = green_rating), position = "stack"
  labs(x="Class a", y='Number of buildings', title = 'Class A vs Green Buildings',
       fill='Green building')
```

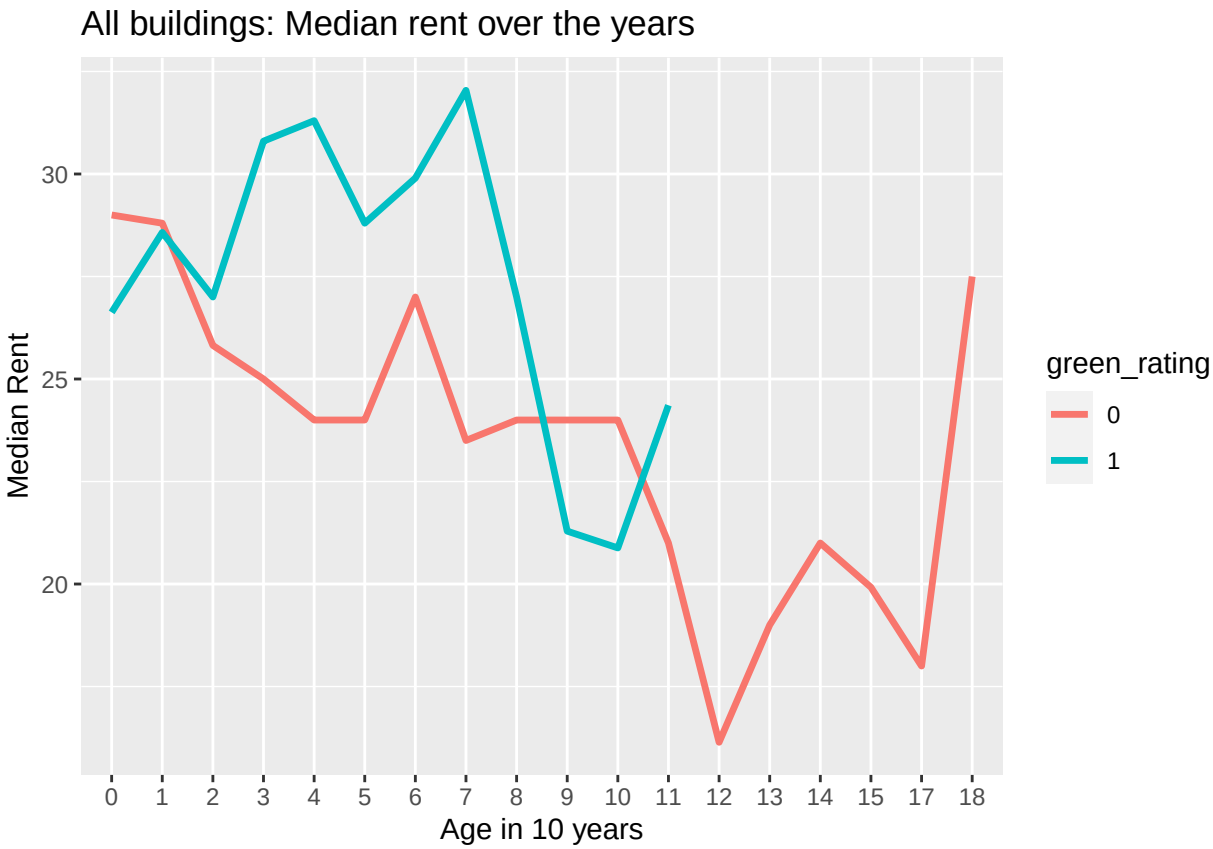



```
g = ggplot(greenbuildings, aes(x=size))
g + geom_density(aes(fill=factor(green_rating)), alpha=0.4)+
  labs(x="Size", y='Density', title = 'Distribution of size',
       fill='Green building')
```



Most of the green buildings are built later than non-green buildings ## More green buildings are in the class A buildings ## more green builings are smaller in size <50

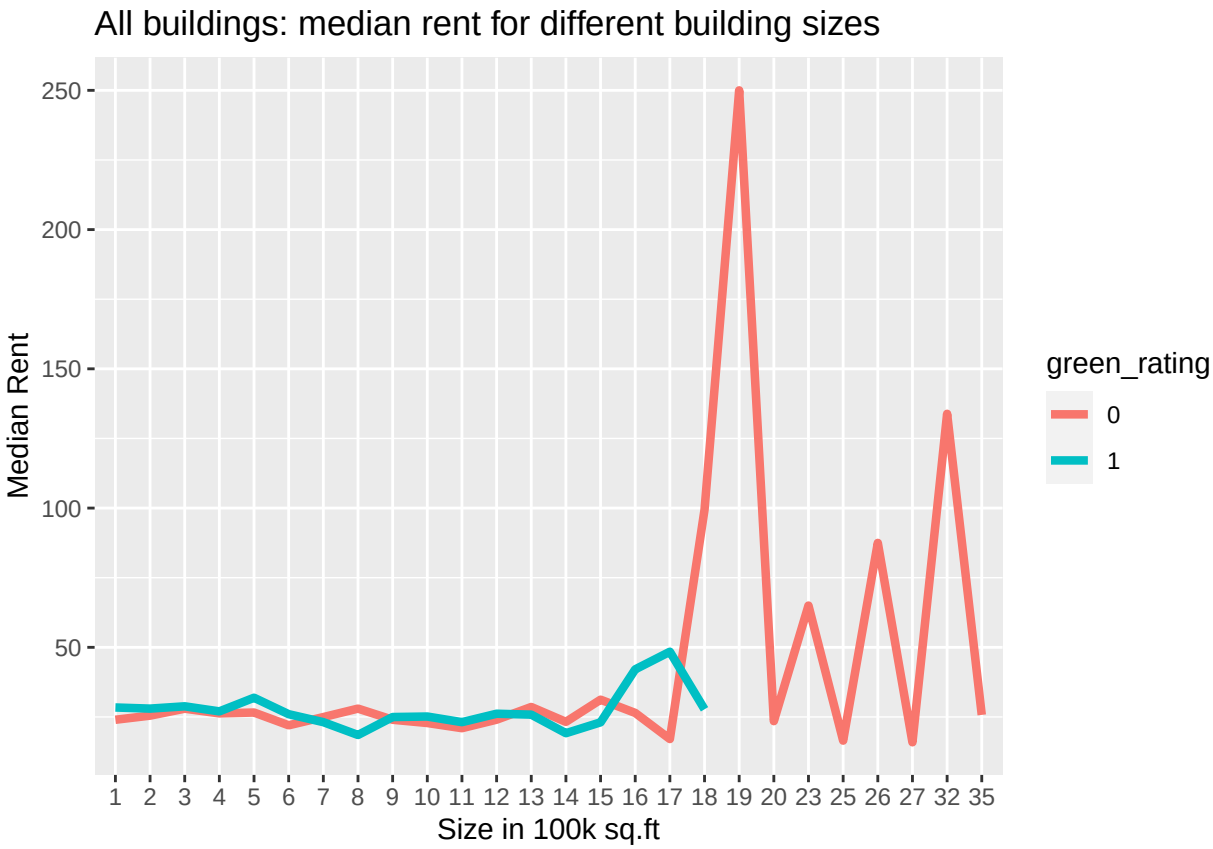
```
greenbuildings$age_cut <- cut(greenbuildings$age, breaks = c(0, seq(10, 190, by = 10)), labels = 0:18,r
medians <- aggregate(Rent~ age_cut + green_rating, greenbuildings, median)
ggplot(data = medians, mapping = aes(y = Rent, x = age_cut ,group = green_rating, color=green_rating)) +
   geom_line(size=1.2) +
  labs(x="Age in 10 years", y='Median Rent', title = 'All buildings: Median rent over the years',
       fill='Green building')
```



again, green buildings are much newer and within 11 years of building age and charge a much higher rent on average between building age 10 to building age 90 years

Size cut in 100k buckets

```
greenbuildings$size_cut <- cut(greenbuildings$size, breaks = c(0, seq(10, 3781045, by = 100000)), labels
medians <- aggregate(Rent ~ size_cut + green_rating, greenbuildings, median)
ggplot(data = medians, mapping = aes(y = Rent, x = size_cut ,group = green_rating, color=green_rating))
   geom_line(size=1.5) +
  labs(x="Size in 100k sq.ft", y='Median Rent', title = 'All buildings: median rent for different build
       fill='Green building')
```



once the building increase pass 1800K sqft. the non_green buildings has much higher median rent.

```
nonclass_a <- subset(greenbuildings, greenbuildings$class_a != 1)
nonclass_a$age_cut <- cut(nonclass_a$age, breaks = c(0, seq(10, 190, by = 10)), labels = 0:18,right=FALS
medians <- aggregate(Rent~ age_cut + green_rating, nonclass_a, median)
ggplot(data = medians, mapping = aes(y = Rent, x = age_cut ,group = green_rating, color=green_rating)) +
   geom_line(size=1.5)+
  labs(x="Age in 10 years", y='Median Rent', title = 'Non-Class A buildings: Median rent over the years
       fill='Green building')
```

Non-class A green buildings only have a higher median rent than non-class A non green building if it's 20-30 years older or 40-90 years old.

Size in 100k

```
nonclass_a$size_cut <- cut(nonclass_a$size, breaks = c(0, seq(10, 3781045, by = 100000)), labels = 0:37
medians <- aggregate(Rent ~ size_cut + green_rating, nonclass_a, median)
ggplot(data = medians, mapping = aes(y = Rent, x = size_cut ,group = green_rating, colour=green_rating)
   geom_line(size=1.5)+
  labs(x="Size in 100k sq.ft", y='Median Rent', title = 'Non-class A buildings: median rent for differe
       fill='Green building')
```

In non-class A buildings, median rent of green buildings is lower than non-green ones

```
data_size <- subset(greenbuildings, greenbuildings$size > 200000 & greenbuildings$size < 300000)
data_size <- subset(greenbuildings, greenbuildings$class_a == 1)
data_size_class <- subset(greenbuildings, nonclass_a$size > 200000 & nonclass_a$size < 300000)
paste("Median leasing rate for class a buildings of sizes ranging from 200k to 300k sq.ft ",
median(data_size$leasing_rate))
```

```
## [1] "Median leasing rate for class a buildings of sizes ranging from 200k to 300k sq.ft  92.89"
```

```
medians <- aggregate(Rent~ age_cut + green_rating, data_size, median)
medians_1 <- subset(medians, medians$green_rating == 1)
rent_1<-medians_1[1:5,]$Rent
medians_0 <- subset(medians, medians$green_rating == 0)
rent_0<-medians_0[1:5,]$Rent
paste("Difference in rent for the first 5 years class a buildings: ",
(sum(rent_1,na.rm = T) - sum(rent_0, na.rm = T)) / 5)
```

```
## [1] "Difference in rent for the first 5 years class a buildings:  3.848"
```

```
medians <- aggregate(Rent~ age_cut + green_rating, data_size_class, median)
medians_1 <- subset(medians, medians$green_rating == 1)
rent_1<-medians_1[1:5,]$Rent
medians_0 <- subset(medians, medians$green_rating == 0)
rent_0<-medians_0[1:5,]$Rent
```

```
paste("Difference in rent for the first 5 years for non-class a buildings: ",
(sum(rent_1,na.rm = T) - sum(rent_0, na.rm = T)) / 5 )
```

```
## [1] "Difference in rent for the first 5 years for non-class a buildings:  -1.167"
```

the analyst fails to account other factors such as building class, size in his analysis.

rent differs at different age and size, since we are looking at 250K sq.ft investment. We should focus on 200-300K sqft

We will use average 5 years return to arrive at final recommendation since data does not having information about class A buildings size from 200K to 300K sq ft.

```
paste("If we build a class a green building and if we assume 91.6% occpancy rate, it is expected to rec
```

```
## [1] "If we build a class a green building and if we assume 91.6% occpancy rate, it is expected to re
```

Recommendation:

Do not invest in non Class A building since difference in rent for first 5 years for green vs non-green in non-class A is -1.167

expect a 92.89% occupancy rate

Difference in rent for the first 5 years for green vs non-green class a buildings is 3.848

we recooperate the 5mil extra investment for green certification in 5.6 years.

```
library(ggplot2)
library(ggpubr)
urlfile="https://raw.githubusercontent.com/jgscott/STA380/master/data/ABIA.csv"
airport = read.csv(url(urlfile),header = TRUE)
head(airport)
```

```
##   Year Month DayofMonth DayOfWeek DepTime CRSDepTime ArrTime CRSArrTime
## 1 2008     1          1         2     120       1935     309       2130
## 2 2008     1          1         2     555        600     826        835
## 3 2008     1          1         2     600        600     728        729
## 4 2008     1          1         2     601        605     727        750
## 5 2008     1          1         2     601        600     654        700
## 6 2008     1          1         2     636        645     934        932
##   UniqueCarrier FlightNum TailNum ActualElapsedTime CRSElapsedTime AirTime
## 1            9E      5746  84129E                109            115      88
```

```
## 2            AA      1614  N438AA               151            155     133
## 3            YV      2883  N922FJ               148            149     125
## 4            9E      5743  89189E                86            105      70
## 5            AA      1157  N4XAAA                53             60      38
## 6            NW      1674   N967N               178            167     145
##   ArrDelay DepDelay Origin Dest Distance TaxiIn TaxiOut Cancelled
## 1      339      345    MEM  AUS      559      3      18         0
## 2       -9       -5    AUS  ORD      978      7      11         0
## 3       -1        0    AUS  PHX      872      7      16         0
## 4      -23       -4    AUS  MEM      559      4      12         0
## 5       -6        1    AUS  DFW      190      5      10         0
## 6        2       -9    AUS  MSP     1042     11      22         0
##   CancellationCode Diverted CarrierDelay WeatherDelay NASDelay SecurityDelay
## 1                        0          339            0        0             0
## 2                        0           NA           NA       NA            NA
## 3                        0           NA           NA       NA            NA
## 4                        0           NA           NA       NA            NA
## 5                        0           NA           NA       NA            NA
## 6                        0           NA           NA       NA            NA
##   LateAircraftDelay
## 1                 0
## 2                NA
## 3                NA
## 4                NA
## 5                NA
## 6                NA
```

```
attach(airport)
colSums(is.na(airport))
```

```
##              Year             Month        DayofMonth         DayOfWeek
##                 0                 0                 0                 0
##           DepTime        CRSDepTime           ArrTime        CRSArrTime
##              1413                 0              1567                 0
##     UniqueCarrier         FlightNum           TailNum ActualElapsedTime
##                 0                 0                 0              1601
##    CRSElapsedTime           AirTime          ArrDelay          DepDelay
##                11              1601              1601              1413
##            Origin              Dest          Distance            TaxiIn
##                 0                 0                 0              1567
##           TaxiOut         Cancelled  CancellationCode          Diverted
##              1419                 0                 0                 0
##      CarrierDelay      WeatherDelay          NASDelay     SecurityDelay
##             79513             79513             79513             79513
## LateAircraftDelay
##             79513
```

```
colnames(airport)
```

```
##  [1] "Year"              "Month"             "DayofMonth"
##  [4] "DayOfWeek"         "DepTime"           "CRSDepTime"
##  [7] "ArrTime"           "CRSArrTime"        "UniqueCarrier"
## [10] "FlightNum"         "TailNum"           "ActualElapsedTime"
```

```
## [13] "CRSElapsedTime"    "AirTime"           "ArrDelay"
## [16] "DepDelay"          "Origin"            "Dest"
## [19] "Distance"          "TaxiIn"            "TaxiOut"
## [22] "Cancelled"         "CancellationCode"  "Diverted"
## [25] "CarrierDelay"      "WeatherDelay"      "NASDelay"
## [28] "SecurityDelay"     "LateAircraftDelay"
```

```
dim(airport)
```

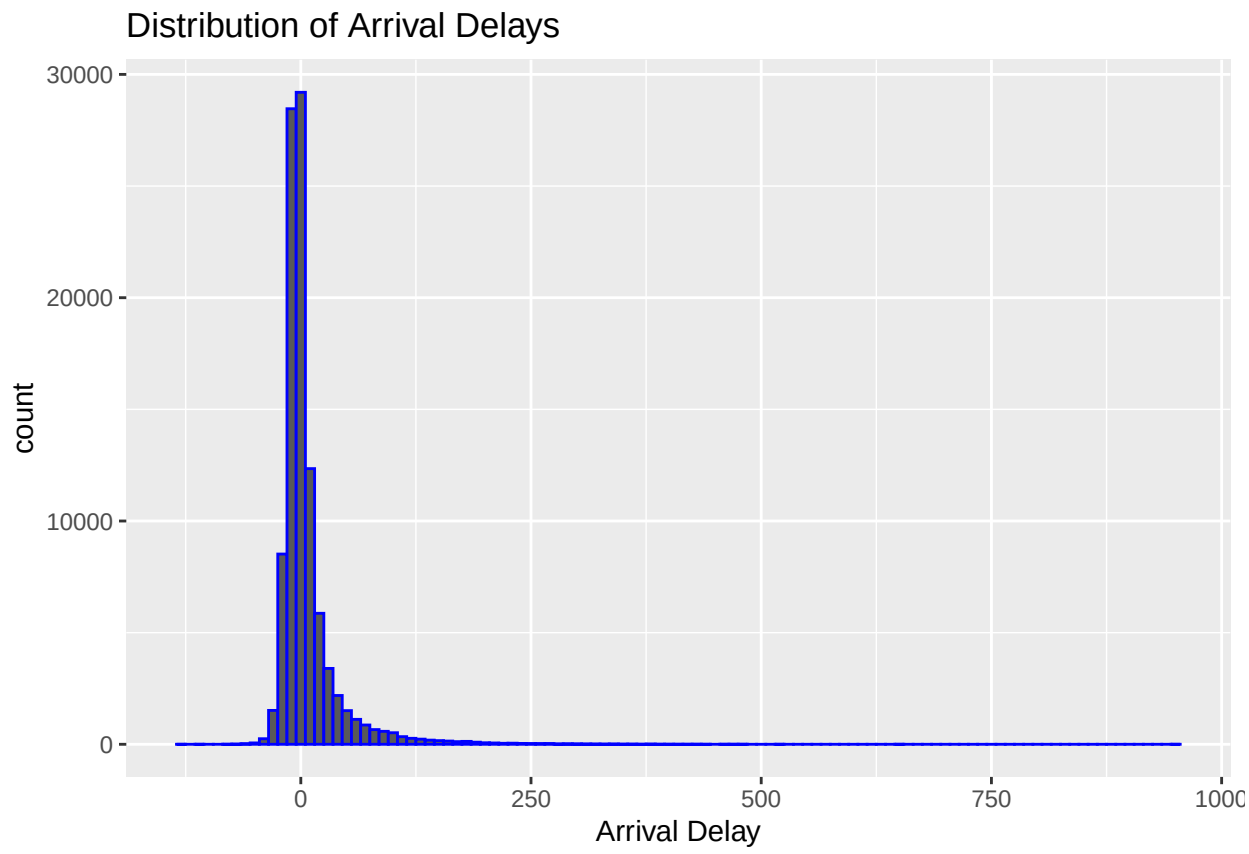
```
## [1] 99260    29
```

```
airport[is.na(airport)] <- 0
colSums(is.na(airport))
```

```
##              Year             Month        DayofMonth         DayOfWeek
##                 0                 0                 0                 0
##           DepTime        CRSDepTime           ArrTime        CRSArrTime
##                 0                 0                 0                 0
##     UniqueCarrier         FlightNum           TailNum ActualElapsedTime
##                 0                 0                 0                 0
##    CRSElapsedTime           AirTime          ArrDelay          DepDelay
##                 0                 0                 0                 0
##            Origin              Dest          Distance            TaxiIn
##                 0                 0                 0                 0
##           TaxiOut         Cancelled  CancellationCode          Diverted
##                 0                 0                 0                 0
##      CarrierDelay      WeatherDelay          NASDelay     SecurityDelay
##                 0                 0                 0                 0
## LateAircraftDelay
##                 0
```

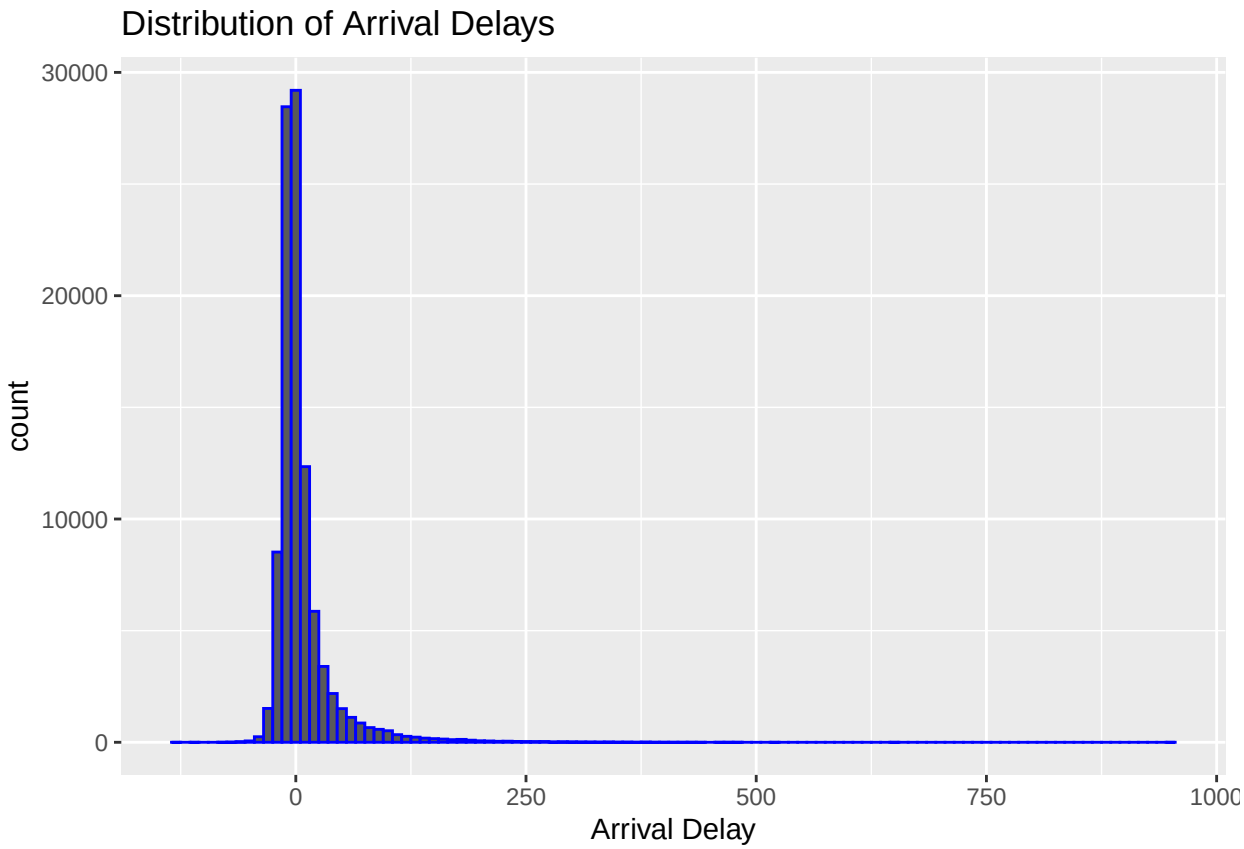
```
col_factors <- c('Month', 'DayofMonth', 'DayOfWeek', 'Cancelled', 'Diverted')
airport[,col_factors] <- lapply(airport[,col_factors], as.factor)
airport$Dep_Hr <- sapply(DepTime, function(x) x%/%100)
airport$CRSDep_Hr <- sapply(CRSDepTime, function(x) x%/%100)
airport$Arr_Hr <- sapply(ArrTime, function(x) x%/%100)
airport$CRSArr_Hr <- sapply(CRSArrTime, function(x) x%/%100)
aus.dep <- subset(airport, Origin == 'AUS')
aus.arr <- subset(airport, Dest == 'AUS')
```

```
pl <- ggplot(data = airport, aes(x=ArrDelay)) +
  geom_histogram(bins = 100, binwidth = 10, color='blue') +
  xlab('Arrival Delay') +
  ggtitle('Distribution of Arrival Delays')
print(pl)
```




```
pl2 <- ggplot(data=airport, aes(x=))
```

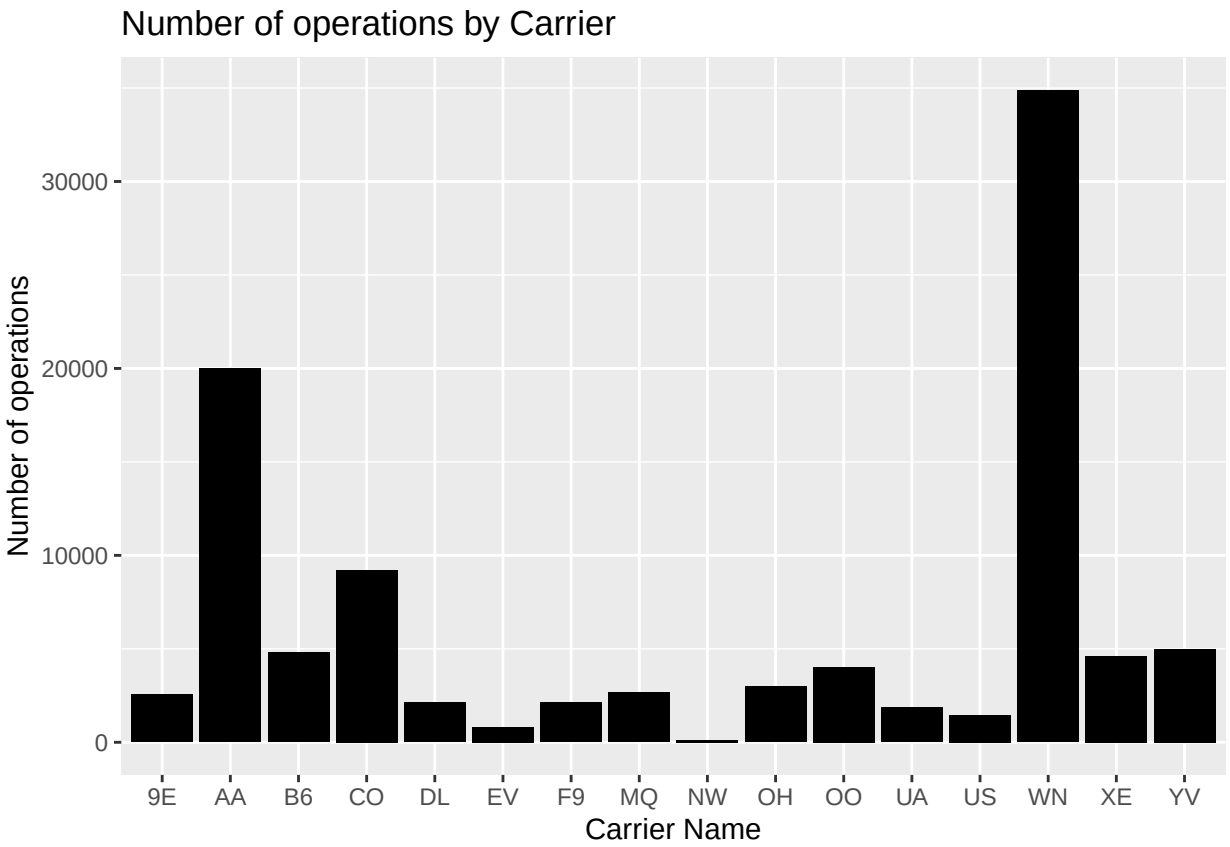
```
pl <- ggplot(data = airport, aes(x=ArrDelay)) +
  geom_histogram(bins = 100, binwidth = 10, color='blue') +
  xlab('Arrival Delay') +
  ggtitle('Distribution of Arrival Delays')
print(pl)
```




```
pl2 <- ggplot(data=airport, aes(x=))
```

##arrival delays and departure delays looks about the same centered around zero.

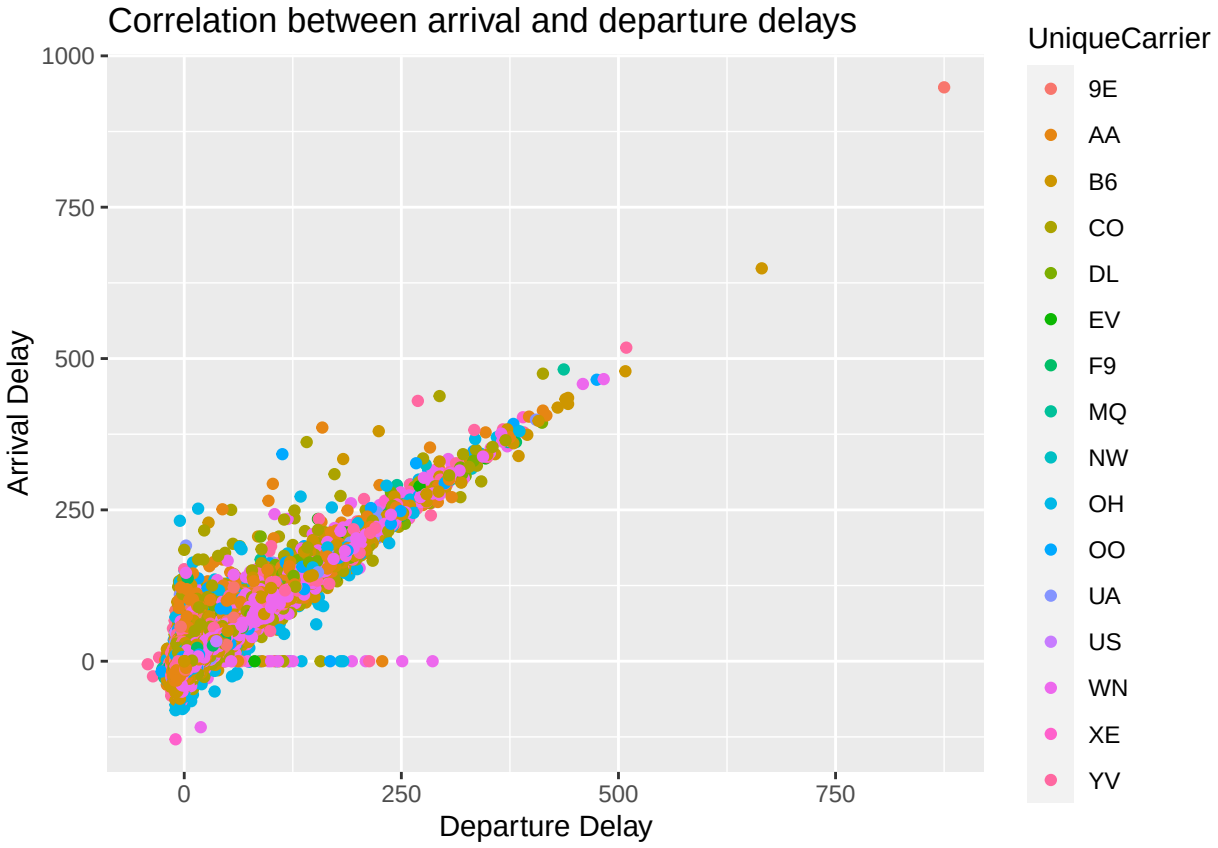
```
pl <- ggplot(aes(x=UniqueCarrier), data=airport) +
  geom_bar(fill='black', position='dodge') +
  ggtitle('Number of operations by Carrier') +
  xlab('Carrier Name') +
  ylab('Number of operations')

print(pl)
```



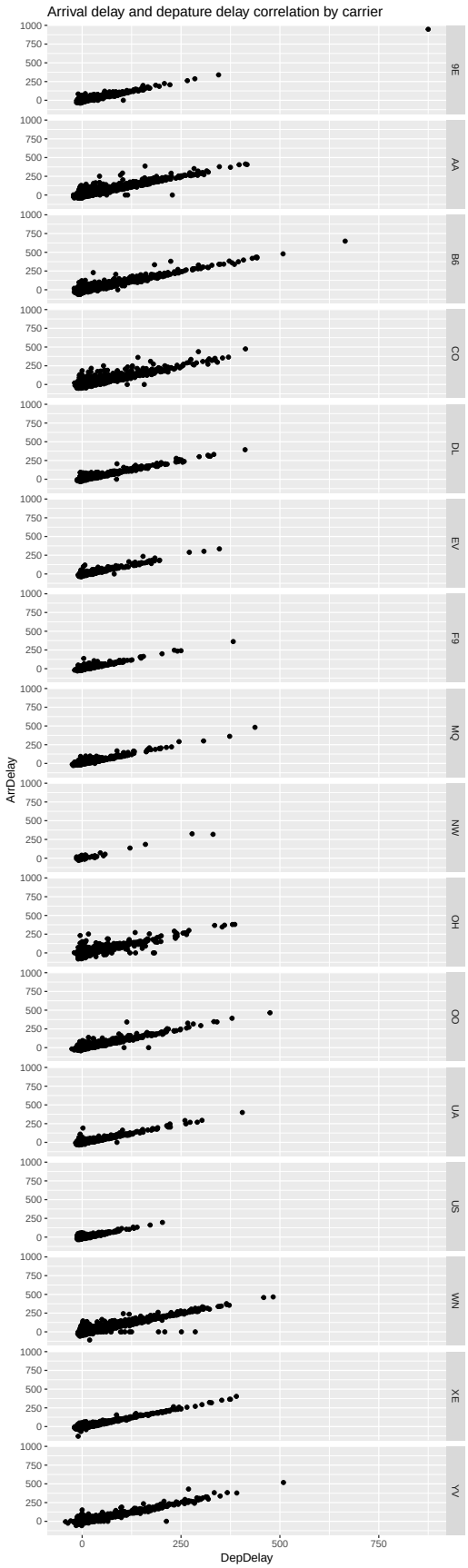
##Southwest(WN) tops the list with almost 40k operations, followed by Alaskan Airlines(AA). Northwest Airlines(NW) has the fewest operations

Correlation of arrival and departure delays.There are a few outliers, but it is a good straight line curve showing

they are directly correlated



We will try to visualize the correlation for carriers individually. Visually, the slope of the plots remain roughly constant, so NO carriers are consistently making up for the departure delays.



the slope of the plots remain roughly constant for each

individual airlines. if the plane arrived late, they will likely to depart late

Portfolio Modeling

```
## Portfolio Modeling
library(mosaic)
library(quantmod)
```

```
## Loading required package: xts

## Loading required package: zoo

##
## Attaching package: 'zoo'

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

##
## Attaching package: 'xts'

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

## Loading required package: TTR

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

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

```
library(foreach)
```

```
##
## Attaching package: 'foreach'

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

random Portfolio 1

IXC: iShares S&P Global Energy Sector

GOEX: GLB X FUNDS/GLB X GOLD EXPLORER

Vanguard Energy ETF

```
mystocks = c("IXC", "GOEX", "VDE")
myprices = getSymbols(mystocks, from = "2015-08-04")
```

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

```
# A chunk of code for adjusting all stocks
# creates a new object adding 'a' to the end
# For example, WMT becomes WMTa, etc
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/IXC?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/IXC?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/GOEX?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/GOEX?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VDE?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/VDE?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

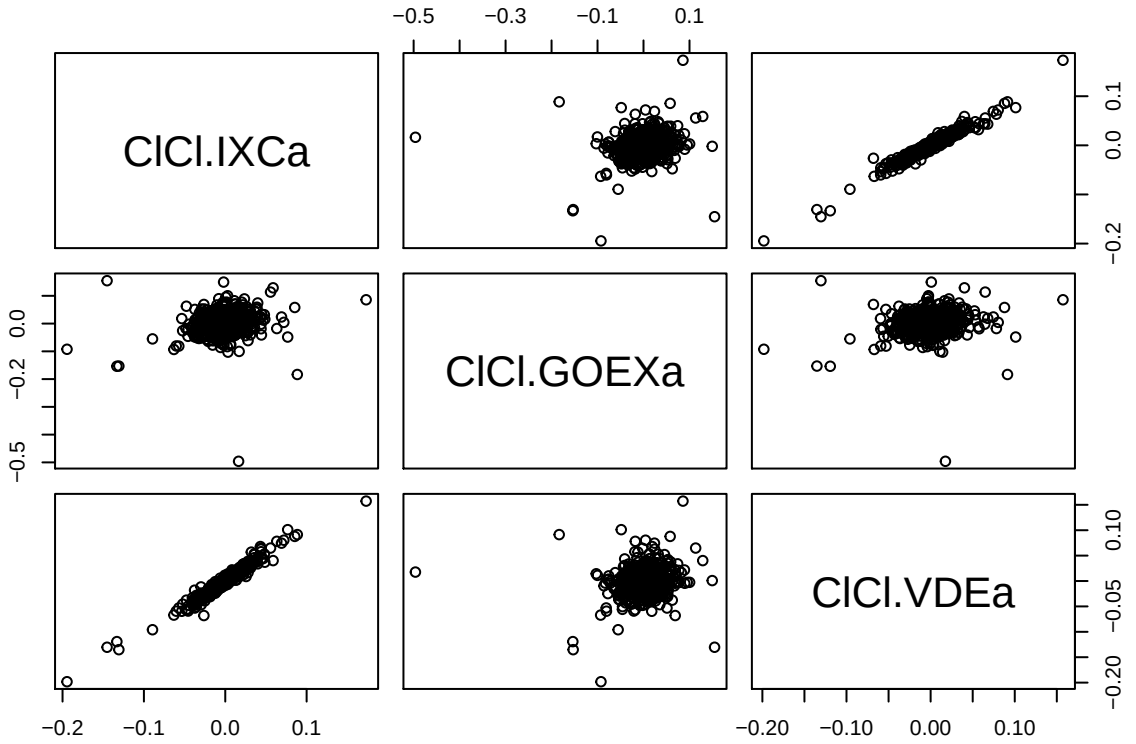
```
# Combine all the returns in a matrix
all_returns = cbind(ClCl(IXCa),
                                ClCl(GOEXa),
                                ClCl(VDEa))
head(all_returns)
```

```
##               ClCl.IXCa  ClCl.GOEXa     ClCl.VDEa
## 2015-08-04           NA          NA            NA
## 2015-08-05 -0.001869066 -0.02517986 -0.0083909695
## 2015-08-06  0.010299563  0.02583026  0.0180875712
## 2015-08-07 -0.012357121  0.01438849 -0.0196363532
## 2015-08-10  0.024085174  0.03073292  0.0335947220
## 2015-08-11 -0.003054459  0.04931181  0.0006151646
```

```
#remove the first day (no return calculated)
all_returns = as.matrix(na.omit(all_returns))
head(all_returns)
```

```
##               ClCl.IXCa  ClCl.GOEXa     ClCl.VDEa
## 2015-08-05 -0.001869066 -0.02517986 -0.0083909695
## 2015-08-06  0.010299563  0.02583026  0.0180875712
## 2015-08-07 -0.012357121  0.01438849 -0.0196363532
## 2015-08-10  0.024085174  0.03073292  0.0335947220
## 2015-08-11 -0.003054459  0.04931181  0.0006151646
## 2015-08-12  0.014705883  0.06120224  0.0176247779
```

```
pairs(all_returns)
```



##We see some trends here. VDE and IXC seems to be positively correlated. the return level of IXC and GOEX seems cluster together

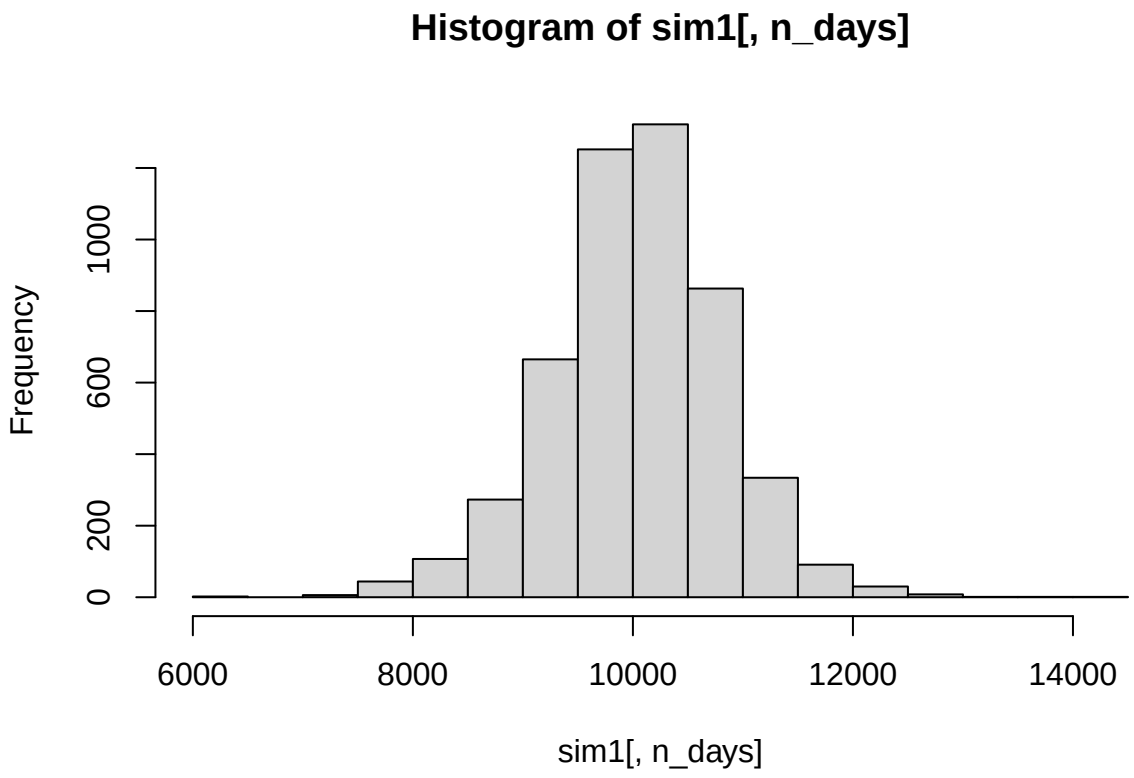
```
initial_wealth = 10000
set.seed(1000)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.5, 0.3, 0.2)
    holdings = weights * total_wealth
    n_days = 20 #capital T in the notes
    wealthtracker = rep(0, n_days) #setup a place holder to tracl total wealth
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        holdings = weights * total_wealth ##rebalance portfolios
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
```

```
head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  9995.587  9947.504  9966.838  9912.406 10022.770 10026.043 10455.185
## result.2  9871.737  9894.297  9995.920 10086.817 10167.841 10222.299 10291.575
```

```
## result.3 10065.732 10071.576 10150.739  9962.258  9817.376  9808.701  9769.725
## result.4  9866.565  9892.929  9936.865  9979.546  9919.095  9844.022  9866.982
## result.5 10144.160 10227.404 10332.715 10327.885 10411.855 10087.061 10051.937
## result.6  9969.118  9947.628  9959.095  9910.674  9953.953 10028.577 10327.239
##               [,8]      [,9]     [,10]     [,11]    [,12]     [,13]     [,14]
## result.1 10599.184 10634.489 10623.222 10730.082 10570.56 10523.771 10512.836
## result.2 10269.404 10311.489 10399.190 10409.049 10495.74 10550.827 11067.606
## result.3  9827.789  9796.099  9870.089  9844.104  9969.70  9940.553  9830.132
## result.4 10045.550  9916.248 10066.313 10049.907 10098.09 10237.172 10337.453
## result.5 10146.768 10157.764 10175.242 10039.469 10024.22 10025.285  9781.577
## result.6 10912.075 10802.382 10745.608 10764.384 10833.81 10715.254 10754.098
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 10694.471 10625.302 10581.835 10424.082 10681.233 10735.926
## result.2 10814.420 10529.470 10288.846 10620.751 10603.178 10653.150
## result.3  9579.985  9691.047  9750.246  9742.613  9755.429  9693.888
## result.4 10291.004 10243.340 10122.091 10114.968 10036.092  9999.964
## result.5  9817.502  9703.690  9903.232  9848.751  9932.613  9753.251
## result.6 10800.188 10707.679 10502.331 10492.515 10436.457 10524.875
```

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



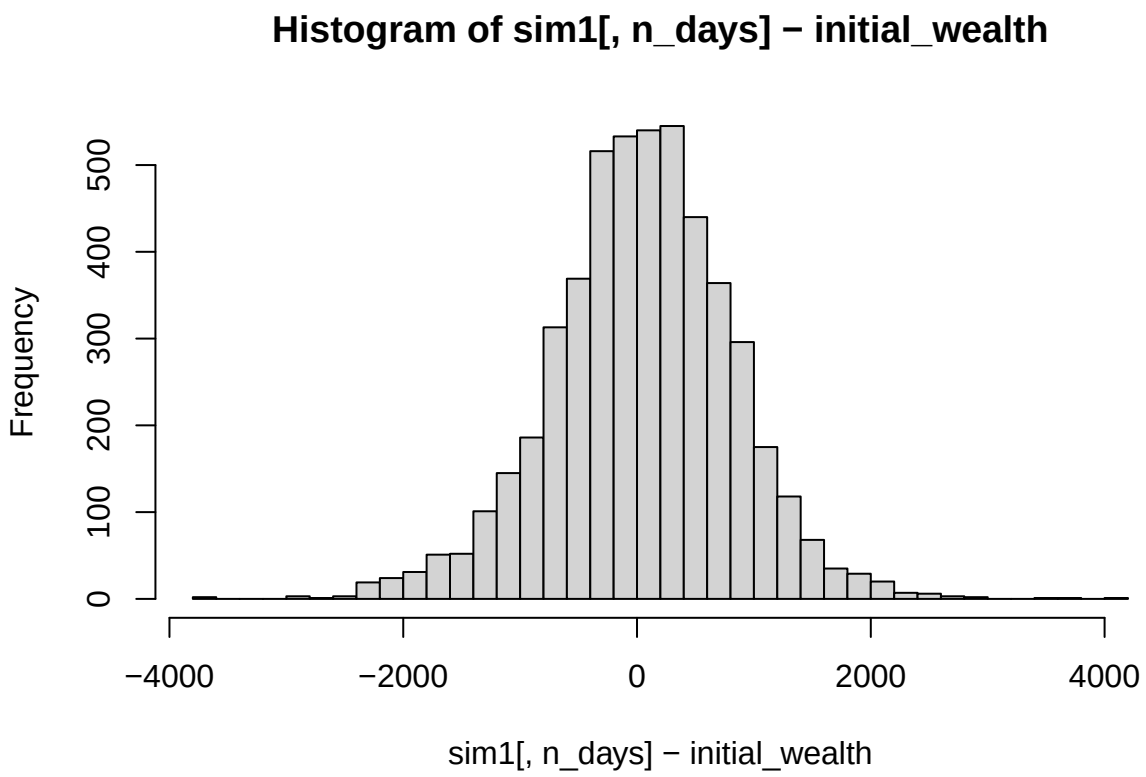

```
## average total wealth after 20 days
mean(sim1[,n_days]) ## average total wealth after 20 days
```

```
## [1] 10040.49
```

```r
## average gains after 20days
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 40.4925
```

```r
## histogram of gains distribution
hist(sim1[,n_days]- initial_wealth, breaks=30) #histogram of gains distribution
```




```r
# 5% value at risk:
quantile(sim1[,n_days]- initial_wealth, prob=0.05)
```

```
##        5%
## -1264.004
```

we are 95% confident that our worst daily loss will not exceed 1254.1 at c(0.5, 0.3, 0.2) for "IXC", "GOEX", "VDE".

random Portfolio 2

AIA: iShares S&P Global Energy Sector

RING: GLB X FUNDS/GLB X GOLD EXPLORER

CSD:Invesco S&P Spin-Off ETF

```
mystocks = c("AIA", "RING", "CSD")
myprices = getSymbols(mystocks, from = "2015-08-04")
# A chunk of code for adjusting all stocks
# creates a new object adding 'a' to the end
# For example, WMT becomes WMTa, etc
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/AIA?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/AIA?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/RING?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/RING?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/CSD?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/CSD?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

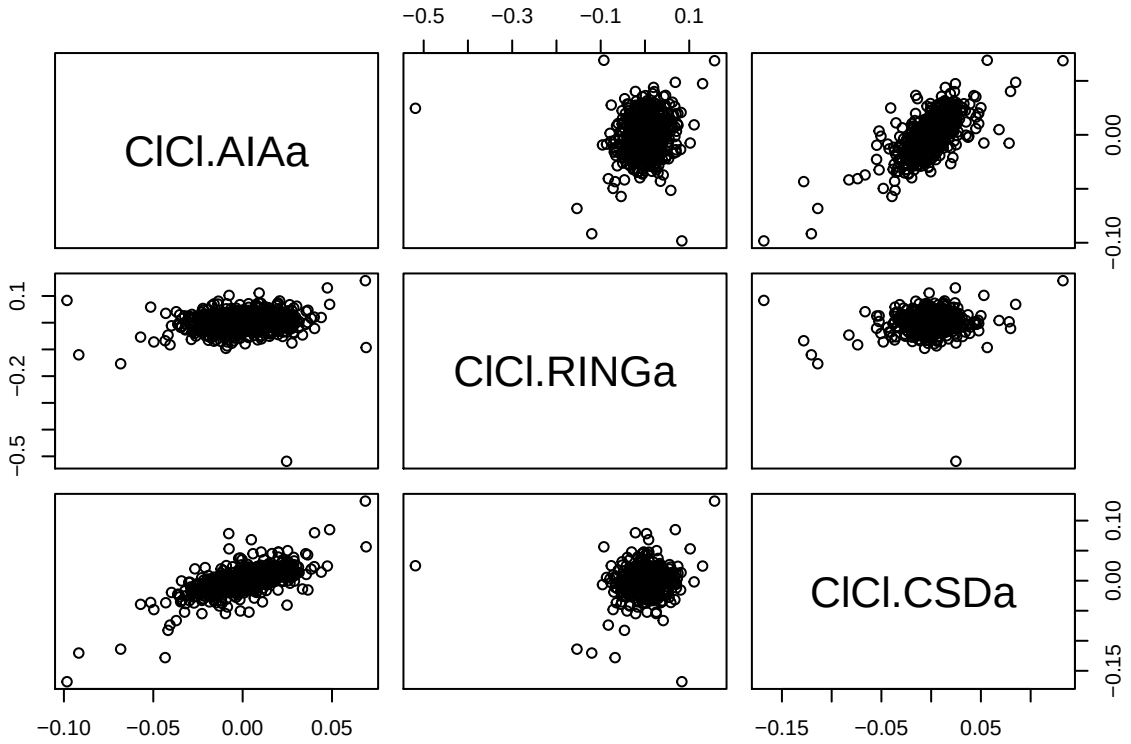
```
all_returns = cbind(ClCl(AIAa),
                                ClCl(RINGa),
                                ClCl(CSDa))
head(all_returns)
```

```
##               ClCl.AIAa   ClCl.RINGa    ClCl.CSDa
## 2015-08-04           NA           NA           NA
## 2015-08-05  0.001073883 -0.013232514 -0.003571429
## 2015-08-06 -0.012229135  0.022988506 -0.014784924
## 2015-08-07  0.004344070  0.001872659 -0.006139177
## 2015-08-10  0.014489575  0.056074766  0.010752711
## 2015-08-11 -0.022170134  0.021238938 -0.009732911
```

```
#remove the first day (no return calculated)
all_returns = as.matrix(na.omit(all_returns))
head(all_returns)
```

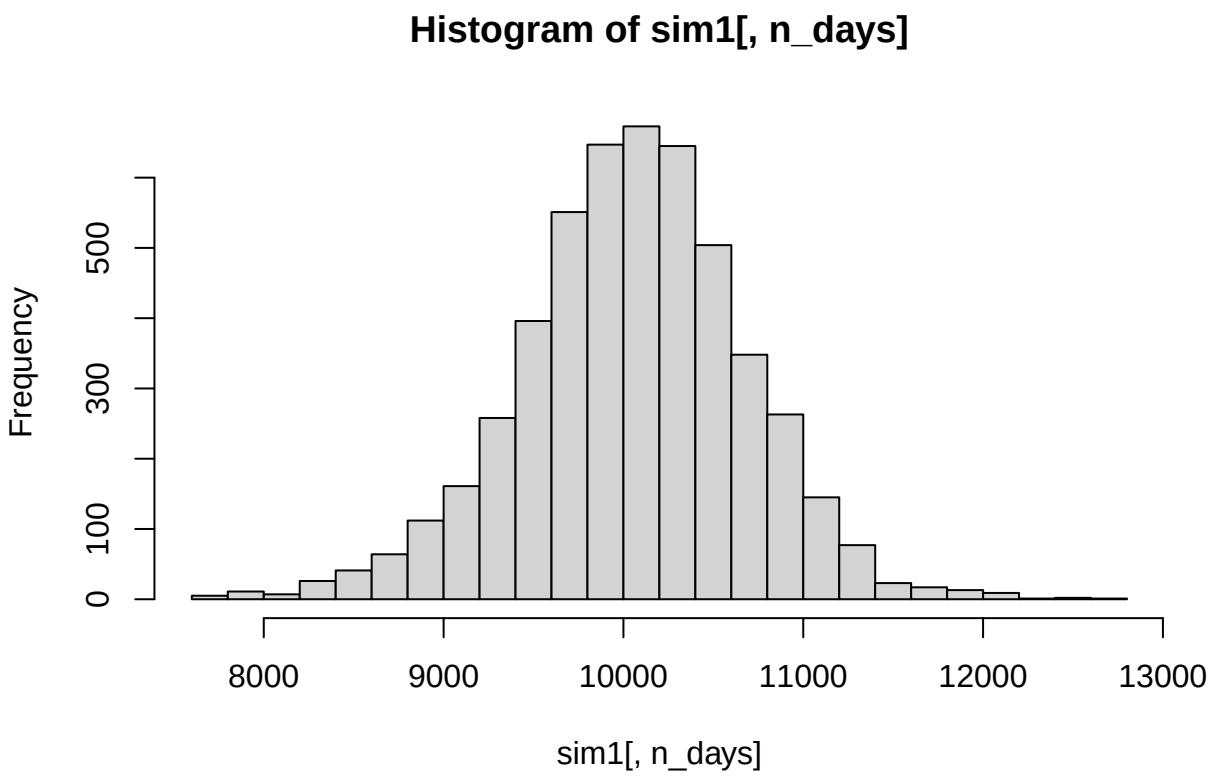
```
##               ClCl.AIAa   ClCl.RINGa    ClCl.CSDa
## 2015-08-05  0.001073883 -0.013232514 -0.003571429
## 2015-08-06 -0.012229135  0.022988506 -0.014784924
## 2015-08-07  0.004344070  0.001872659 -0.006139177
## 2015-08-10  0.014489575  0.056074766  0.010752711
## 2015-08-11 -0.022170134  0.021238938 -0.009732911
## 2015-08-12 -0.017222521  0.064124783 -0.001142834
```

```
## showing pair wise stock return correlations
pairs(all_returns)
```




```
initial_wealth = 10000
set.seed(1000)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.3, 0.05,0.65)
    holdings = weights * total_wealth
    n_days = 20 #capital T in the notes
    wealthtracker = rep(0, n_days) #setup a place holder to tracl total wealth
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = weights * total_wealth
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        holdings = weights * total_wealth
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
```

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



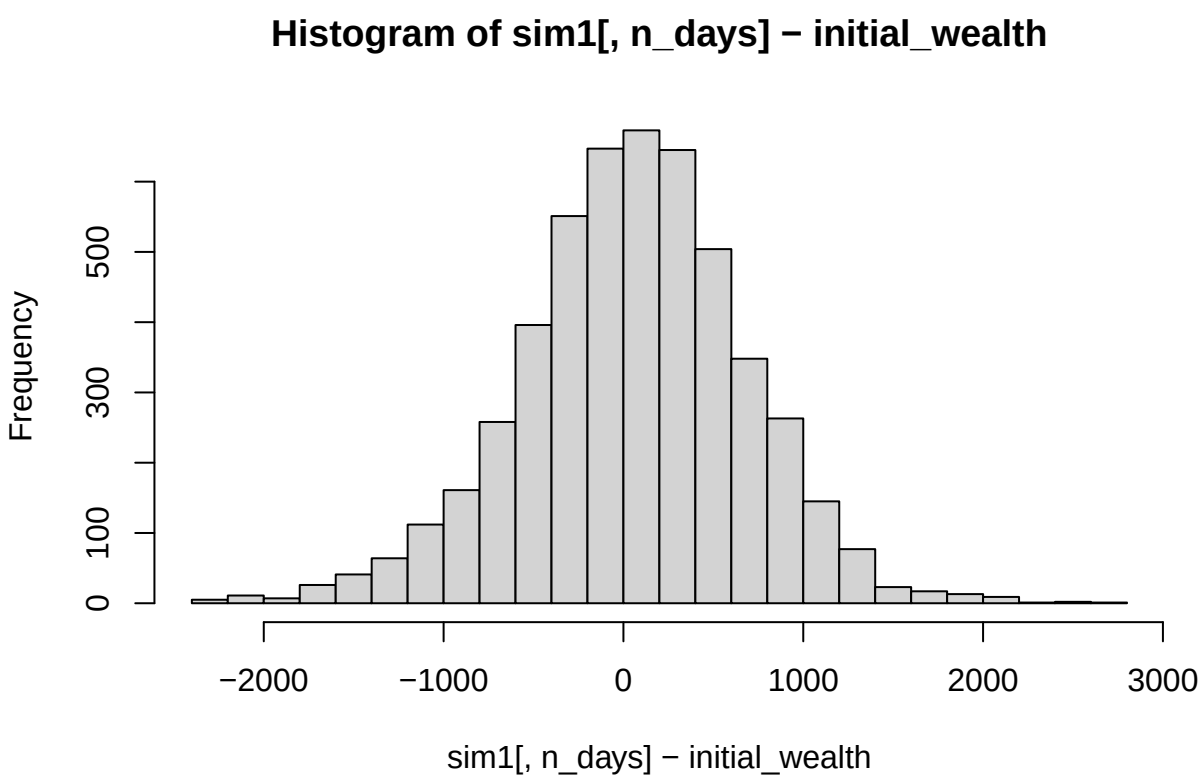

```
## average total wealth after 20 days
mean(sim1[,n_days]) ## average total wealth after 20 days
```

```
## [1] 10051.26
```

```
## average gains after 20days
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 51.26001
```

```
## histogram of gains distribution
hist(sim1[,n_days]- initial_wealth, breaks=30) #histogram of gains distribution
```




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

```
##       5%
## -1026.55
```

we are 95% confident that our worst daily loss will not exceed 1045.881 at c(0.5, 0.3, 0.2) for "IXC", "GOEX", "VDE".

let's try a 4 ETFS porfolios

random Portfolio 3

SLYV: SPDR S&P 600 Small Cap Value ETF

EES: WisdomTree U.S. SmallCap Earnings Fund

VIOV:Vanguard S&P Small-Cap 600 Value Index Fund ETF Shares

FXZ: First Trust Materials AlphaDEX Fund

```r
mystocks = c("SLYV", "EES", "VIOV","FXZ")
myprices = getSymbols(mystocks, from = "2015-08-04")
# A chunk of code for adjusting all stocks
# creates a new object adding 'a' to the end
# For example, WMT becomes WMTa, etc
for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SLYV?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/SLYV?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/EES?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/EES?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/VIOV?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VIOV?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'

## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/FXZ?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query2.finance.yahoo.com/v7/finance/download/FXZ?
## period1=-2208988800&period2=1597449600&interval=1d&events=split&crumb=q/
## NBcy3hYpQ'
```

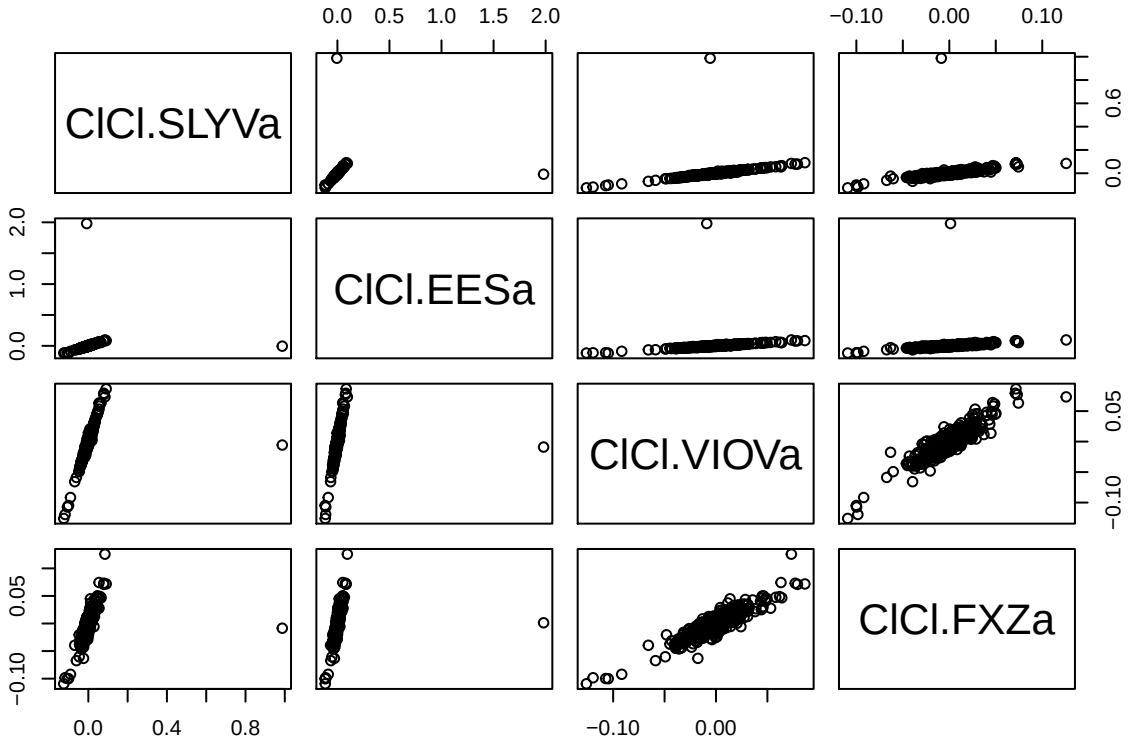
```
all_returns = cbind(ClCl(SLYVa),
                                ClCl(EESa),
                                ClCl(VIOVa),
                                ClCl(FXZa))
head(all_returns)
```

```
##               ClCl.SLYVa    ClCl.EESa   ClCl.VIOVa     ClCl.FXZa
## 2015-08-04            NA           NA           NA            NA
## 2015-08-05  0.0002924637  0.001365888  0.004577398  0.0058517880
## 2015-08-06 -0.0038011502 -0.002728125 -0.005062778 -0.0009696509
## 2015-08-07 -0.0075334899 -0.010072121 -0.008955882 -0.0077644775
## 2015-08-10  0.0125196961  0.016832069  0.016943952  0.0293446365
## 2015-08-11 -0.0062311362 -0.006300185 -0.012319509 -0.0171048147
```

```
#remove the first day (no return calculated)
all_returns = as.matrix(na.omit(all_returns))
head(all_returns)
```

```
##               ClCl.SLYVa     ClCl.EESa   ClCl.VIOVa     ClCl.FXZa
## 2015-08-05  0.0002924637  0.0013658885  0.004577398  0.0058517880
## 2015-08-06 -0.0038011502 -0.0027281251 -0.005062778 -0.0009696509
## 2015-08-07 -0.0075334899 -0.0100721212 -0.008955882 -0.0077644775
## 2015-08-10  0.0125196961  0.0168320686  0.016943952  0.0293446365
## 2015-08-11 -0.0062311362 -0.0063001851 -0.012319509 -0.0171048147
## 2015-08-12 -0.0022533947 -0.0002486449  0.000204519  0.0019335803
```

```
## showing pair wise stock return correlations
pairs(all_returns)
```



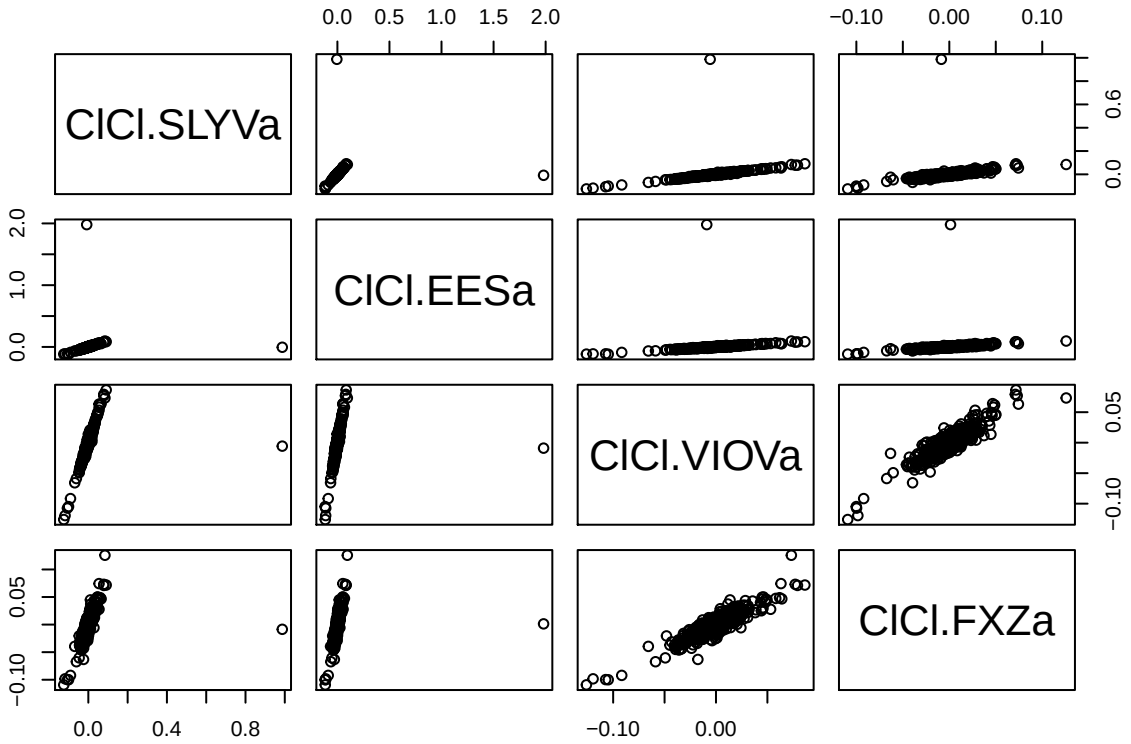

```
all_returns = cbind(ClCl(SLYVa),
                                ClCl(EESa),
                                ClCl(VIOVa),
                                ClCl(FXZa))
head(all_returns)
```

```
##               ClCl.SLYVa    ClCl.EESa   ClCl.VIOVa     ClCl.FXZa
## 2015-08-04            NA           NA           NA            NA
## 2015-08-05  0.0002924637  0.001365888  0.004577398  0.0058517880
## 2015-08-06 -0.0038011502 -0.002728125 -0.005062778 -0.0009696509
## 2015-08-07 -0.0075334899 -0.010072121 -0.008955882 -0.0077644775
## 2015-08-10  0.0125196961  0.016832069  0.016943952  0.0293446365
## 2015-08-11 -0.0062311362 -0.006300185 -0.012319509 -0.0171048147
```

```
#remove the first day (no return calculated)
all_returns = as.matrix(na.omit(all_returns))
head(all_returns)
```

```
##               ClCl.SLYVa     ClCl.EESa   ClCl.VIOVa     ClCl.FXZa
## 2015-08-05  0.0002924637  0.0013658885  0.004577398  0.0058517880
## 2015-08-06 -0.0038011502 -0.0027281251 -0.005062778 -0.0009696509
## 2015-08-07 -0.0075334899 -0.0100721212 -0.008955882 -0.0077644775
## 2015-08-10  0.0125196961  0.0168320686  0.016943952  0.0293446365
## 2015-08-11 -0.0062311362 -0.0063001851 -0.012319509 -0.0171048147
## 2015-08-12 -0.0022533947 -0.0002486449  0.000204519  0.0019335803
```

```
## showing pair wise stock return correlations
pairs(all_returns)
```




```
initial_wealth = 10000
set.seed(1000)
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.25,0.25,0.25,0.25) #equal 25% weight
    holdings = weights * total_wealth
    n_days = 20 #capital T in the notes
    wealthtracker = rep(0, n_days) #setup a place holder to tracl total wealth
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        holdings = weights * total_wealth
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}
```

```
## average total wealth after 20 days
mean(sim1[,n_days]) ## average total wealth after 20 days
```

```
## [1] 10160.16
```

```r
## average gains after 20days
mean(sim1[,n_days] - initial_wealth)
```

```
## [1] 160.1625
```

```r
## histogram of gains distribution
hist(sim1[,n_days]- initial_wealth, breaks=30) #histogram of gains distribution
```


```r
# 5% value at risk:
quantile(sim1[,n_days]- initial_wealth, prob=0.05)
```

```
##        5%
## -1092.314
```

we are 95% confident that our worst daily loss will not exceed 1103.32 at c(0.5, 0.3, 0.2) for "IXC", "GOEX", "VDE".

```r
library(ggplot2)
library(ggthemes)
```

```
##
## Attaching package: 'ggthemes'
```

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

```
library(reshape2)
```

```
##
## Attaching package: 'reshape2'
```

```
## The following object is masked from 'package:tidyr':
##
##     smiths
```

```
library(RCurl)
```

```
##
## Attaching package: 'RCurl'
```

```
## The following object is masked from 'package:tidyr':
##
##     complete
```

```
library(foreach)
library(fpc)
library(cluster)
urlfile="https://raw.githubusercontent.com/jgscott/STA380/master/data/social_marketing.csv"
social_m_raw <- read.csv(url(urlfile))
social_m <- read.csv(url(urlfile))
```

Market segmentation

Steps taken

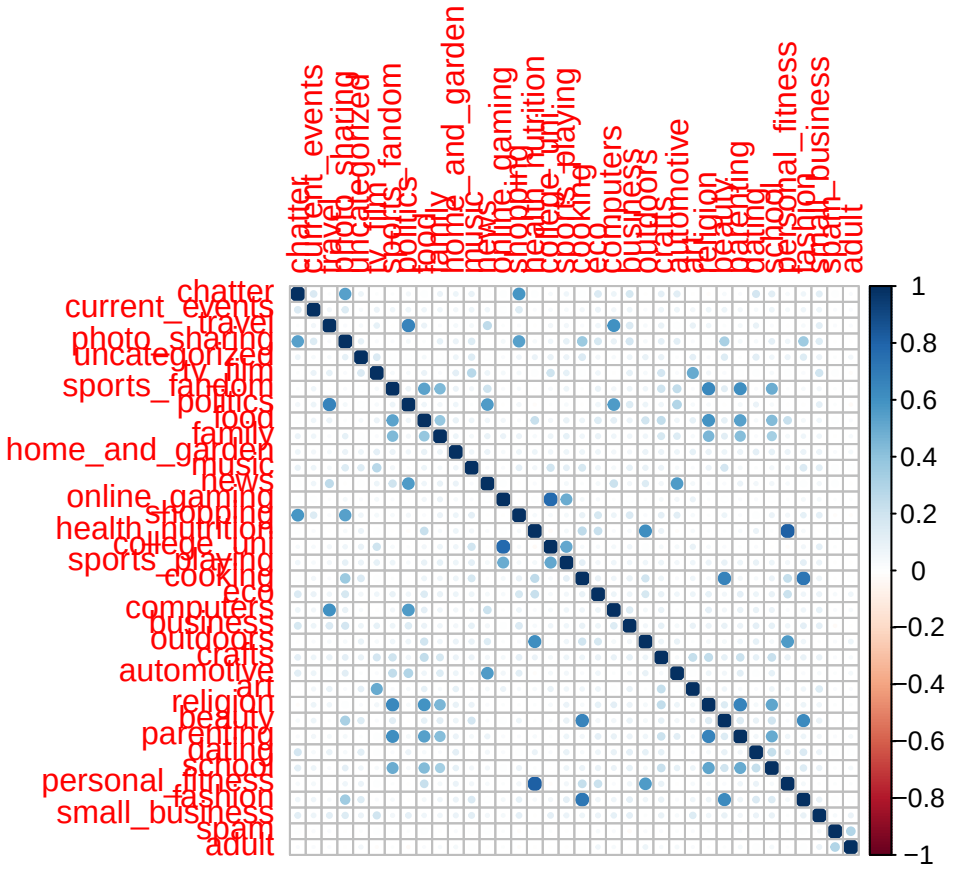
K-means with the raw data

K-means with k-means++ initialization

K-means with k-means++ initialization using PCA data

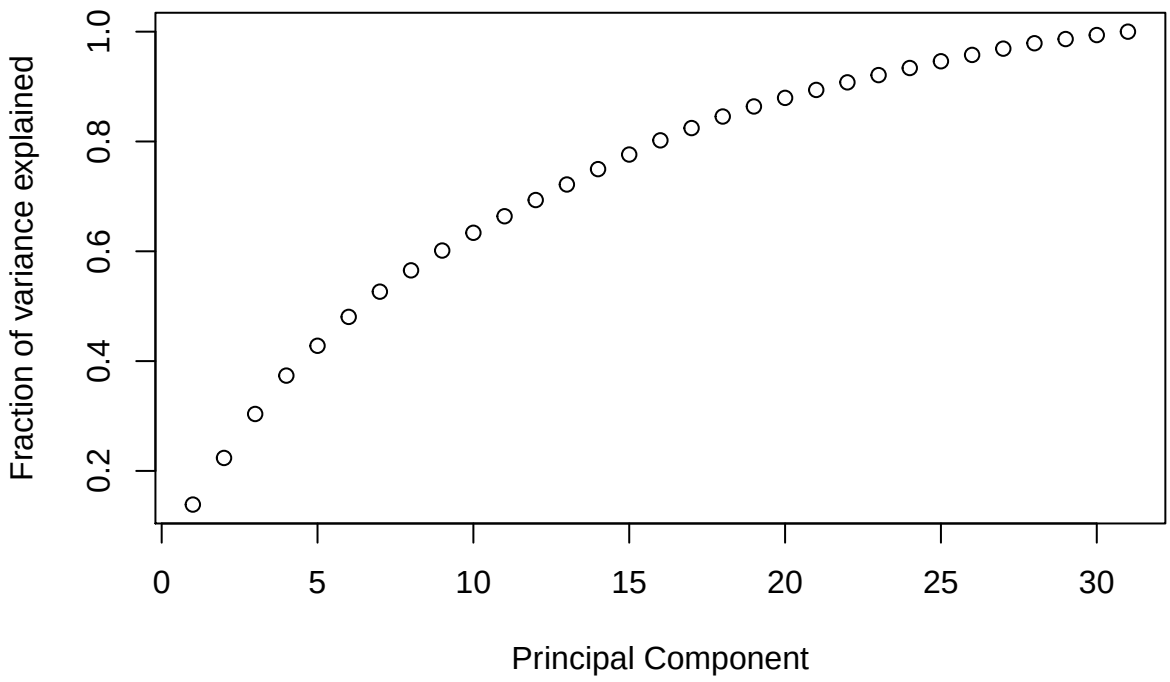
Hierarchial clustering using PCA data

Correlation plot



#personal fitness and health nutrition are highly correlated. online gaming and college university variables are correlated.

PCA

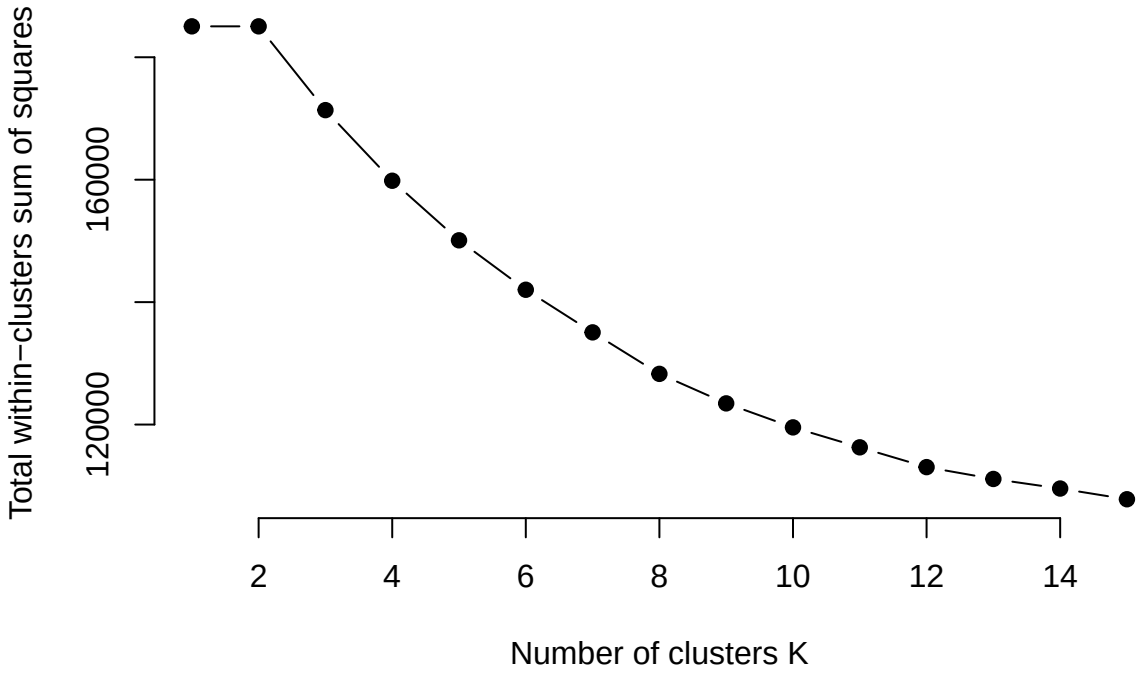



```
cumsum(pca_var1)[10]
```

```
## [1] 0.6337156
```

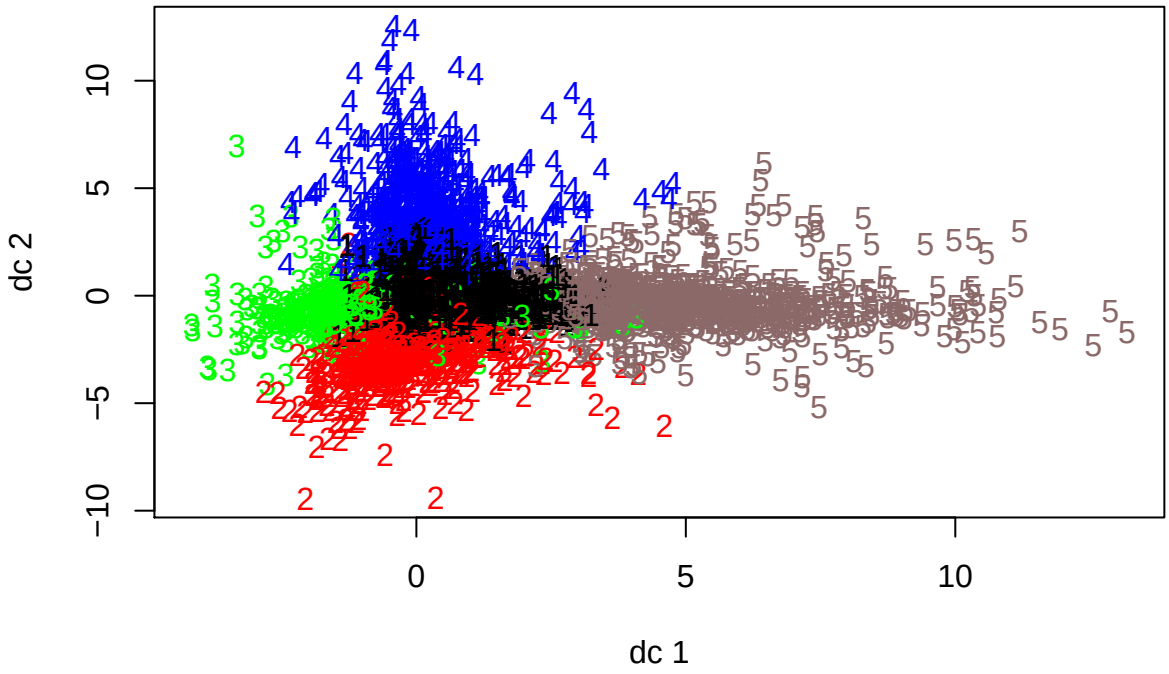
At 10th PC, around 63.37% of the variation is explained. According to Kaiser criterion, we should drop all the principal components with eigen values less than 1.0. Hence, let's pick 10 principal components.

Kmean



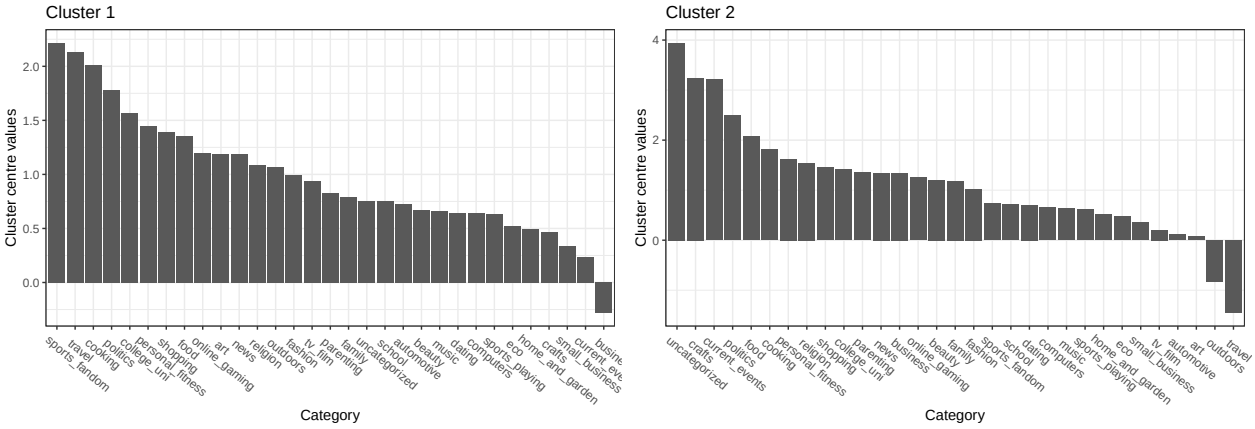
##o use 5 cluster since it is easier to intrepret and identify market segments.

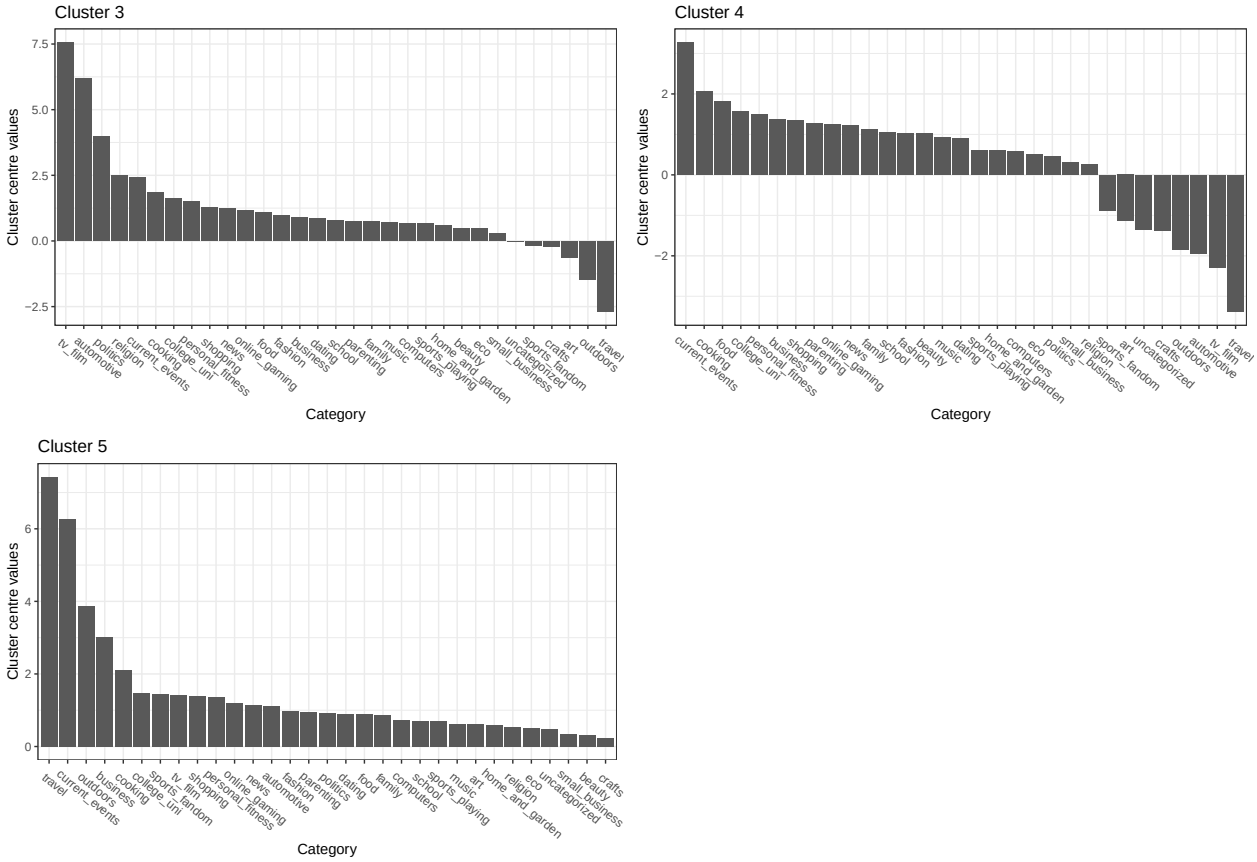
Cluster visualization



#The clusters look well separated. Let's identify the characteristics of the clusters.

##Results



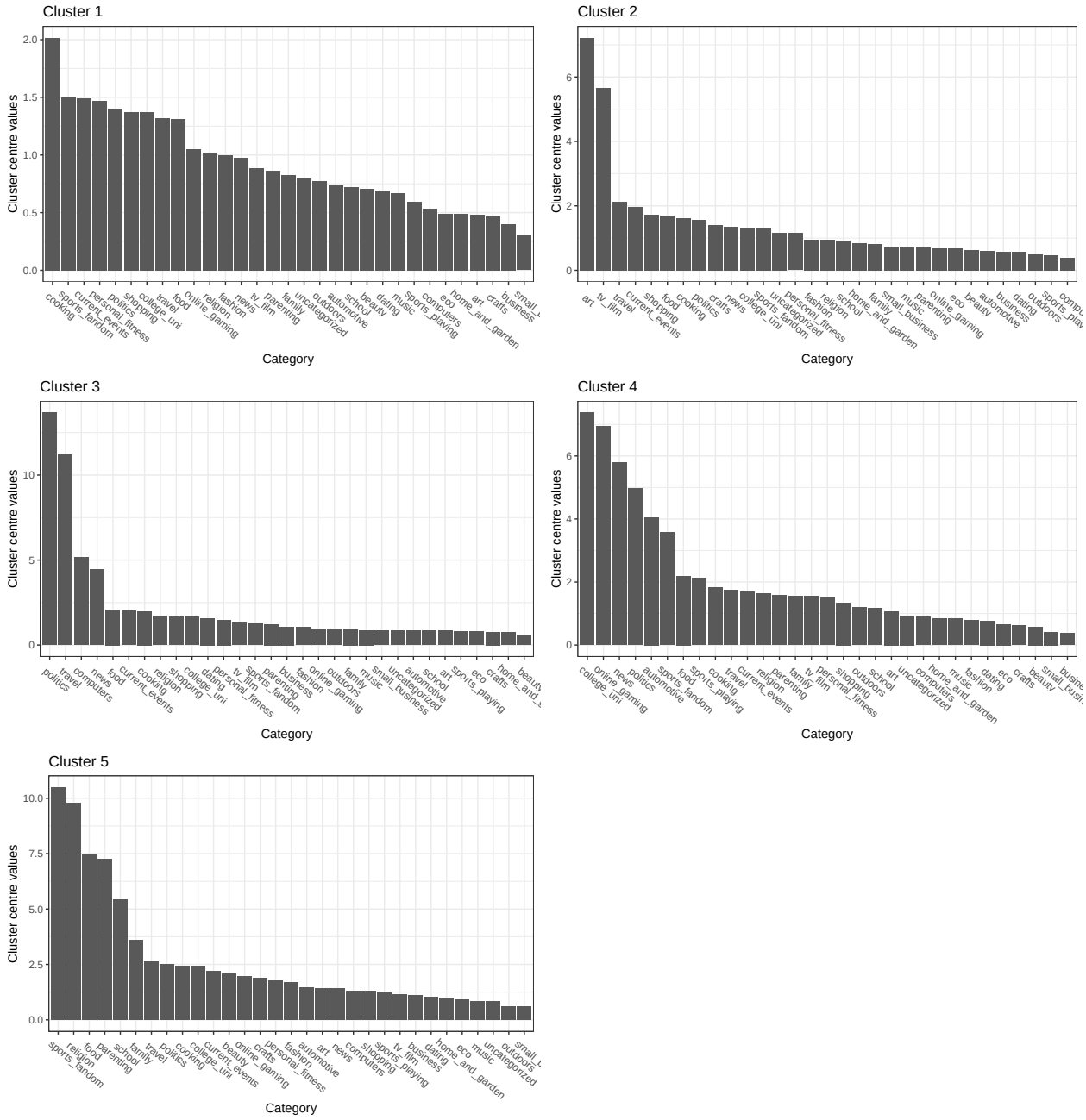


##Market segments identified ##1. Sports Fandom, Travel, Cooking ##2. Crafts, Current Events ##3. TV Film, Automotive, Politics ##4. Cooking, Personal Fitness, Food, Shopping, Fashion ##5. Travel, Outdoors, Business

#Cluster 4 - "Cooking, Personal Fitness, Food, Shopping, Fashion" and Cluster 5 - "Travel, Outdoors, Business" differ vastly in terms of interests. Cluster 5 consists mainly of people who love travelling and people in Cluster 4 are more focused on personal grooming. #Furthermore,Cluster in 1 is primarily composed of people who have a penchant for sports, travel and cooking. In contrast, cluster 2 has people who are artistic #Cluster 3 seems like people with eclectic interests - starting from movies to automotive to politics and religion as well!

Hierarchial Clustering

Results



##Market segments identified ##1. Cooking, Personal Fitness ##2. Art, TV Film, Shopping ##3. Politics, Travel, Computers ##4. College Universities, Online Gaming, News ##5. Religion, Food, Parenting, School, Family

Clustering reveals some interesting segments that differ by demographics. Intuitively, Cluster 4 - "College Universities, Online Gaming, News" will consist of a younger population as compared to Cluster 5 - "Religion, Food, Parenting, School, Family". NutrientH20 can design demographic specific marketing campaigns to get the most effective message across to their audience.

Cluster 2 has a group of people who have artsy interests - art, tv film and travel. They sure do know how to enjoy life

Cluster 3 has the computer lovers, who also have an interest for politics and travel.

Market segementation can allows us to derive insights that will help send the right message to the right group of people that will maximize the profits of the company and help build better relationships with the audience.

Author Attribution

reading the files, processing data & Tokenization

reading all the books in training set

```
train=Sys.glob('C:/Users/Lai Jiang/Desktop/UT MSBA Summer Courses/Predictive Modeling/STA380-James Scot
```

Creating training dataset

```
comb_art=NULL
labels=NULL
for (name in train)
{
  author=substring(name,first=50)
  article=Sys.glob(paste0(name,'/*.txt'))
  comb_art=append(comb_art,article)
  labels=append(labels,rep(author,length(article)))
}
```

cleaning the file name

```
readerPlain <- function(fname)
  {
                readPlain(elem=list(content=readLines(fname)),
                          id=fname, language='en')
  }
comb = lapply(comb_art, readerPlain)
names(comb) = comb_art
names(comb) = sub('.txt', '', names(comb))
```

Create a text mining corpus

```
corp_tr=Corpus(VectorSource(comb))
```

processing and tokenization:

change alphabet to lower cases

Removing numbers

Removing punctuation

Removing excess space

stop words*

```
## <<DocumentTermMatrix (documents: 2500, terms: 32575)>>
## Non-/sparse entries: 552741/80884759
## Sparsity           : 99%
## Maximal term length: 58
## Weighting          : term frequency (tf)

## <<DocumentTermMatrix (documents: 2500, terms: 3398)>>
## Non-/sparse entries: 382851/8112149
## Sparsity           : 95%
## Maximal term length: 58
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

Repeat for the test directories**

Read Test files

```
test=Sys.glob('C:/Users/Lai Jiang/Desktop/UT MSBA Summer Courses/Predictive Modeling/STA380-James Scott/
```

```
comb_art1=NULL
labels1=NULL
for (name in test)
{
  author1=substring(name,first=50)
  article1=Sys.glob(paste0(name,'/*.txt'))
  comb_art1=append(comb_art1,article1)
  labels1=append(labels1,rep(author1,length(article1)))
}
```

cleaning the file name

```
comb1 = lapply(comb_art1, readerPlain)
names(comb1) = comb_art1
names(comb1) = sub('.txt', '', names(comb1))
```

#Create a text mining corpus

```
corp_ts=Corpus(VectorSource(comb1))
```

processing and tokenization*

making sure we have the same test and training sets

```
## <<DocumentTermMatrix (documents: 2500, terms: 3398)>>
## Non-/sparse entries: 379232/8115768
## Sparsity           : 96%
## Maximal term length: 58
## Weighting          : term frequency - inverse document frequency (normalized) (tf-idf)
```

Reducing dimentionality

Using PCA to extract relevant features from the huge set of variables and eliminate the effect of multicollinearity

Data Preparation for PCA

Eliminate NA columns

```
DTM_trr_1<-DTM_trr[,which(colSums(DTM_trr) != 0)]
DTM_tss_1<-DTM_tss[,which(colSums(DTM_tss) != 0)]
```

only using intersecting columns

```
#8312500 elements in both.
DTM_tss_1 = DTM_tss_1[,intersect(colnames(DTM_tss_1),colnames(DTM_trr_1))]
DTM_trr_1 = DTM_trr_1[,intersect(colnames(DTM_tss_1),colnames(DTM_trr_1))]
```

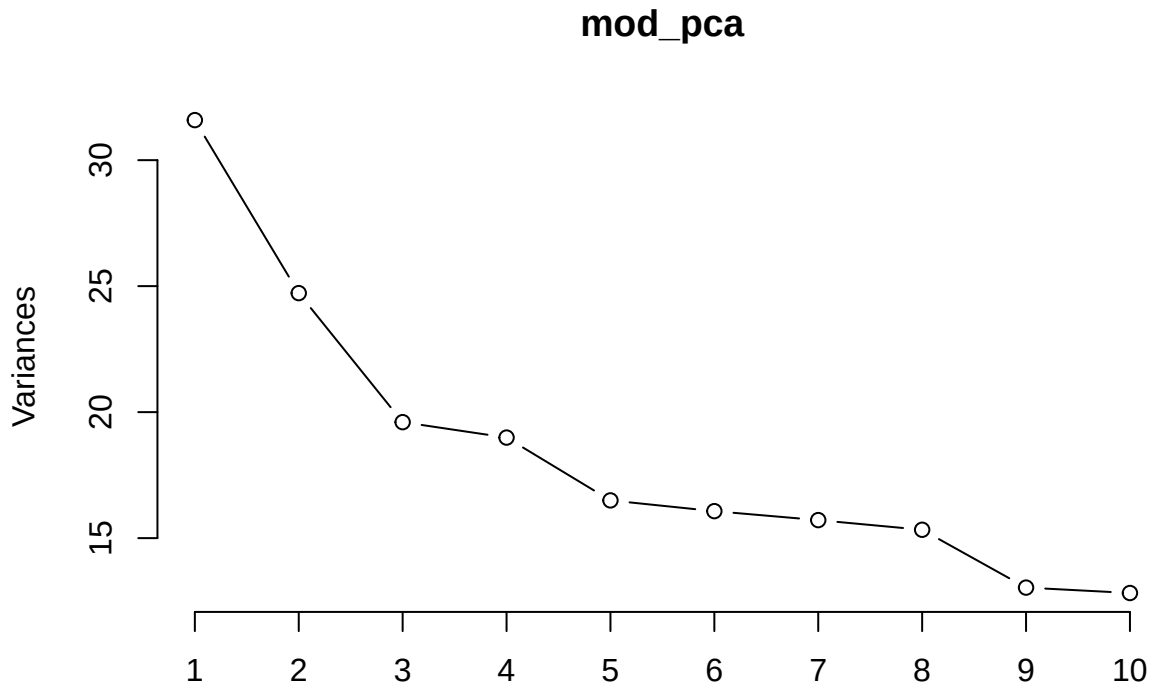
apply PCA function

```
mod_pca = prcomp(DTM_trr_1,scale=TRUE)
pred_pca=predict(mod_pca,newdata = DTM_tss_1)
```

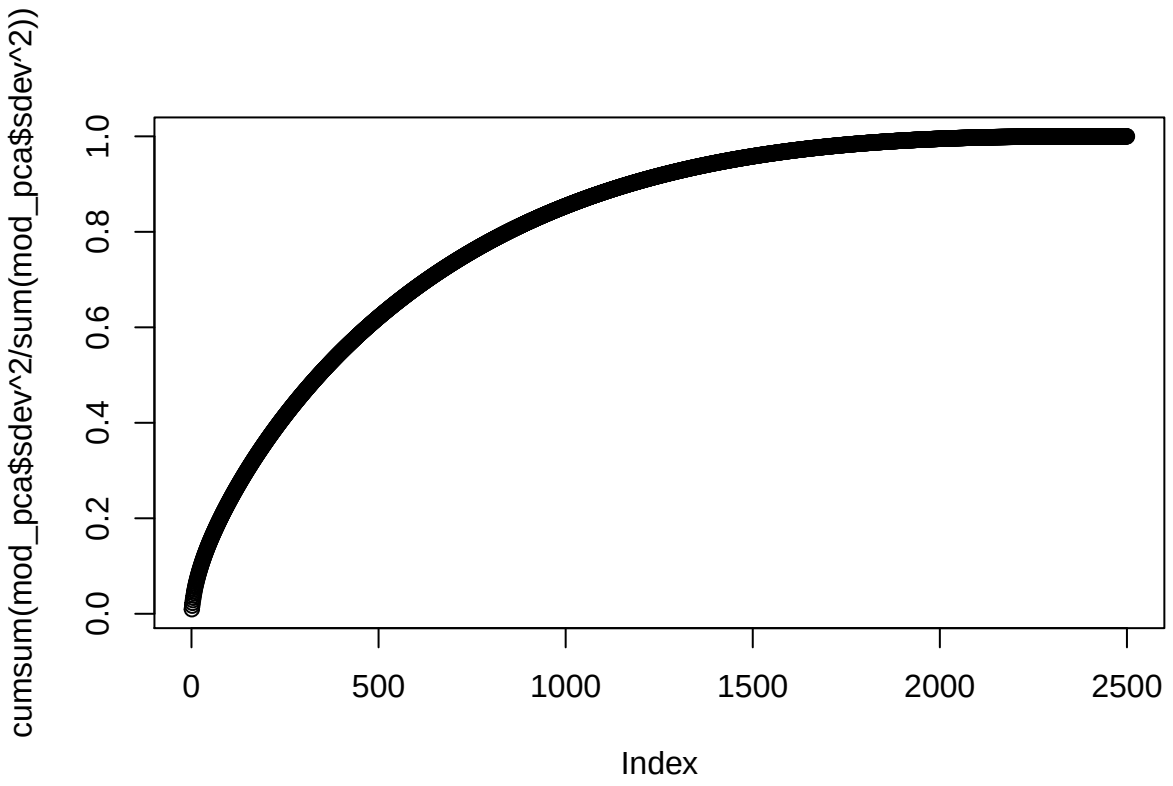
choosing # of PC

##75% variance explained at PC724

```
# 724 out of 2500 principal components since 75% varaince explained
plot(mod_pca,type='line')
```




```
var <- apply(mod_pca$x, 2, var)
prop <- var / sum(var)
#cumsum(prop)
plot(cumsum(mod_pca$sdev^2/sum(mod_pca$sdev^2)))
```



Preparing data for classification models

```
tr_class = data.frame(mod_pca$x[,1:724])
tr_class['author']=labels
tr_load = mod_pca$rotation[,1:724]
ts_class_pre <- scale(DTM_tss_1) %*% tr_load
ts_class <- as.data.frame(ts_class_pre)
ts_class['author']=labels1
```

random forest

What accuracy does Random Forest give us?

##1908 documents is correctly classified by its author; accuracy rate of 76.32%*
test took about 15 min on my computer....

```
pre_rand<-predict(mod_rand,data=ts_class)
tab_rand<-as.data.frame(table(pre_rand,as.factor(ts_class$author)))
predicted<-pre_rand
actual<-as.factor(ts_class$author)
temp<-as.data.frame(cbind(actual,predicted))
temp$flag<-ifelse(temp$actual==temp$predicted,1,0)
sum(temp$flag)
```

```
## [1] 1908
```

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

```
## [1] 76.32
```

Naive Bayes

Fit the training set and test on testing set*

```
library('e1071')
mod_naive=naiveBayes(as.factor(author)~.,data=tr_class)
pred_naive=predict(mod_naive,ts_class)
```

Classification accuracy obtained:

810 documents are correctly classified by its authors, accuracyrate of 32.4%* ## took 1min to run this test

```
library(caret)
predicted_nb=pred_naive
actual_nb=as.factor(ts_class$author)
temp_nb<-as.data.frame(cbind(actual_nb,predicted_nb))
temp_nb$flag<-ifelse(temp_nb$actual_nb==temp_nb$predicted_nb,1,0)
sum(temp_nb$flag)
```

```
## [1] 810
```

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

```
## [1] 32.4
```

Comparing train and test accuracy

Random Forest provides the best accurary at 74%, while Naive Bayes have accurary around 32%

Association Rule Mining

```
library(tidyverse)
library(arules)
```

```
##
## Attaching package: 'arules'
```

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

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

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

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

```
library(arulesViz)
```

```
## Loading required package: grid

## Registered S3 methods overwritten by 'registry':
##   method               from
##   print.registry_field proxy
##   print.registry_entry proxy

## Registered S3 method overwritten by 'seriation':
##   method         from
##   reorder.hclust gclus
```

```
groceries_raw = scan("https://raw.githubusercontent.com/jgscott/STA380/master/data/groceries.txt", what
head(groceries_raw)
```

```
## [1] "citrus fruit,semi-finished bread,margarine,ready soups"
## [2] "tropical fruit,yogurt,coffee"
## [3] "whole milk"
## [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,abrasive cleaner"
```

```
str(groceries_raw)
```

```
##  chr [1:9835] "citrus fruit,semi-finished bread,margarine,ready soups" ...
```

```
summary(groceries_raw)
```

```
##    Length     Class      Mode
##      9835 character character
```

##there are 9835 transactions

##most frequent items are milk, veggie, buns, soda and yogurt in decreasing order

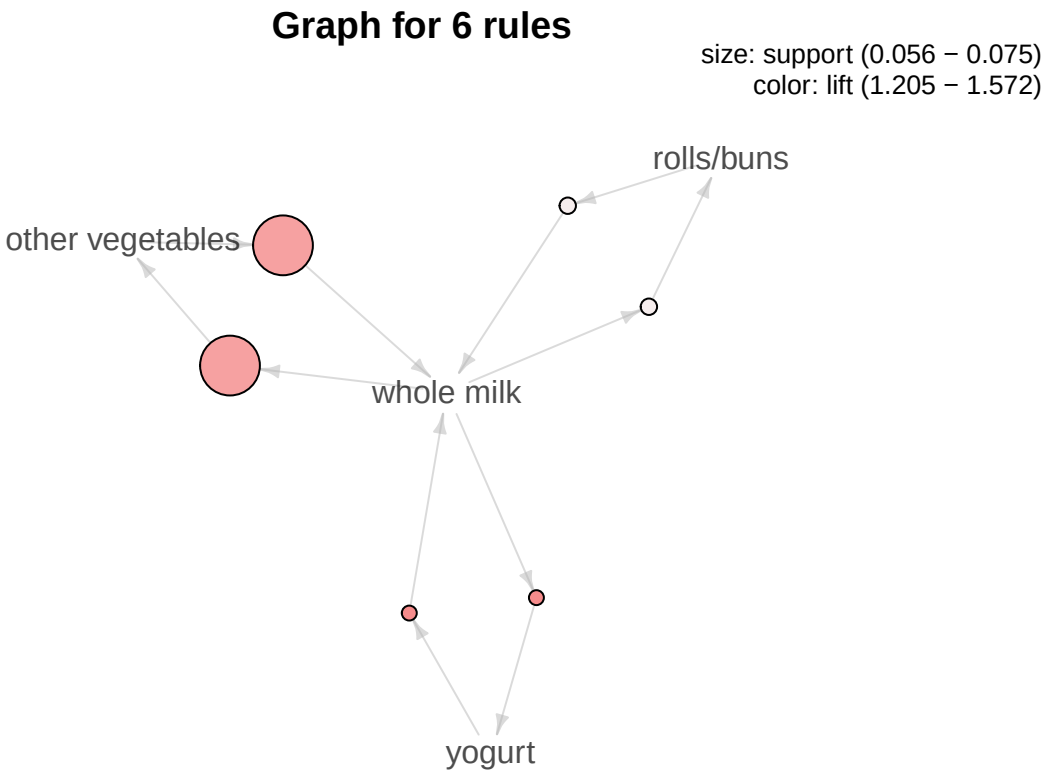
```r
# Now run the 'apriori' algorithm
# Look at rules with support > .05 & confidence >.1 & length min is 2
groctranscrules = apriori(groctrans,
    parameter=list(support=.05, confidence=.1, minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5    0.05      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 491
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [28 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 done [0.00s].
## writing ... [6 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```r
arules::inspect(groctranscrules)
```

```
##     lhs                   rhs                support    confidence coverage
## [1] {yogurt}           => {whole milk}       0.05602440 0.4016035  0.1395018
## [2] {whole milk}       => {yogurt}           0.05602440 0.2192598  0.2555160
## [3] {rolls/buns}       => {whole milk}       0.05663447 0.3079049  0.1839349
## [4] {whole milk}       => {rolls/buns}       0.05663447 0.2216474  0.2555160
## [5] {other vegetables} => {whole milk}       0.07483477 0.3867578  0.1934926
## [6] {whole milk}       => {other vegetables} 0.07483477 0.2928770  0.2555160
##     lift     count
## [1] 1.571735 551
## [2] 1.571735 551
## [3] 1.205032 557
## [4] 1.205032 557
## [5] 1.513634 736
## [6] 1.513634 736
```

```r
plot(groctranscrules, method='graph')
```



We visualize 6 rules. We see most linkage involing veggie, wholemilk, rolls and yogurts. This makes sense since they are most frequently bought.

decrease support to 0.03 and increase confidence to 0.2, min len still 2

```
groctranscrules1 = apriori(groctrans,
                    parameter=list(support=0.03, confidence=.2, minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.2    0.1    1 none FALSE            TRUE       5    0.03      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 295
##
## set item appearances ...[0 item(s)] done [0.00s].
```

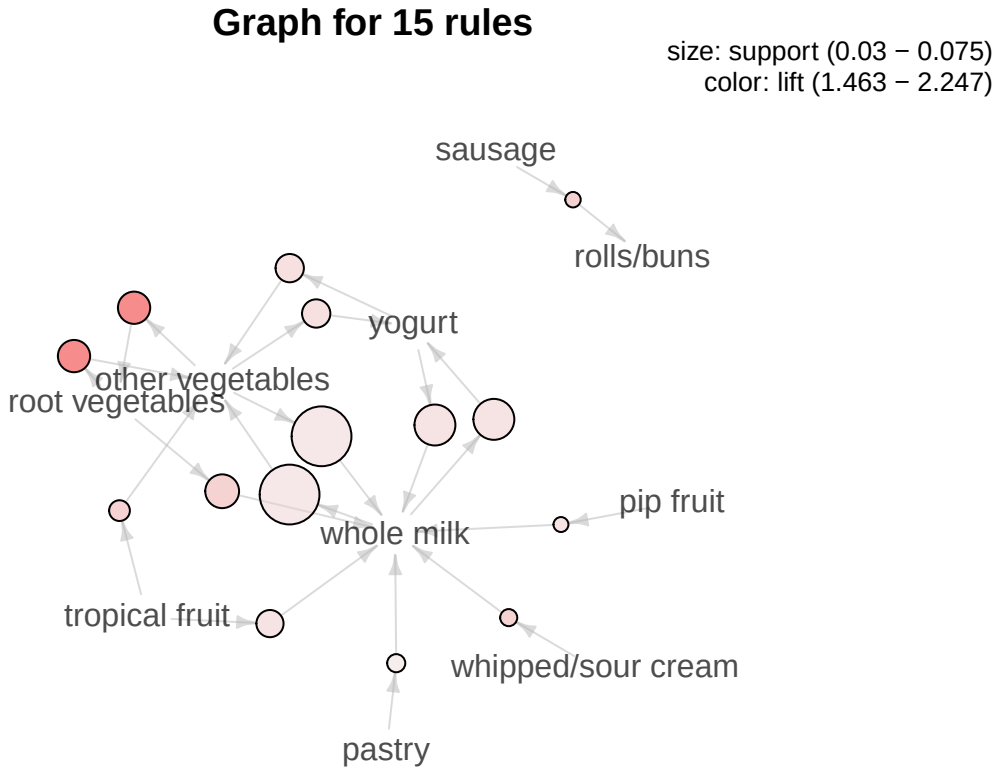
```
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.01s].
## sorting and recoding items ... [44 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 done [0.00s].
## writing ... [25 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
arules::inspect(groctranscrules1)
```

```
##      lhs                     rhs                support    confidence
## [1]  {whipped/sour cream} => {whole milk}       0.03223183 0.4496454
## [2]  {pip fruit}          => {whole milk}       0.03009659 0.3978495
## [3]  {pastry}             => {whole milk}       0.03324860 0.3737143
## [4]  {citrus fruit}       => {whole milk}       0.03050330 0.3685504
## [5]  {sausage}            => {rolls/buns}       0.03060498 0.3257576
## [6]  {bottled water}      => {whole milk}       0.03436706 0.3109476
## [7]  {tropical fruit}     => {other vegetables} 0.03589222 0.3420543
## [8]  {tropical fruit}     => {whole milk}       0.04229792 0.4031008
## [9]  {root vegetables}    => {other vegetables} 0.04738180 0.4347015
## [10] {other vegetables}   => {root vegetables}  0.04738180 0.2448765
## [11] {root vegetables}    => {whole milk}       0.04890696 0.4486940
## [12] {soda}               => {rolls/buns}       0.03833249 0.2198251
## [13] {rolls/buns}         => {soda}             0.03833249 0.2084024
## [14] {soda}               => {whole milk}       0.04006101 0.2297376
## [15] {yogurt}             => {rolls/buns}       0.03436706 0.2463557
## [16] {yogurt}             => {other vegetables} 0.04341637 0.3112245
## [17] {other vegetables}   => {yogurt}           0.04341637 0.2243826
## [18] {yogurt}             => {whole milk}       0.05602440 0.4016035
## [19] {whole milk}         => {yogurt}           0.05602440 0.2192598
## [20] {rolls/buns}         => {other vegetables} 0.04260295 0.2316197
## [21] {other vegetables}   => {rolls/buns}       0.04260295 0.2201787
## [22] {rolls/buns}         => {whole milk}       0.05663447 0.3079049
## [23] {whole milk}         => {rolls/buns}       0.05663447 0.2216474
## [24] {other vegetables}   => {whole milk}       0.07483477 0.3867578
## [25] {whole milk}         => {other vegetables} 0.07483477 0.2928770
##      coverage   lift      count
## [1]  0.07168277 1.7597542 317
## [2]  0.07564820 1.5570432 296
## [3]  0.08896797 1.4625865 327
## [4]  0.08276563 1.4423768 300
## [5]  0.09395018 1.7710480 301
## [6]  0.11052364 1.2169396 338
## [7]  0.10493137 1.7677896 353
## [8]  0.10493137 1.5775950 416
## [9]  0.10899847 2.2466049 466
## [10] 0.19349263 2.2466049 466
## [11] 0.10899847 1.7560310 481
## [12] 0.17437722 1.1951242 377
## [13] 0.18393493 1.1951242 377
## [14] 0.17437722 0.8991124 394
## [15] 0.13950178 1.3393633 338
## [16] 0.13950178 1.6084566 427
## [17] 0.19349263 1.6084566 427
## [18] 0.13950178 1.5717351 551
```

```
## [19] 0.25551601 1.5717351 551
## [20] 0.18393493 1.1970465 419
## [21] 0.19349263 1.1970465 419
## [22] 0.18393493 1.2050318 557
## [23] 0.25551601 1.2050318 557
## [24] 0.19349263 1.5136341 736
## [25] 0.25551601 1.5136341 736
```

```
plot(head(groctranscrules1,15,by='lift'), method='graph')
```



This graph contain 15 rules, and include a little bit more items, however we can see milk still at the center with most assocation, there is a micro center around veggies

```
groctranscrules2 = apriori(groctrans,
                    parameter=list(support=0.01, confidence=.3, minlen=2))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.3    0.1    1 none FALSE            TRUE       5    0.01      2
##  maxlen target  ext
##      10  rules TRUE
##
## Algorithmic control:
##  filter tree heap memopt load sort verbose
```

```
##     0.1 TRUE TRUE  FALSE TRUE    2    TRUE
##
## Absolute minimum support count: 98
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [88 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [125 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
arules::inspect(groctranscrules2)
```

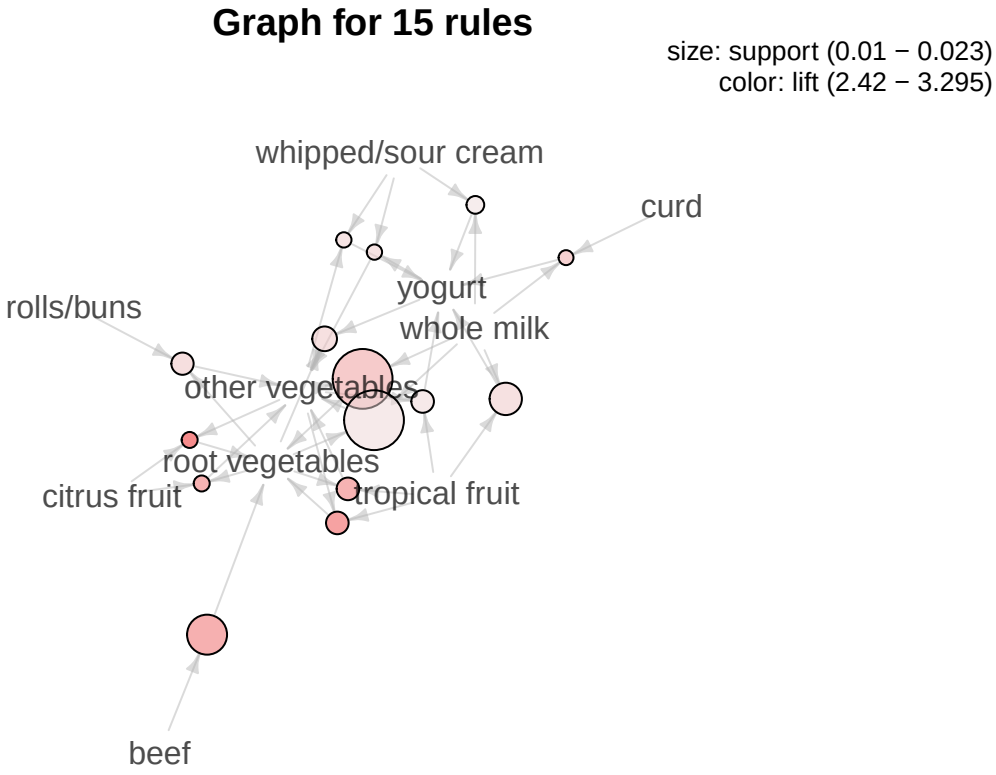
```
##       lhs                           rhs                    support confidence   coverage     lift cou
## [1]   {hard cheese}              => {whole milk}       0.01006609  0.4107884 0.02450432 1.607682    9
## [2]   {butter milk}              => {other vegetables} 0.01037112  0.3709091 0.02796136 1.916916   10
## [3]   {butter milk}              => {whole milk}       0.01159126  0.4145455 0.02796136 1.622385   11
## [4]   {ham}                      => {whole milk}       0.01148958  0.4414062 0.02602949 1.727509   11
## [5]   {sliced cheese}            => {whole milk}       0.01077783  0.4398340 0.02450432 1.721356   10
## [6]   {oil}                      => {whole milk}       0.01128622  0.4021739 0.02806304 1.573968   11
## [7]   {onions}                   => {other vegetables} 0.01423488  0.4590164 0.03101169 2.372268   14
## [8]   {onions}                   => {whole milk}       0.01209964  0.3901639 0.03101169 1.526965   11
## [9]   {berries}                  => {yogurt}           0.01057448  0.3180428 0.03324860 2.279848   10
## [10]  {berries}                  => {other vegetables} 0.01026945  0.3088685 0.03324860 1.596280   10
## [11]  {berries}                  => {whole milk}       0.01179461  0.3547401 0.03324860 1.388328   11
## [12]  {hamburger meat}           => {other vegetables} 0.01382816  0.4159021 0.03324860 2.149447   13
## [13]  {hamburger meat}           => {whole milk}       0.01474326  0.4434251 0.03324860 1.735410   14
## [14]  {hygiene articles}         => {whole milk}       0.01281139  0.3888889 0.03294357 1.521975   12
## [15]  {sugar}                    => {other vegetables} 0.01077783  0.3183183 0.03385867 1.645119   10
## [16]  {sugar}                    => {whole milk}       0.01504830  0.4444444 0.03385867 1.739400   14
## [17]  {waffles}                  => {whole milk}       0.01270971  0.3306878 0.03843416 1.294196   12
## [18]  {long life bakery product} => {whole milk}       0.01352313  0.3614130 0.03741739 1.414444   13
## [19]  {dessert}                  => {other vegetables} 0.01159126  0.3123288 0.03711235 1.614164   11
## [20]  {dessert}                  => {whole milk}       0.01372649  0.3698630 0.03711235 1.447514   13
## [21]  {cream cheese }            => {yogurt}           0.01240468  0.3128205 0.03965430 2.242412   12
## [22]  {cream cheese }            => {other vegetables} 0.01372649  0.3461538 0.03965430 1.788977   13
## [23]  {cream cheese }            => {whole milk}       0.01647178  0.4153846 0.03965430 1.625670   16
## [24]  {chicken}                  => {other vegetables} 0.01789527  0.4170616 0.04290798 2.155439   17
## [25]  {chicken}                  => {whole milk}       0.01759024  0.4099526 0.04290798 1.604411   17
## [26]  {white bread}              => {other vegetables} 0.01372649  0.3260870 0.04209456 1.685268   13
## [27]  {white bread}              => {whole milk}       0.01708185  0.4057971 0.04209456 1.588147   16
## [28]  {chocolate}                => {whole milk}       0.01667514  0.3360656 0.04961871 1.315243   16
## [29]  {coffee}                   => {whole milk}       0.01870869  0.3222417 0.05805796 1.261141   18
## [30]  {frozen vegetables}        => {other vegetables} 0.01779359  0.3699789 0.04809354 1.912108   17
## [31]  {frozen vegetables}        => {whole milk}       0.02043721  0.4249471 0.04809354 1.663094   20
## [32]  {beef}                     => {root vegetables}  0.01738688  0.3313953 0.05246568 3.040367   17
## [33]  {beef}                     => {other vegetables} 0.01972547  0.3759690 0.05246568 1.943066   19
## [34]  {beef}                     => {whole milk}       0.02125064  0.4050388 0.05246568 1.585180   20
## [35]  {curd}                     => {yogurt}           0.01728521  0.3244275 0.05327911 2.325615   17
## [36]  {curd}                     => {other vegetables} 0.01718353  0.3225191 0.05327911 1.666829   16
## [37]  {curd}                     => {whole milk}       0.02613116  0.4904580 0.05327911 1.919481   25
## [38]  {napkins}                  => {whole milk}       0.01972547  0.3766990 0.05236401 1.474268   19
## [39]  {pork}                     => {other vegetables} 0.02165735  0.3756614 0.05765125 1.941476   21
```

```
## [40]  {pork}                    => {whole milk}       0.02216573  0.3844797 0.05765125 1.504719   2
## [41]  {frankfurter}             => {rolls/buns}       0.01921708  0.3258621 0.05897306 1.771616   1
## [42]  {frankfurter}             => {whole milk}       0.02053889  0.3482759 0.05897306 1.363029   2
## [43]  {brown bread}             => {whole milk}       0.02521607  0.3887147 0.06487036 1.521293   2
## [44]  {margarine}               => {other vegetables} 0.01972547  0.3368056 0.05856634 1.740663   1
## [45]  {margarine}               => {whole milk}       0.02419929  0.4131944 0.05856634 1.617098   2
## [46]  {butter}                  => {other vegetables} 0.02003050  0.3614679 0.05541434 1.868122   1
## [47]  {butter}                  => {whole milk}       0.02755465  0.4972477 0.05541434 1.946053   2
## [48]  {newspapers}              => {whole milk}       0.02735130  0.3426752 0.07981698 1.341110   2
## [49]  {domestic eggs}           => {other vegetables} 0.02226741  0.3509615 0.06344687 1.813824   2
## [50]  {domestic eggs}           => {whole milk}       0.02999492  0.4727564 0.06344687 1.850203   2
## [51]  {fruit/vegetable juice}   => {whole milk}       0.02663955  0.3684951 0.07229283 1.442160   2
## [52]  {whipped/sour cream}      => {other vegetables} 0.02887646  0.4028369 0.07168277 2.081924   2
## [53]  {whipped/sour cream}      => {whole milk}       0.03223183  0.4496454 0.07168277 1.759754   3
## [54]  {pip fruit}               => {other vegetables} 0.02613116  0.3454301 0.07564820 1.785237   2
## [55]  {pip fruit}               => {whole milk}       0.03009659  0.3978495 0.07564820 1.557043   2
## [56]  {pastry}                  => {whole milk}       0.03324860  0.3737143 0.08896797 1.462587   3
## [57]  {citrus fruit}            => {other vegetables} 0.02887646  0.3488943 0.08276563 1.803140   2
## [58]  {citrus fruit}            => {whole milk}       0.03050330  0.3685504 0.08276563 1.442377   3
## [59]  {sausage}                 => {rolls/buns}       0.03060498  0.3257576 0.09395018 1.771048   3
## [60]  {sausage}                 => {whole milk}       0.02989324  0.3181818 0.09395018 1.245252   2
## [61]  {bottled water}           => {whole milk}       0.03436706  0.3109476 0.11052364 1.216940   3
## [62]  {tropical fruit}          => {other vegetables} 0.03589222  0.3420543 0.10493137 1.767790   3
## [63]  {tropical fruit}          => {whole milk}       0.04229792  0.4031008 0.10493137 1.577595   4
## [64]  {root vegetables}         => {other vegetables} 0.04738180  0.4347015 0.10899847 2.246605   4
## [65]  {root vegetables}         => {whole milk}       0.04890696  0.4486940 0.10899847 1.756031   4
## [66]  {yogurt}                  => {other vegetables} 0.04341637  0.3112245 0.13950178 1.608457   4
## [67]  {yogurt}                  => {whole milk}       0.05602440  0.4016035 0.13950178 1.571735   5
## [68]  {rolls/buns}              => {whole milk}       0.05663447  0.3079049 0.18393493 1.205032   5
## [69]  {other vegetables}        => {whole milk}       0.07483477  0.3867578 0.19349263 1.513634   7
## [70]  {curd,
##        yogurt}                  => {whole milk}       0.01006609  0.5823529 0.01728521 2.279125   
## [71]  {curd,
##        whole milk}              => {yogurt}           0.01006609  0.3852140 0.02613116 2.761356   
## [72]  {other vegetables,
##        pork}                    => {whole milk}       0.01016777  0.4694836 0.02165735 1.837394   1
## [73]  {pork,
##        whole milk}              => {other vegetables} 0.01016777  0.4587156 0.02216573 2.370714   1
## [74]  {butter,
##        other vegetables}        => {whole milk}       0.01148958  0.5736041 0.02003050 2.244885   1
## [75]  {butter,
##        whole milk}              => {other vegetables} 0.01148958  0.4169742 0.02755465 2.154987   1
## [76]  {domestic eggs,
##        other vegetables}        => {whole milk}       0.01230300  0.5525114 0.02226741 2.162336   1
## [77]  {domestic eggs,
##        whole milk}              => {other vegetables} 0.01230300  0.4101695 0.02999492 2.119820   1
## [78]  {fruit/vegetable juice,
##        other vegetables}        => {whole milk}       0.01047280  0.4975845 0.02104728 1.947371   1
## [79]  {fruit/vegetable juice,
##        whole milk}              => {other vegetables} 0.01047280  0.3931298 0.02663955 2.031756   1
## [80]  {whipped/sour cream,
##        yogurt}                  => {other vegetables} 0.01016777  0.4901961 0.02074225 2.533410   1
## [81]  {other vegetables,
##        whipped/sour cream}      => {yogurt}           0.01016777  0.3521127 0.02887646 2.524073   1
```

```
## [82]  {whipped/sour cream,
##        yogurt}                  => {whole milk}       0.01087951  0.5245098 0.02074225 2.052747   10
## [83]  {whipped/sour cream,
##        whole milk}              => {yogurt}           0.01087951  0.3375394 0.03223183 2.419607   10
## [84]  {other vegetables,
##        whipped/sour cream}      => {whole milk}       0.01464159  0.5070423 0.02887646 1.984385   14
## [85]  {whipped/sour cream,
##        whole milk}              => {other vegetables} 0.01464159  0.4542587 0.03223183 2.347679   14
## [86]  {other vegetables,
##        pip fruit}               => {whole milk}       0.01352313  0.5175097 0.02613116 2.025351   13
## [87]  {pip fruit,
##        whole milk}              => {other vegetables} 0.01352313  0.4493243 0.03009659 2.322178   13
## [88]  {other vegetables,
##        pastry}                  => {whole milk}       0.01057448  0.4684685 0.02257245 1.833421   10
## [89]  {pastry,
##        whole milk}              => {other vegetables} 0.01057448  0.3180428 0.03324860 1.643695   10
## [90]  {citrus fruit,
##        root vegetables}         => {other vegetables} 0.01037112  0.5862069 0.01769192 3.029608   10
## [91]  {citrus fruit,
##        other vegetables}        => {root vegetables}  0.01037112  0.3591549 0.02887646 3.295045   10
## [92]  {citrus fruit,
##        yogurt}                  => {whole milk}       0.01026945  0.4741784 0.02165735 1.855768   10
## [93]  {citrus fruit,
##        whole milk}              => {yogurt}           0.01026945  0.3366667 0.03050330 2.413350   10
## [94]  {citrus fruit,
##        other vegetables}        => {whole milk}       0.01301474  0.4507042 0.02887646 1.763898   13
## [95]  {citrus fruit,
##        whole milk}              => {other vegetables} 0.01301474  0.4266667 0.03050330 2.205080   13
## [96]  {other vegetables,
##        sausage}                 => {whole milk}       0.01016777  0.3773585 0.02694459 1.476849   10
## [97]  {sausage,
##        whole milk}              => {other vegetables} 0.01016777  0.3401361 0.02989324 1.757876   10
## [98]  {bottled water,
##        other vegetables}        => {whole milk}       0.01077783  0.4344262 0.02480935 1.700192   10
## [99]  {bottled water,
##        whole milk}              => {other vegetables} 0.01077783  0.3136095 0.03436706 1.620783   10
## [100] {root vegetables,
##        tropical fruit}          => {other vegetables} 0.01230300  0.5845411 0.02104728 3.020999   12
## [101] {other vegetables,
##        tropical fruit}          => {root vegetables}  0.01230300  0.3427762 0.03589222 3.144780   12
## [102] {root vegetables,
##        tropical fruit}          => {whole milk}       0.01199797  0.5700483 0.02104728 2.230969   12
## [103] {tropical fruit,
##        yogurt}                  => {other vegetables} 0.01230300  0.4201389 0.02928317 2.171343   12
## [104] {other vegetables,
##        tropical fruit}          => {yogurt}           0.01230300  0.3427762 0.03589222 2.457146   12
## [105] {tropical fruit,
##        yogurt}                  => {whole milk}       0.01514997  0.5173611 0.02928317 2.024770   14
## [106] {tropical fruit,
##        whole milk}              => {yogurt}           0.01514997  0.3581731 0.04229792 2.567516   14
## [107] {rolls/buns,
##        tropical fruit}          => {whole milk}       0.01098119  0.4462810 0.02460600 1.746587   10
## [108] {other vegetables,
##        tropical fruit}          => {whole milk}       0.01708185  0.4759207 0.03589222 1.862587   10
```

```
## [109] {tropical fruit,
##        whole milk}              => {other vegetables} 0.01708185  0.4038462 0.04229792 2.087140   10
## [110] {root vegetables,
##        yogurt}                  => {other vegetables} 0.01291307  0.5000000 0.02582613 2.584078   11
## [111] {root vegetables,
##        yogurt}                  => {whole milk}       0.01453991  0.5629921 0.02582613 2.203354   14
## [112] {rolls/buns,
##        root vegetables}         => {other vegetables} 0.01220132  0.5020921 0.02430097 2.594890   11
## [113] {rolls/buns,
##        root vegetables}         => {whole milk}       0.01270971  0.5230126 0.02430097 2.046888   11
## [114] {other vegetables,
##        root vegetables}         => {whole milk}       0.02318251  0.4892704 0.04738180 1.914833   21
## [115] {root vegetables,
##        whole milk}              => {other vegetables} 0.02318251  0.4740125 0.04890696 2.449770   21
## [116] {other vegetables,
##        whole milk}              => {root vegetables}  0.02318251  0.3097826 0.07483477 2.842082   21
## [117] {soda,
##        yogurt}                  => {whole milk}       0.01047280  0.3828996 0.02735130 1.498535   10
## [118] {other vegetables,
##        soda}                    => {whole milk}       0.01392984  0.4254658 0.03274021 1.665124   13
## [119] {soda,
##        whole milk}              => {other vegetables} 0.01392984  0.3477157 0.04006101 1.797049   13
## [120] {rolls/buns,
##        yogurt}                  => {other vegetables} 0.01148958  0.3343195 0.03436706 1.727815   11
## [121] {rolls/buns,
##        yogurt}                  => {whole milk}       0.01555669  0.4526627 0.03436706 1.771563   15
## [122] {other vegetables,
##        yogurt}                  => {whole milk}       0.02226741  0.5128806 0.04341637 2.007235   21
## [123] {whole milk,
##        yogurt}                  => {other vegetables} 0.02226741  0.3974592 0.05602440 2.054131   21
## [124] {other vegetables,
##        rolls/buns}              => {whole milk}       0.01789527  0.4200477 0.04260295 1.643919   17
## [125] {rolls/buns,
##        whole milk}              => {other vegetables} 0.01789527  0.3159785 0.05663447 1.633026   17
```

```
plot(head(groctranscrules2,15,by='lift'), method='graph')
```



Conclusions:

##People likely to purchase sausages if they purchase rolls and buns ##root veggies and other veggies are associated together ##milk is the most common item bought by customers ##when you purchase citrus fruits and tropical fruits, you will buy veggies

##TheEnd##########################

Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.