STA380

2024-08-06

Probability practice

PART A

p(random_clickers)=0.3 p(truthful_clickers)=0.7

p(yes)=0.65 p(no)=0.35

p(yes|random_clicker)=0.5 #randomness p(no|random_clicker)=0.5

P(Y|TC) =?

rule of total probability: P(Y) = P(Y|TC) P(TC) + P(Y|RC) P(RC)

0.65 = P(Y|TC) (0.7) + 0.5 (0.3) P(Y|TC)= .714 The fraction of people who are truthful clickers answered yes is .714.

PART B

sensitivity: p(tests positive|has disease)= .993

specificity: p(tests negative|has NO disease)= 0.9999 p(tests postive| has NO disease)=0.0001 p(has disease)= 0.000025 p(no disease)= 0.999975

p(has disease|tests positive)=?

p(test positive) = 0.0001248224

```
(.993)*(0.000025) + (0.0001)*(.999975)
```

```
## [1] 0.0001248225
```

```
#0.0001248225
```

Bayes theorem: p(has disease|tests positive)= p(has disease) p(tests positive|has disease) / p(tests positive)

```
(.993)*(0.000025) /(0.0001248225) #=0.198824
```

```
## [1] 0.1988824
```

p(has disease|tests positive)=0.198824 Given someone tests positive, the probability that they have the disease is .1989 or approximately 19.89%

Wrangling the Billboard Top 100

PART A

```
library(dplyr)
```

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

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

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

```
billboard=read.csv("/Users/sarahlee/Downloads/billboard.csv")

billboard=billboard%>%
  select(performer, song, year, week, week_position)

top10songs=billboard%>%
  group_by(performer, song)%>%
  summarise(count=n())%>%
  ungroup()%>%
  arrange(desc(count))%>%
  head(10)
```

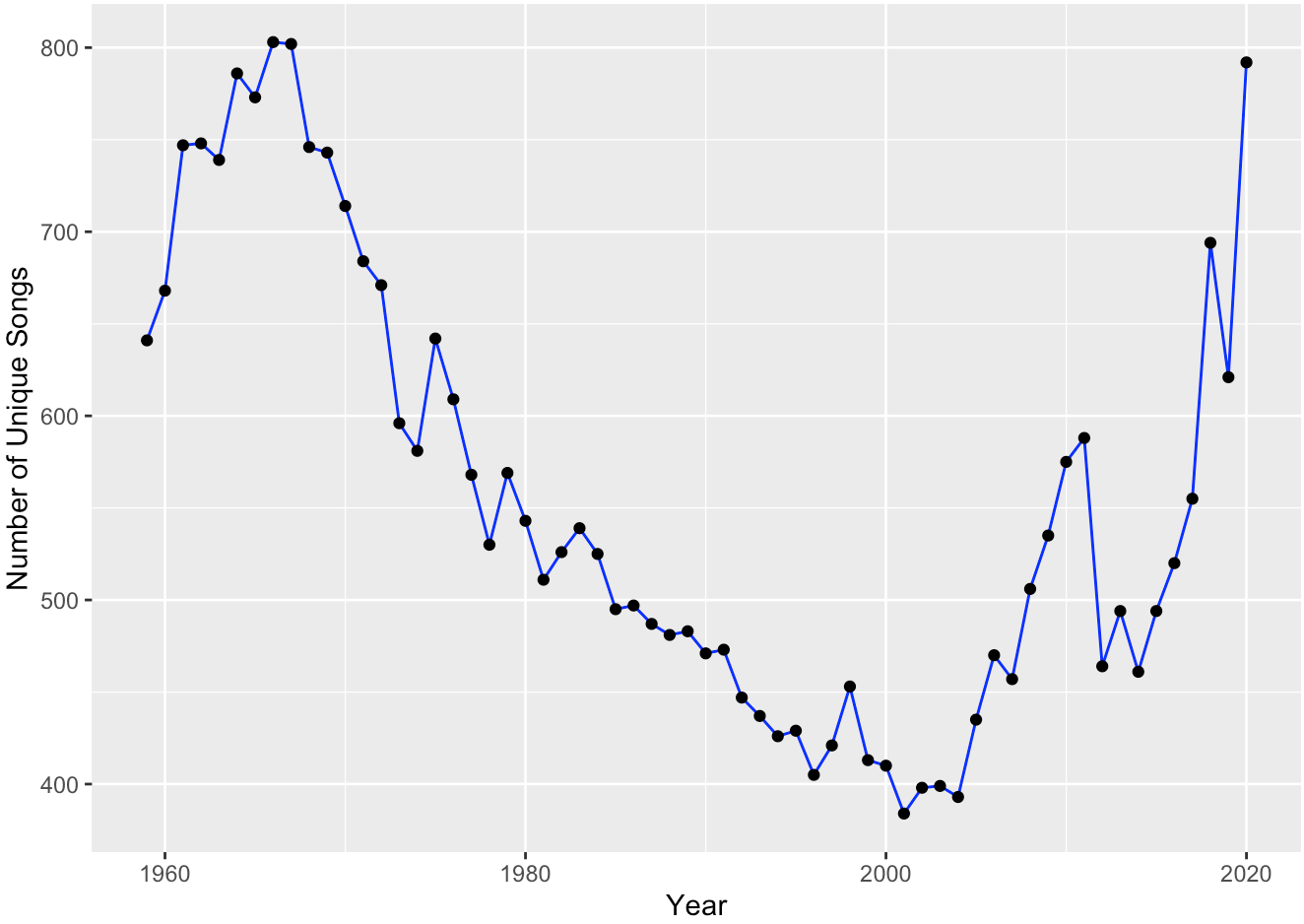
```
## `summarise()` has grouped output by 'performer'. You can override using the
## `.groups` argument.
```

```
top10songs
```

```
## # A tibble: 10 × 3
##    performer                                song                           count
##    <chr>                                    <chr>                          <int>
##  1 Imagine Dragons                          Radioactive                       87
##  2 AWOLNATION                               Sail                              79
##  3 Jason Mraz                               I'm Yours                         76
##  4 The Weeknd                               Blinding Lights                   76
##  5 LeAnn Rimes                              How Do I Live                     69
##  6 LMFAO Featuring Lauren Bennett & GoonRock Party Rock Anthem                68
##  7 OneRepublic                              Counting Stars                    68
##  8 Adele                                    Rolling In The Deep               65
##  9 Jewel                                    Foolish Games/You Were Meant…     65
## 10 Carrie Underwood                         Before He Cheats                  64
```

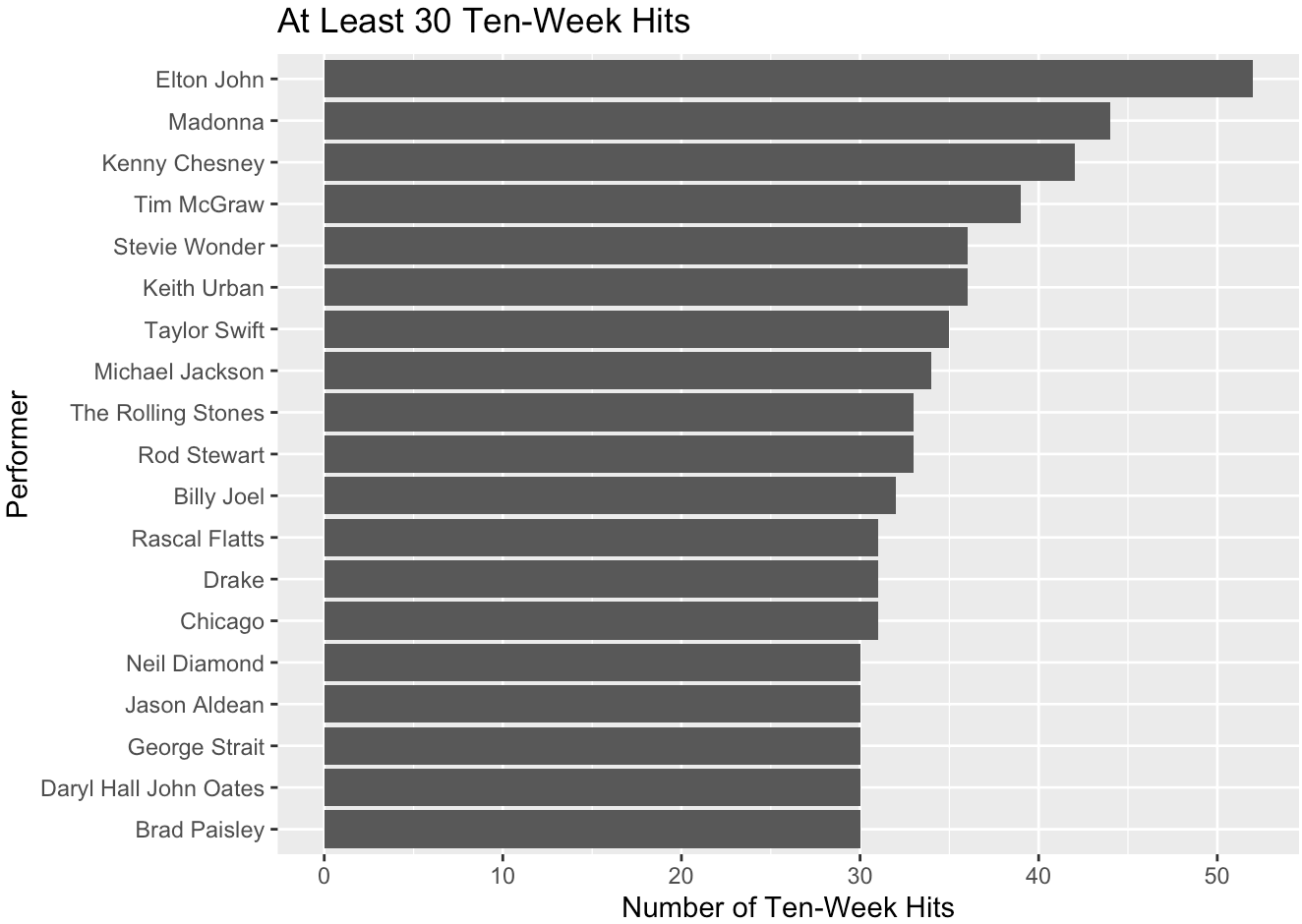
PART B

```
library(ggplot2)
unique_songs_per_year <- billboard %>%
  filter(year != 1958 & year != 2021)%>%
  group_by(year) %>%
  summarise(unique_songs_count = n_distinct(song)) %>%
  ggplot(aes(x = year, y = unique_songs_count)) +
  geom_line(color = "blue") +
  geom_point() +
  labs(x = "Year",
       y = "Number of Unique Songs")

unique_songs_per_year
```



PART C

```
hits <- billboard %>%
  group_by(performer, song) %>%
  summarise(weeks_count = n(), .groups = 'drop') %>%
  filter(weeks_count >= 10)

atleast30hits <- hits %>%
  group_by(performer) %>%
  summarise(hits_count = n(), .groups = 'drop')%>%
  filter(hits_count >= 30)%>%
  ggplot(aes(x = reorder(performer, hits_count), y = hits_count)) +
  geom_bar(stat = "identity") +
  coord_flip() +
  labs(title = "At Least 30 Ten-Week Hits",
       x = "Performer",
       y = "Number of Ten-Week Hits")

atleast30hits
```



Visual story telling part 1: green buildings

```
gb=read.csv("/Users/sarahlee/Downloads/greenbuildings.csv")

gb_filtered=gb%>%
  filter(leasing_rate >= 10)

#median rent
gb_filtered %>%
  group_by(green_rating) %>%
  summarise(median_rent = median(Rent))
```

```
## # A tibble: 2 × 2
##   green_rating median_rent
##          <int>       <dbl>
## 1            0        25.0
## 2            1        27.6
```

```
#mean of green vs non-green
gb_filtered%>%
  group_by(green_rating)%>%
  summarise(across(c(size, empl_gr, Rent, leasing_rate, stories, age, renovated, class_
a, class_b, net, amenities, cd_total_07, hd_total07, Precipitation, Gas_Costs, Electrici
ty_Costs, cluster_rent), mean))%>%
  ungroup()
```

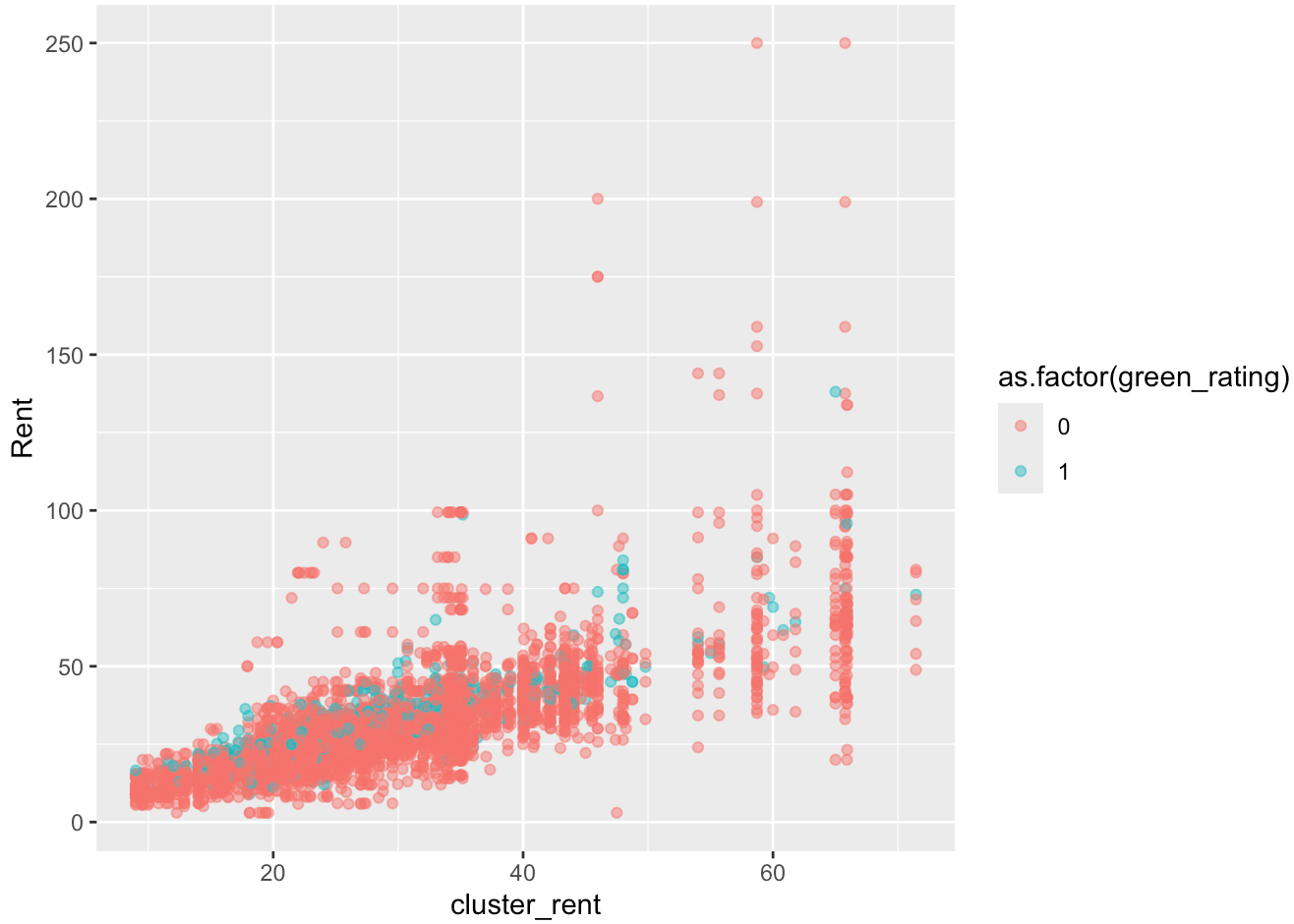
```
## # A tibble: 2 × 18
##   green_rating   size empl_gr  Rent leasing_rate stories   age renovated class_a
##          <int>  <dbl>   <dbl> <dbl>        <dbl>   <dbl> <dbl>     <dbl>   <dbl>
## 1            0 2.31e5      NA  28.4         84.4    13.7  49.3     0.398   0.370
## 2            1 3.26e5      NA  30.0         89.4    15.3  23.9     0.213   0.798
## # ℹ 9 more variables: class_b <dbl>, net <dbl>, amenities <dbl>,
## #   cd_total_07 <dbl>, hd_total07 <dbl>, Precipitation <dbl>, Gas_Costs <dbl>,
## #   Electricity_Costs <dbl>, cluster_rent <dbl>
```

```
#median of green vs non-green
gb_filtered%>%
  group_by(green_rating)%>%
  summarise(across(c(size, empl_gr, Rent, leasing_rate, stories, age, renovated, class_
a, class_b, net, amenities, cd_total_07, hd_total07, Precipitation, Gas_Costs, Electrici
ty_Costs, cluster_rent), median))%>%
  ungroup()
```

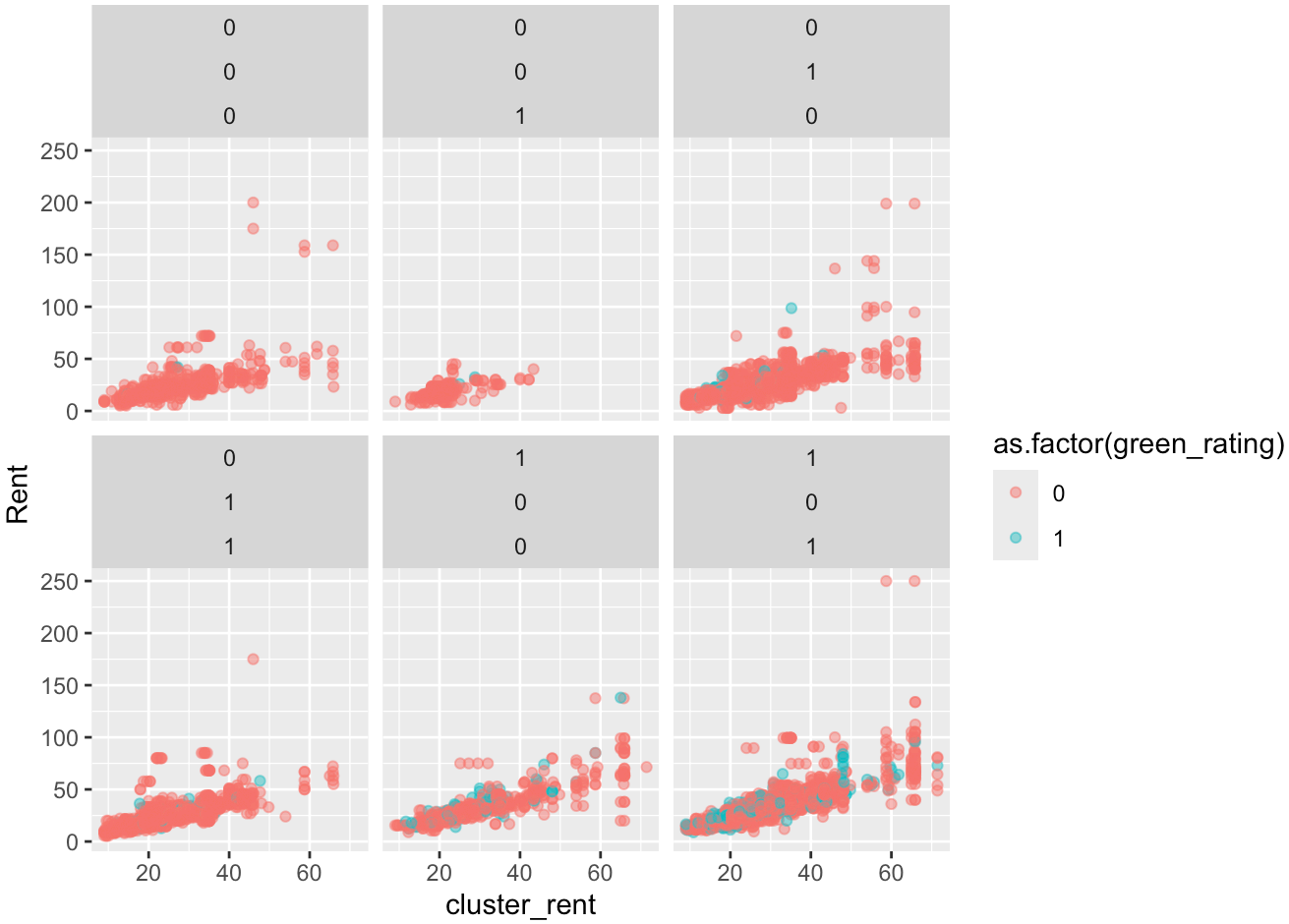
```
## # A tibble: 2 × 18
##   green_rating   size empl_gr  Rent leasing_rate stories   age renovated class_a
##          <int>  <dbl>   <dbl> <dbl>        <dbl>   <dbl> <dbl>     <dbl>   <dbl>
## 1            0 123250      NA  25.0         89.6      10    36         0       0
## 2            1 241199      NA  27.6         92.9      11    22         0       1
## # ℹ 9 more variables: class_b <dbl>, net <dbl>, amenities <dbl>,
## #   cd_total_07 <dbl>, hd_total07 <dbl>, Precipitation <dbl>, Gas_Costs <dbl>,
## #   Electricity_Costs <dbl>, cluster_rent <dbl>
```

According to the mean and medians, we can observe that green buildings have a younger age population, larger size in sqft, better quality buildings (more 'class a') with amenities. This analysis suggests that the difference in rent might not be solely due to the buildings' green rating.

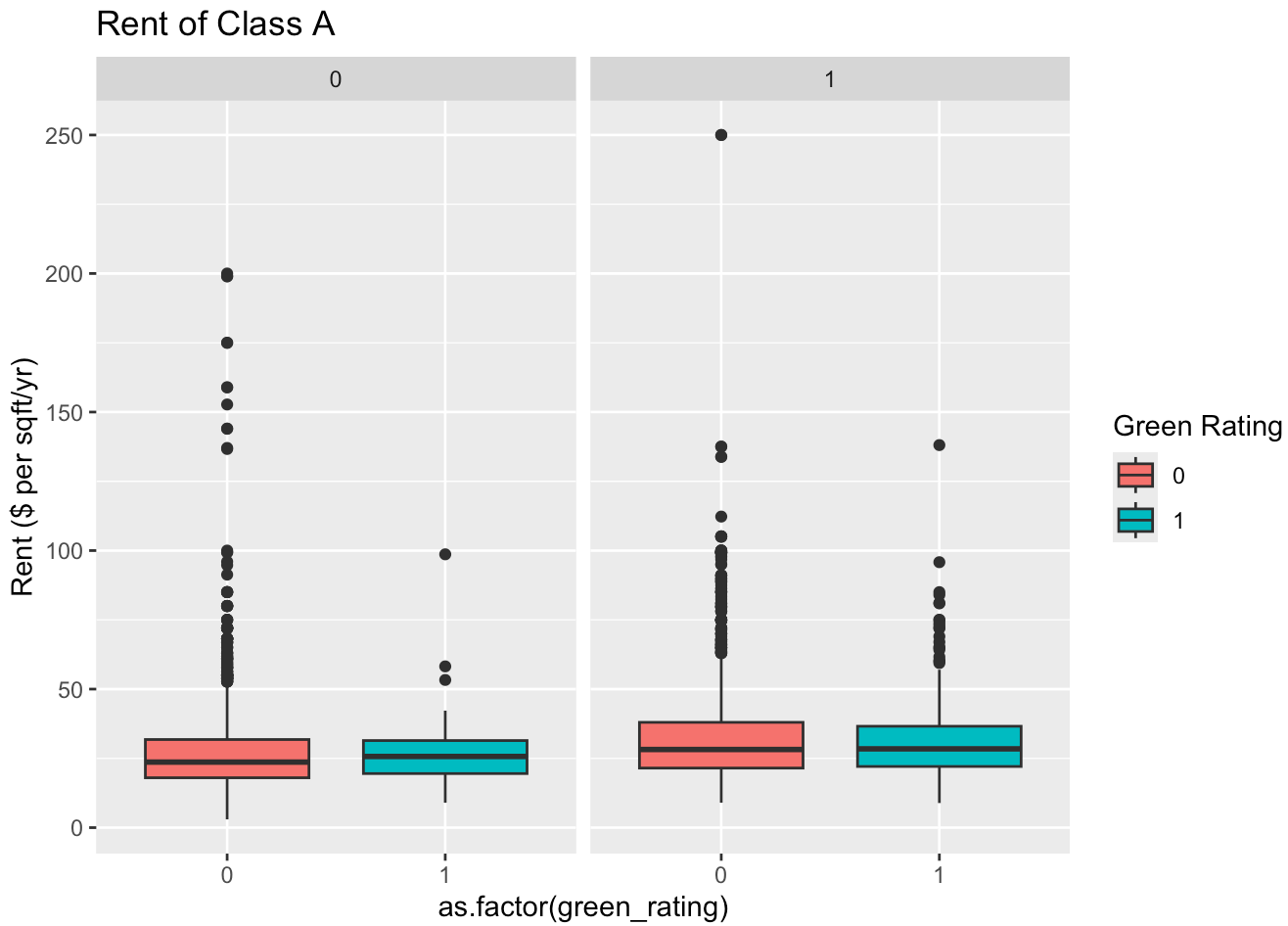
```
#cluster rent vs rent by green rating
ggplot(gb_filtered, aes(x = cluster_rent, y = Rent, color = as.factor(green_rating))) +
  geom_point(alpha = 0.5)
```




```
ggplot(gb_filtered, aes(x = cluster_rent, y = Rent, color = as.factor(green_rating))) +
  geom_point(alpha = 0.5) +
  facet_wrap(~ class_a + class_b + amenities)
```




```
#by class a
ggplot(gb_filtered, aes(x = as.factor(green_rating), y = Rent, fill = as.factor(green_ra
ting))) +
  geom_boxplot() +
  facet_wrap(~ class_a) +
  labs(title = "Rent of Class A",
       y = "Rent ($ per sqft/yr)",
       fill = "Green Rating")
```

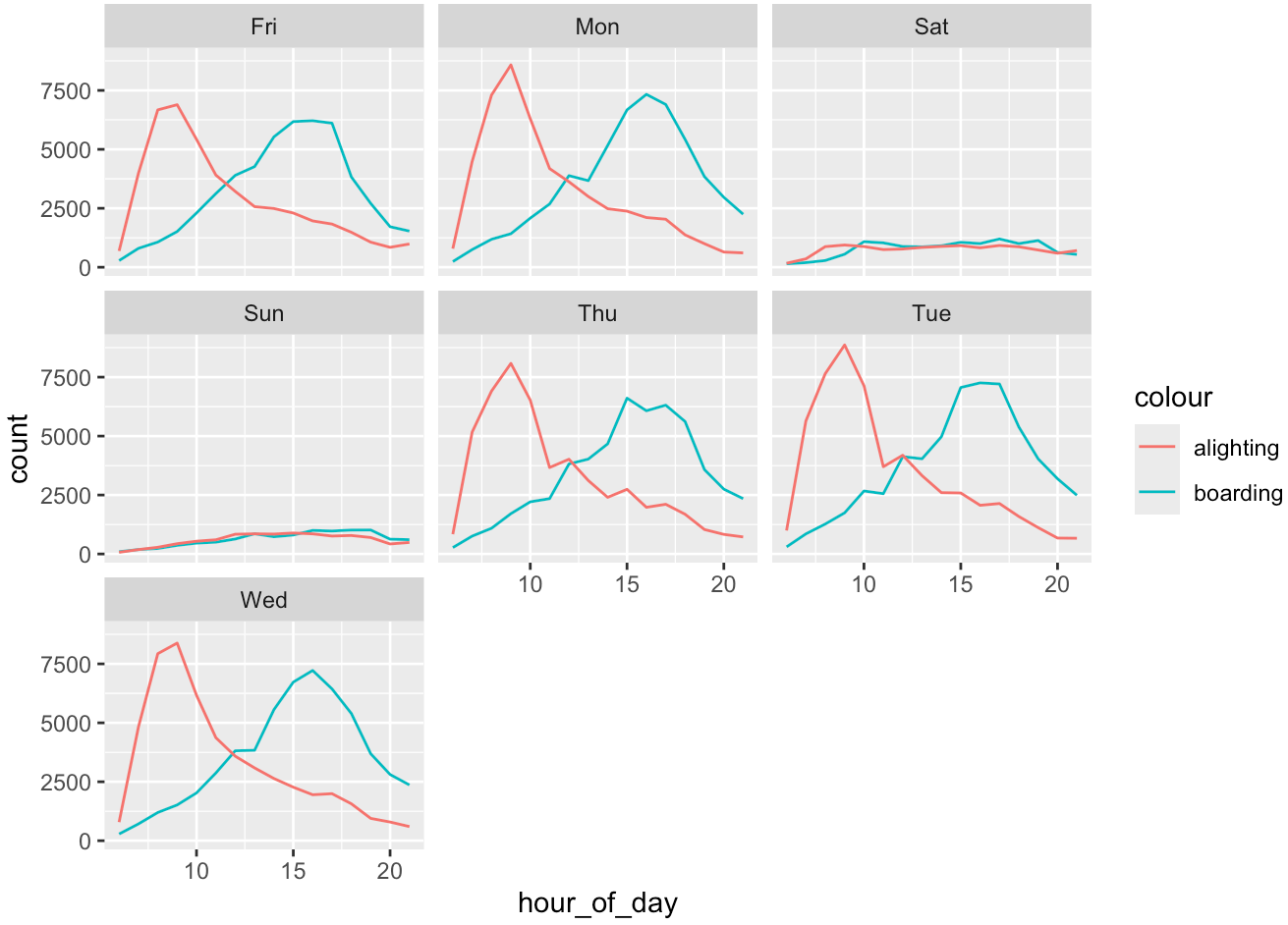


Although I agree with guru's median observation, I observed additional points that might interfere with his conclusions. After looking at both mean and median distributions across the various variables, I saw that green buildings have a younger age population, larger size in sqft, better quality buildings (more 'class a') with amenities. Furthermore, to assess if these variables could potentially be confounding, I adjusted the data accounting for these variables. However, rent premium for green buildings did not show a clear premium against non green buildings for class_a, class_b, and amenities. When adjusting for these variables, the rent premium for green buildings does not appear as significant, especially when focusing on Class A buildings. Therefore, the financial advantage of investing in green buildings might not be as clear as initially suggested. The rent premium that green buildings command could be largely attributed to their associated characteristics rather than their green certification alone.

Visual story telling part 2: Capital Metro data

```
capmetro=read.csv("/Users/sarahlee/Downloads/capmetro_UT.csv")

capmetro%>%
  group_by(day_of_week,hour_of_day)%>%
  summarize(total_boarding=sum(boarding),
            total_alighting=sum(alighting))%>%
  ungroup()%>%
  ggplot(aes(x=hour_of_day))+
  geom_line(aes(y=total_boarding,color="boarding"))+
  geom_line(aes(y=total_alighting,color="alighting"))+
  facet_wrap(~ day_of_week)+
  labs(y="count")
```

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



#The plots represent the total boarding and alighting counts through out each operating hour of each days of the week. Generally, we see that most activities occur during the w eekdays. Furthermore, during these weekdays, we see a general trend of higher alighting counts during the earlier hours of the day and higher boarding counts during the later h ours of the day.

```
capmetro%>%
  group_by(temperature,hour_of_day)%>%
  summarize(tot_boarding=sum(boarding),
            tot_alighting=sum(alighting))%>%
  ungroup()%>%
  ggplot(aes(x=temperature))+
  geom_line(aes(y=tot_boarding,color="boarding"))+
  geom_line(aes(y=tot_alighting,color="alighting"))+
  facet_wrap(~ hour_of_day)+
  labs(y="count")
```

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

#The plots represent the total alighting and boarding counts compared to temperature for each operating hours. Generally, during the earlier hours we see higher rates of alighting as more students get to campus for the day after a similar counts for both alighting and boarding during the midday, we see higher rates of boarding as many people use the metro to leave the campus to go back home. Although most of the peaks occured was during the higher temperatures, there wasn't a significant impact or trend of temperature.

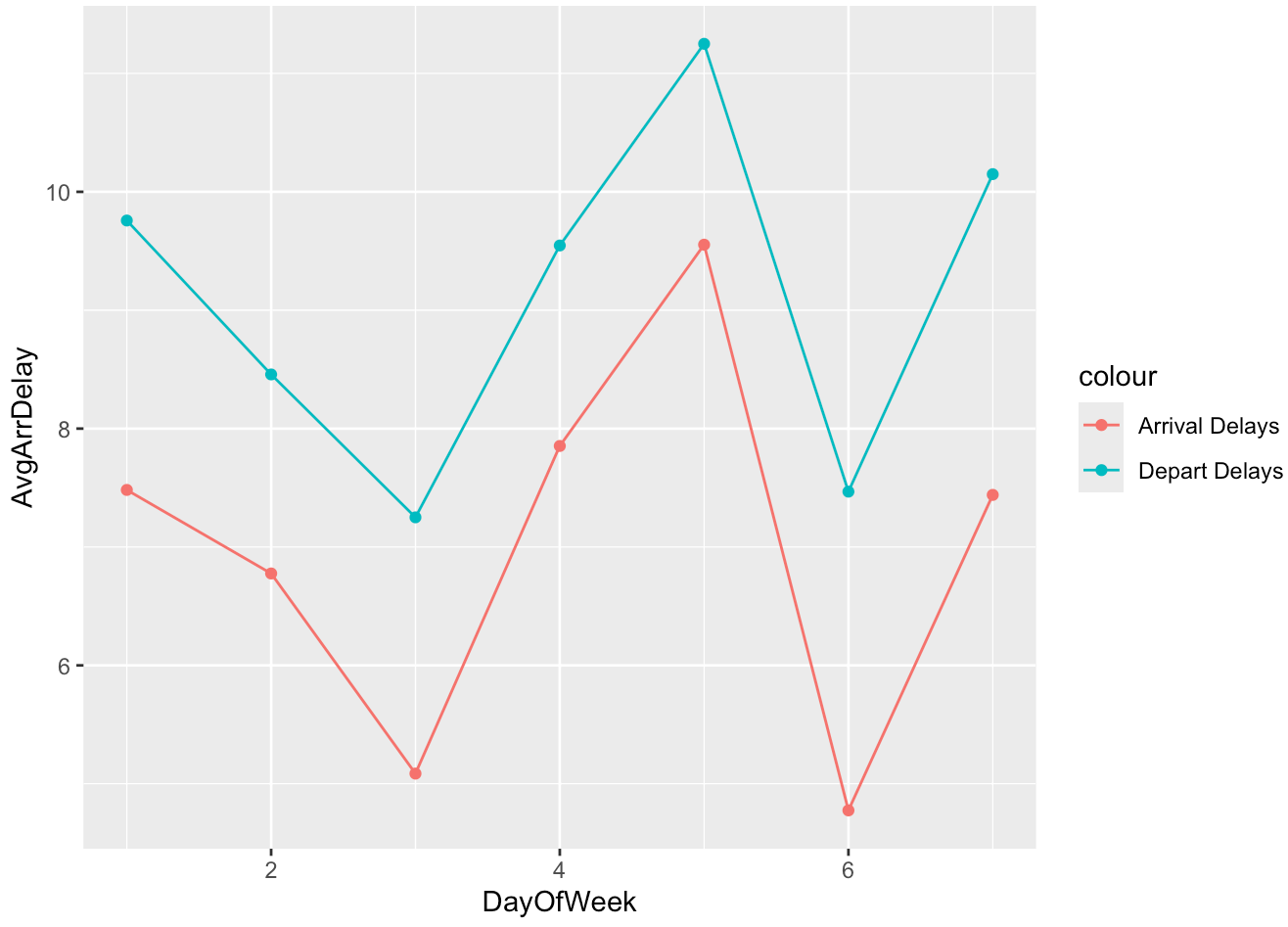
Visual story telling part 2: flights at ABIA

```
library(tidyr)
```

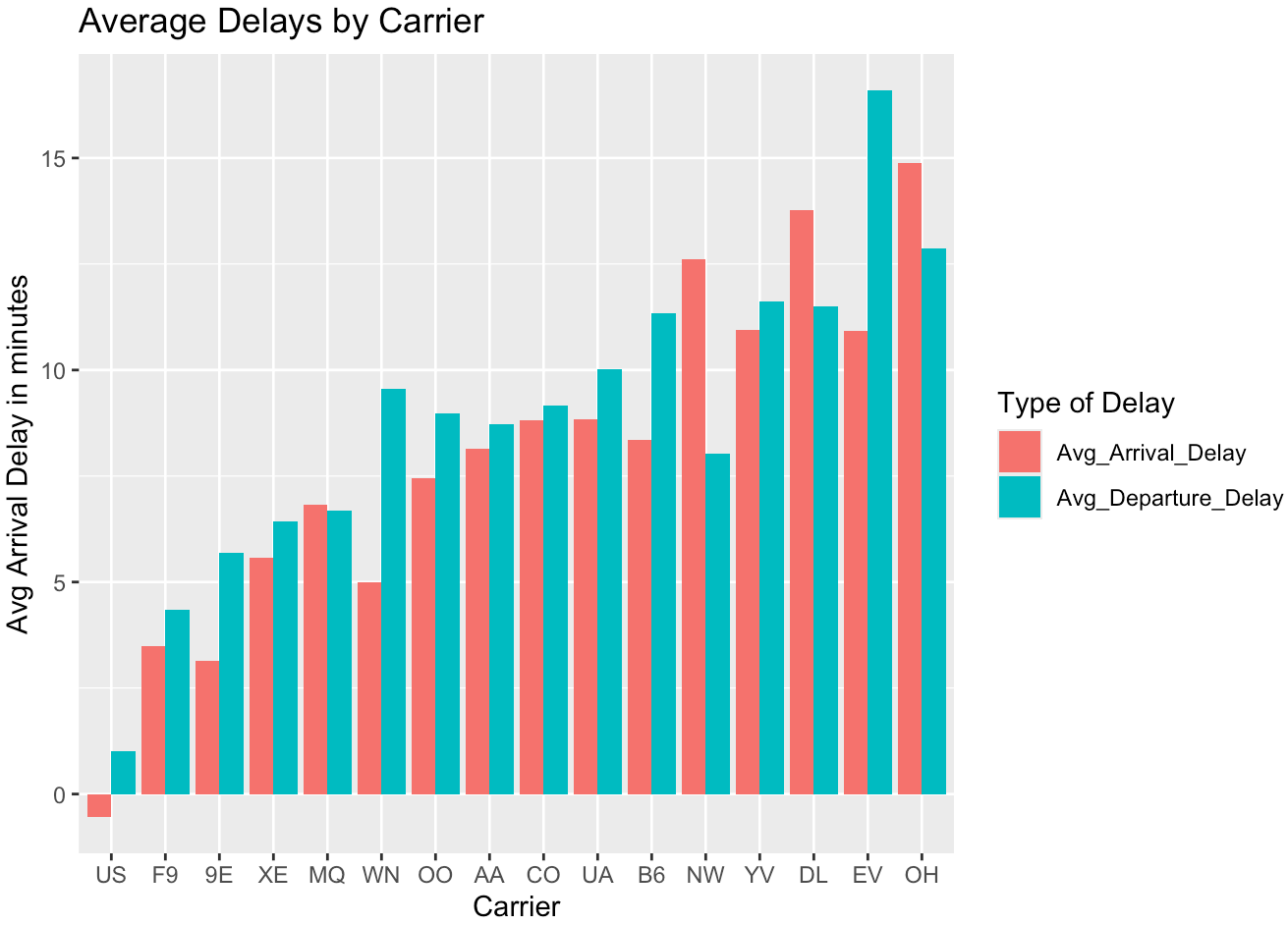
```
##
## Attaching package: 'tidyr'
```

```
## The following object is masked _by_ '.GlobalEnv':
##
##     billboard
```

```
abia=read.csv("/Users/sarahlee/Downloads/ABIA.csv")

abia%>%
  group_by(DayOfWeek)%>%
  summarize(AvgArrDelay=mean(ArrDelay, na.rm=TRUE),
            AvgDepDelay=mean(DepDelay, na.rm=TRUE)) %>%
  ggplot(aes(x=DayOfWeek))+
  geom_point(aes(y=AvgArrDelay, color="Arrival Delays"),stat = "identity")+
  geom_point(aes(y=AvgDepDelay, color="Depart Delays"),stat = "identity")+
  geom_line(aes(y=AvgArrDelay,color="Arrival Delays"))+
  geom_line(aes(y=AvgDepDelay,color="Depart Delays"))
```



```
abia%>%
  group_by(UniqueCarrier)%>%
  summarize(Avg_Arrival_Delay=mean(ArrDelay, na.rm=TRUE),
            Avg_Departure_Delay=mean(DepDelay, na.rm=TRUE),
            flightcount=n()) %>%
  pivot_longer(cols = c(Avg_Arrival_Delay, Avg_Departure_Delay), names_to = "DelayType",
values_to = "Average_Delay")%>%
  ggplot(aes(x = reorder(UniqueCarrier, Average_Delay), y = Average_Delay, fill = DelayT
ype))+
  geom_bar(stat = "identity", position = "dodge")+
  labs(title="Average Delays by Carrier",
       x="Carrier",
       y="Avg Arrival Delay in minutes",
       fill="Type of Delay")
```



Portfolio modeling

```
library(mosaic)
```

```
## 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.
```

```
##
## 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(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
```

```
##
## ######################### Warning from 'xts' package #########################
## #                                                                            #
## # The dplyr lag() function breaks how base R's lag() function is supposed to #
## # work, which breaks lag(my_xts). Calls to lag(my_xts) that you type or      #
## # source() into this session won't work correctly.                           #
## #                                                                            #
## # Use stats::lag() to make sure you're not using dplyr::lag(), or you can add #
## # conflictRules('dplyr', exclude = 'lag') to your .Rprofile to stop          #
## # dplyr from breaking base R's lag() function.                               #
## #                                                                            #
## # Code in packages is not affected. It's protected by R's namespace mechanism #
## # Set `options(xts.warn_dplyr_breaks_lag = FALSE)` to suppress this warning. #
## #                                                                            #
## ##############################################################################
```

```
##
## Attaching package: 'xts'
```

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

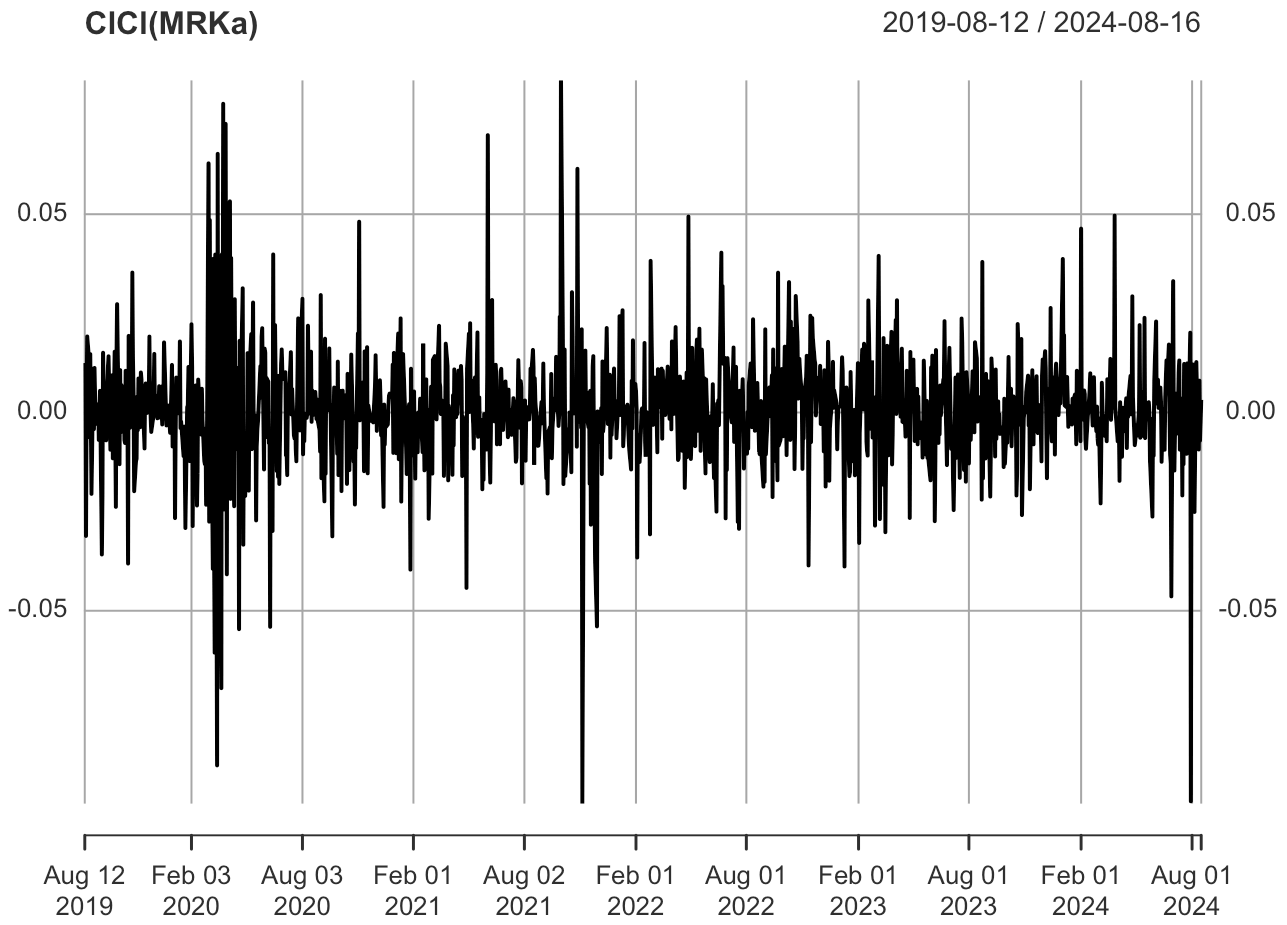
```
## Loading required package: TTR
```

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

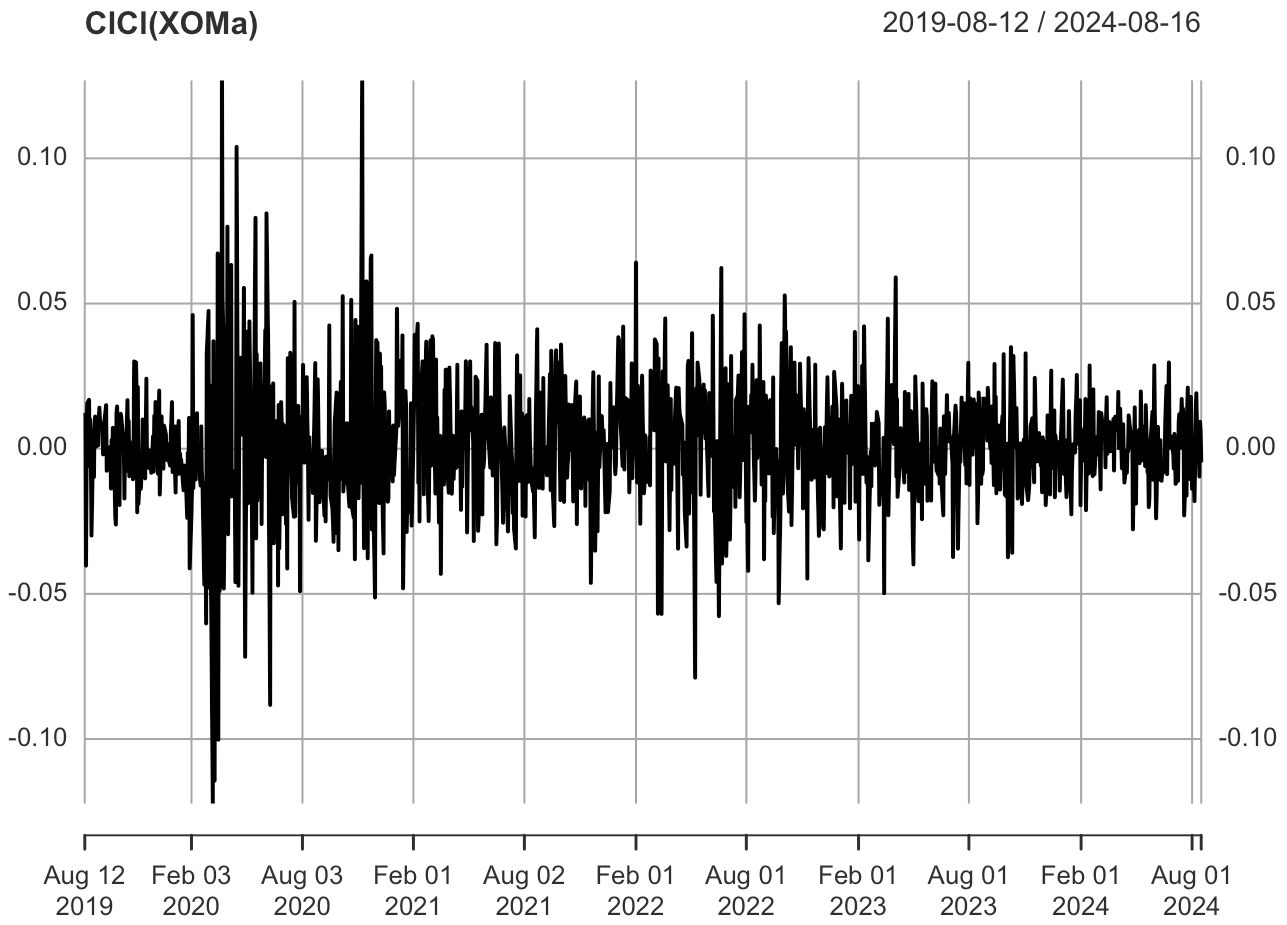
```
library(foreach)

mystocks = c("MRK", "XOM", "SPY")
getSymbols(mystocks, from="2019-08-12")
```

```
## [1] "MRK" "XOM" "SPY"
```

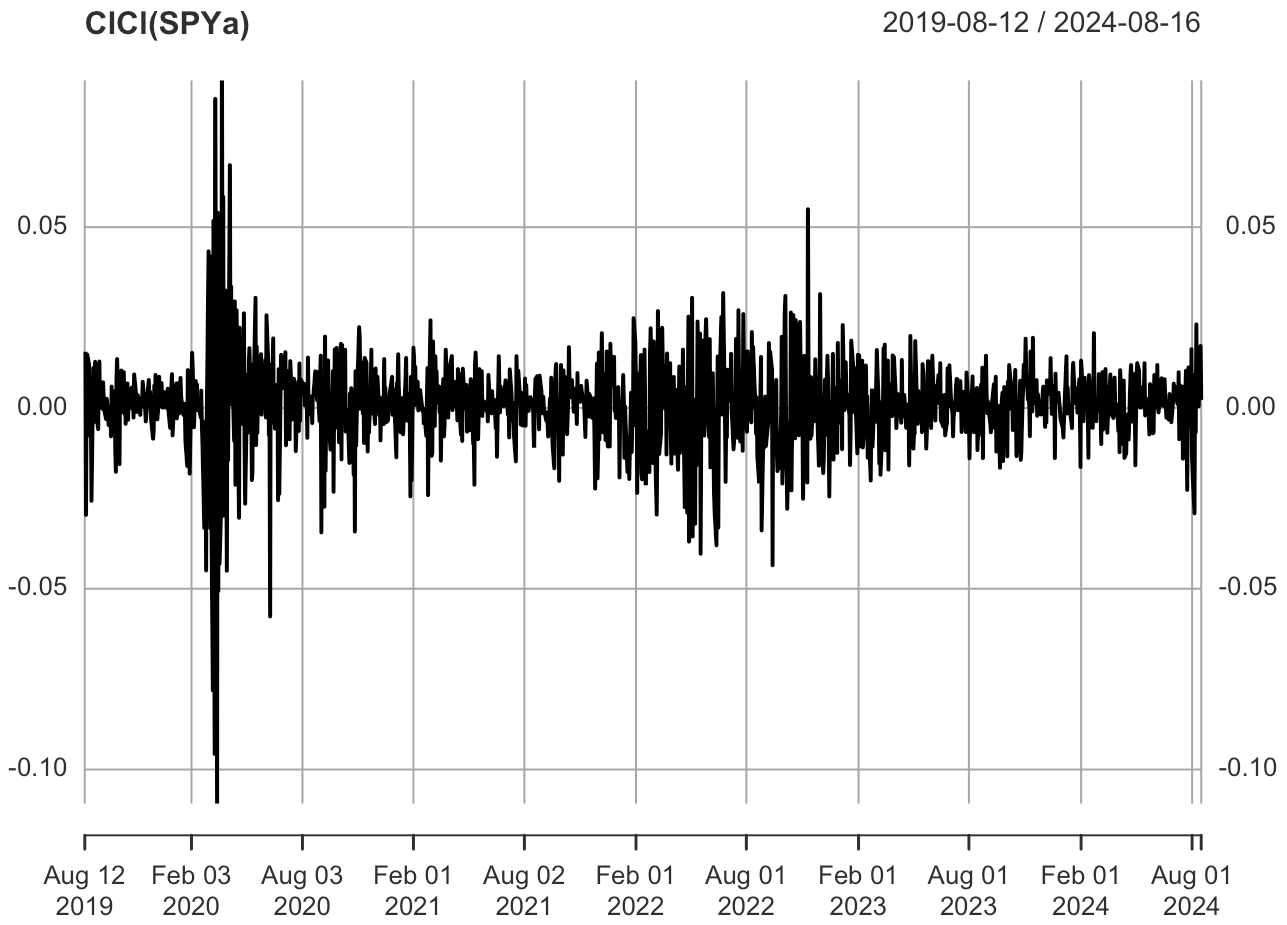
```
MRKa = adjustOHLC(MRK)
XOMa = adjustOHLC(XOM)
SPYa = adjustOHLC(SPY)

for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
plot(ClCl(MRKa))
```



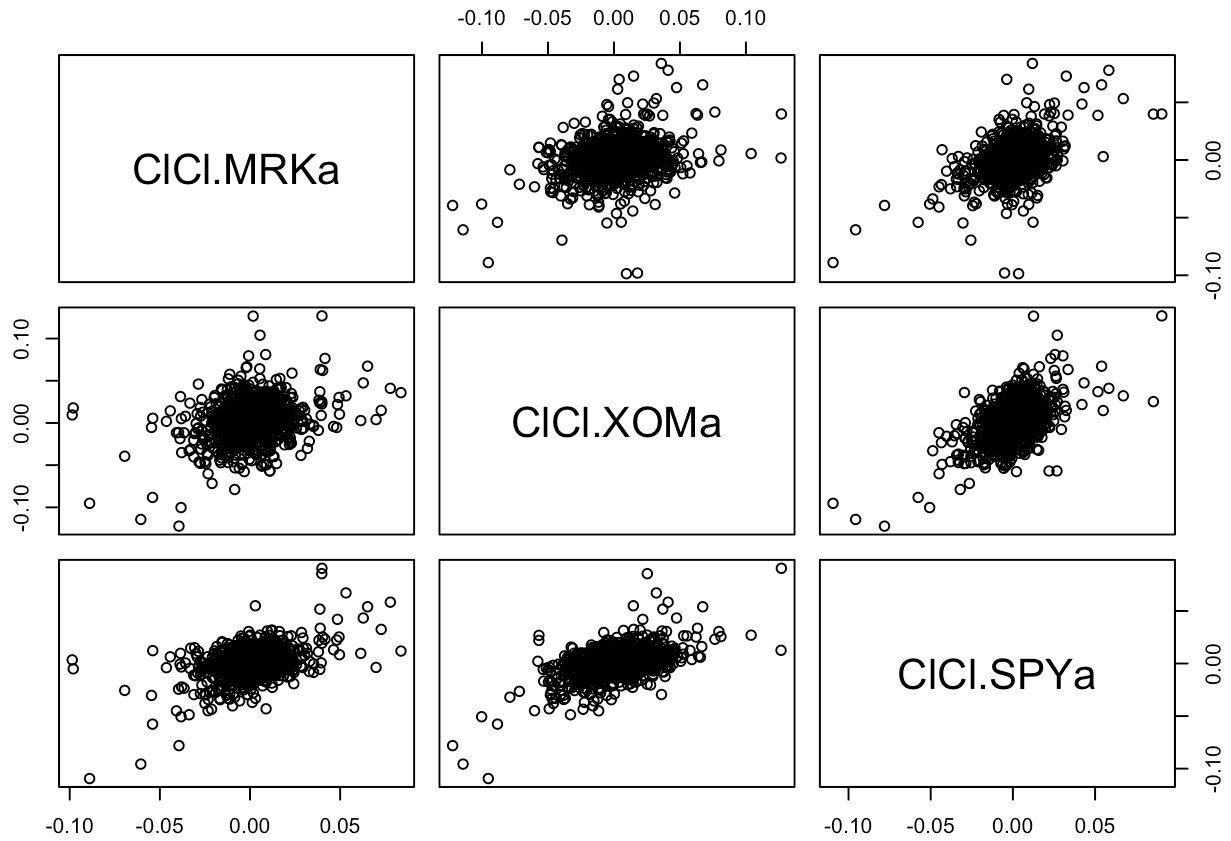

```
plot(ClCl(XOMa))
```




```
plot(ClCl(SPYa))
```



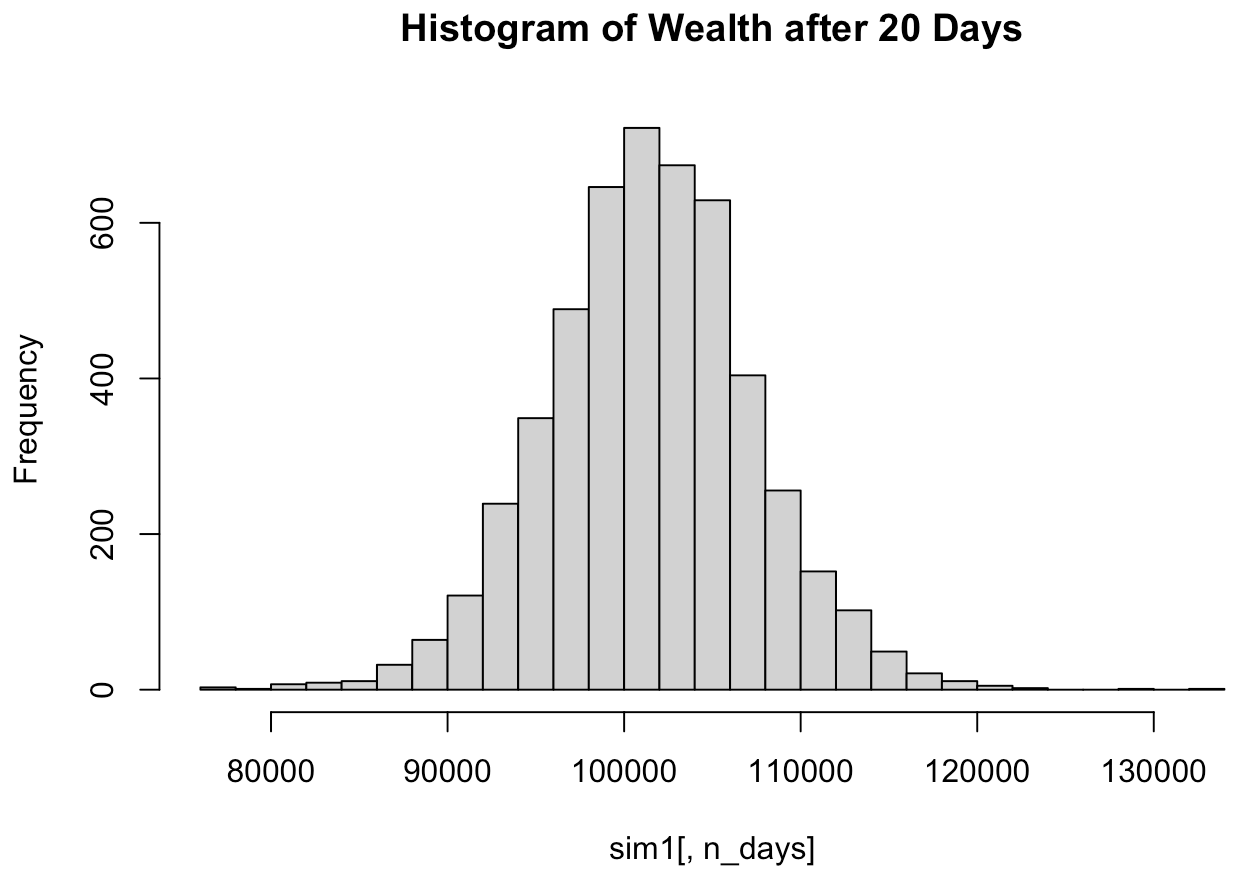

```
all_returns = cbind(ClCl(MRKa),ClCl(XOMa),ClCl(SPYa))
all_returns = as.matrix(na.omit(all_returns))

pairs(all_returns)
```




```
#simulates a random day
return.today = resample(all_returns, 1, orig.ids=FALSE)

total_wealth = 100000
my_weights = c(0.33,0.33,0.34)
holdings = total_wealth*my_weights
holdings = holdings*(1 + return.today)

initial_wealth = 100000
sim1 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.33,0.33,0.34)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}

head(sim1)
```

```
##               [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  99239.62  99083.79  99048.27  99388.92  99894.03  99479.37  97146.35
## result.2  97916.57  96163.95  95605.96  95404.13  95060.25  94860.90  95682.98
## result.3 102265.85 103994.53 105709.85 106386.84 106945.05 105991.83 106459.88
## result.4 101828.27 103141.40 103456.98 103643.31 102772.94 103499.83 104420.32
## result.5 100516.52 104286.53 103980.41 104069.19 104101.95 104493.65 104233.24
## result.6 100769.83 101443.22 102734.89 103225.42 103869.33 103558.01 104161.80
##               [,8]      [,9]    [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  97699.10  98409.51  98335.4  95350.15  95878.17  94636.84  94492.97
## result.2  94857.23  93699.72  92544.4  90863.91  90675.70  89683.46  84078.20
## result.3 105611.97 105834.47 105354.3 105601.08 105979.03 105520.61 106012.16
## result.4 104645.35 105321.13 105839.2 105167.44 104833.24 106353.61 106282.82
## result.5 103711.49 103700.30 100856.7 101809.33 101226.82 101107.13 101196.98
## result.6 108057.75 108284.98 108599.9 108675.12 109778.10 109866.34 109152.86
##              [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  94135.70  94856.40  95807.30  97995.87  98240.58  99535.57
## result.2  83770.27  83938.66  83802.18  82719.84  82727.21  82486.53
## result.3 107407.28 108009.09 112086.98 112969.25 113577.77 113532.22
## result.4 106610.13 106444.23 106747.79 104227.72 103734.81 104829.67
## result.5  99499.73 100344.83 102106.19 101325.03 101206.31 101543.29
## result.6 109268.41 110952.71 112688.53 111391.34 111524.85 112266.84
```

```
hist(sim1[,n_days], 25,
     main = "Histogram of Wealth after 20 Days")
```



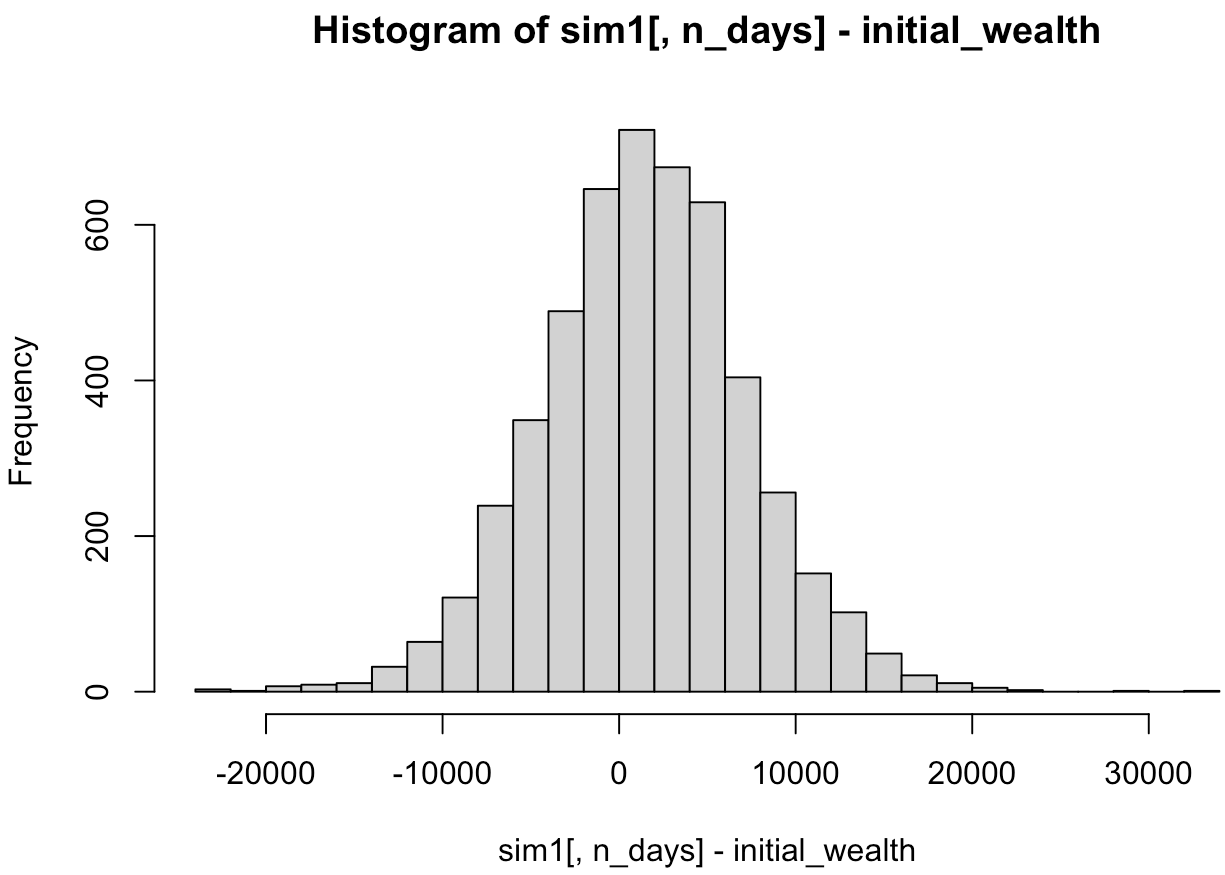
```
# final welath
mean(sim1[,n_days])
```

```
## [1] 101465.8
```

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

```
## [1] 1465.827
```

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



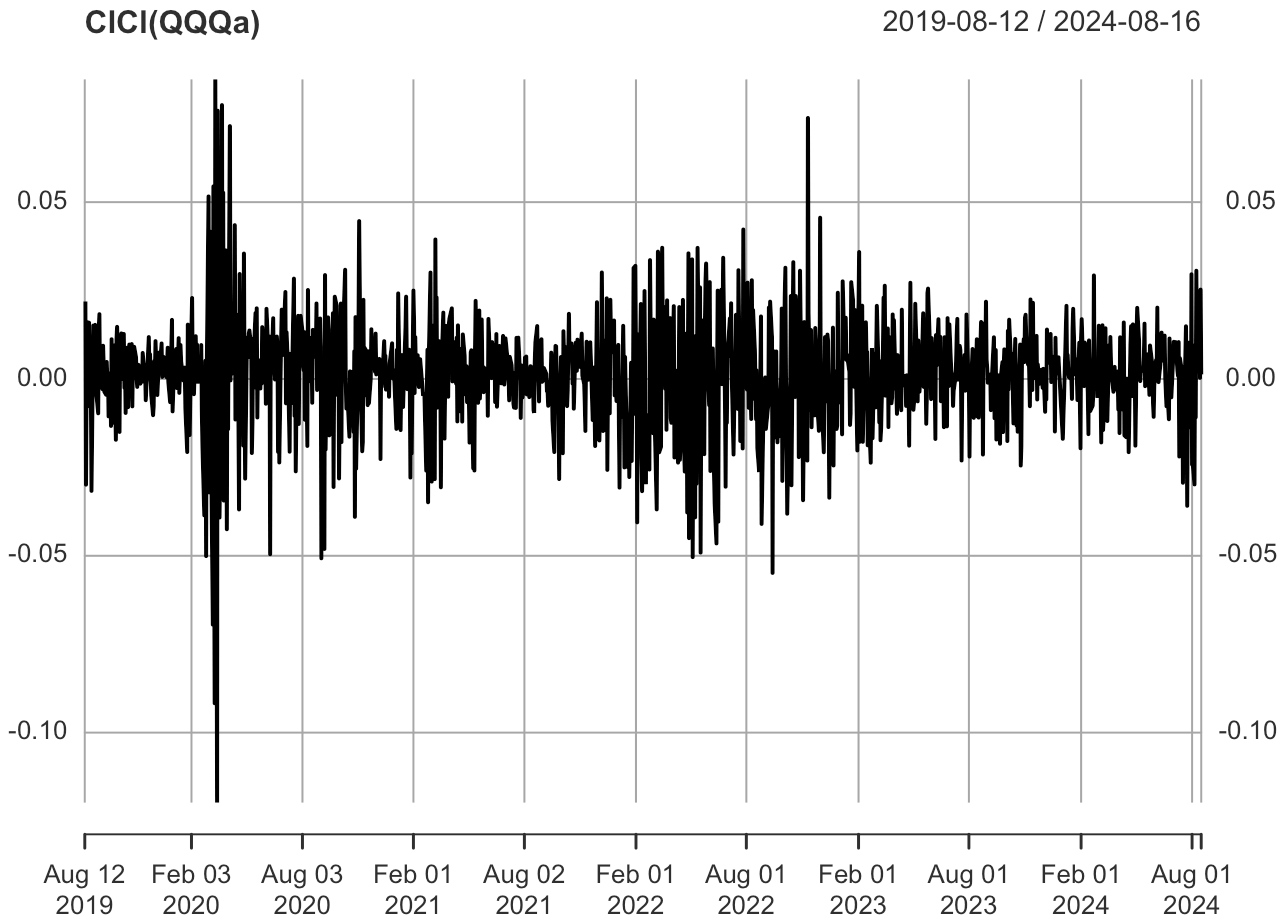

```
#5% VaR
quantile(sim1[,n_days]- initial_wealth, prob=0.05)
```

```
##        5%
## -7955.972
```

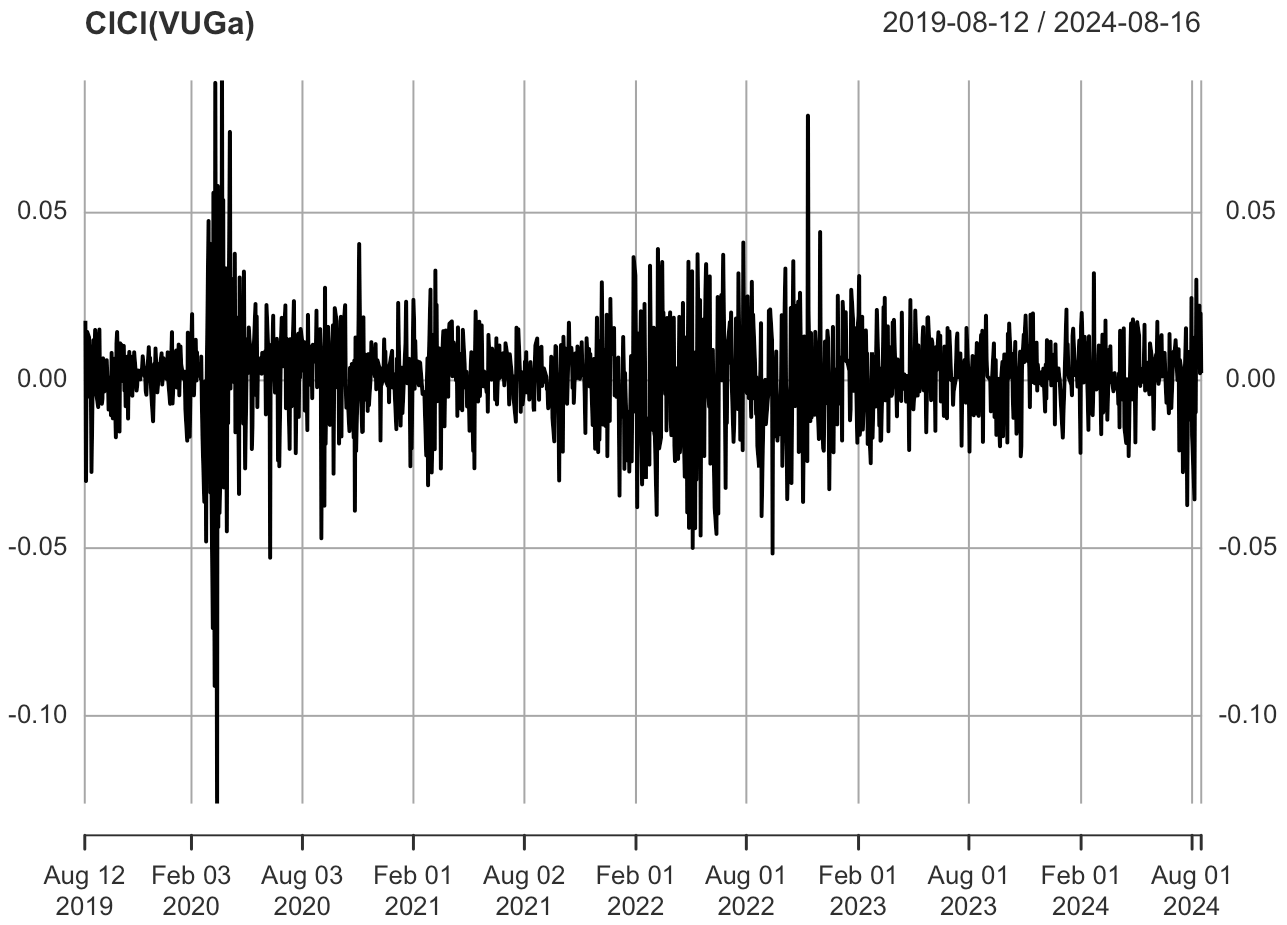
```
#aggressive ETFs

mystocks = c("QQQ", "VUG", "IWF","VGT","XLK")
getSymbols(mystocks, from="2019-08-12")
```

```
## [1] "QQQ" "VUG" "IWF" "VGT" "XLK"
```

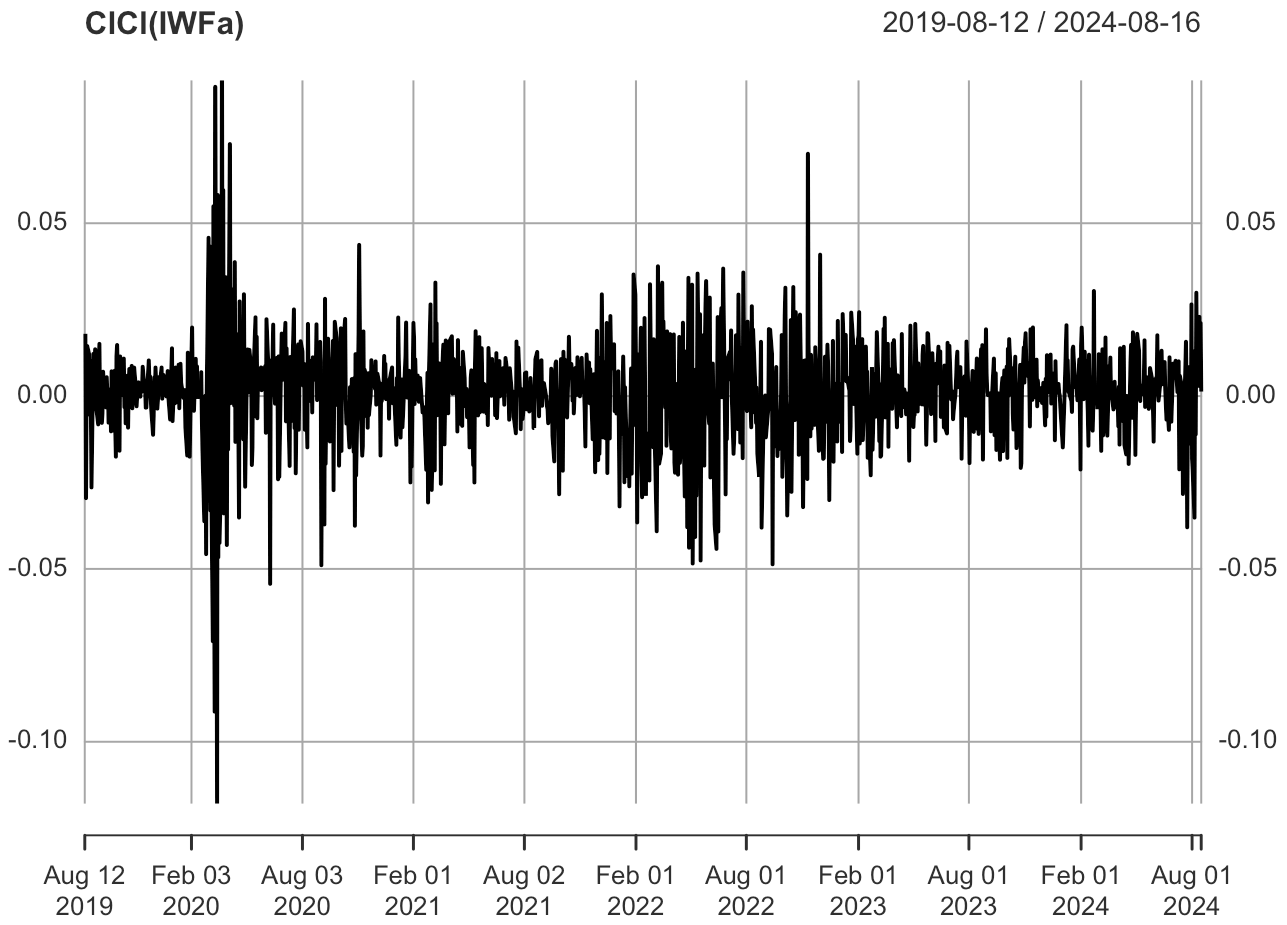
```
QQQa = adjustOHLC(QQQ)
VUGa = adjustOHLC(VUG)
IWFa = adjustOHLC(IWF)
VGTa = adjustOHLC(VGT)
XLKa = adjustOHLC(XLK)


for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
plot(ClCl(QQQa))
```



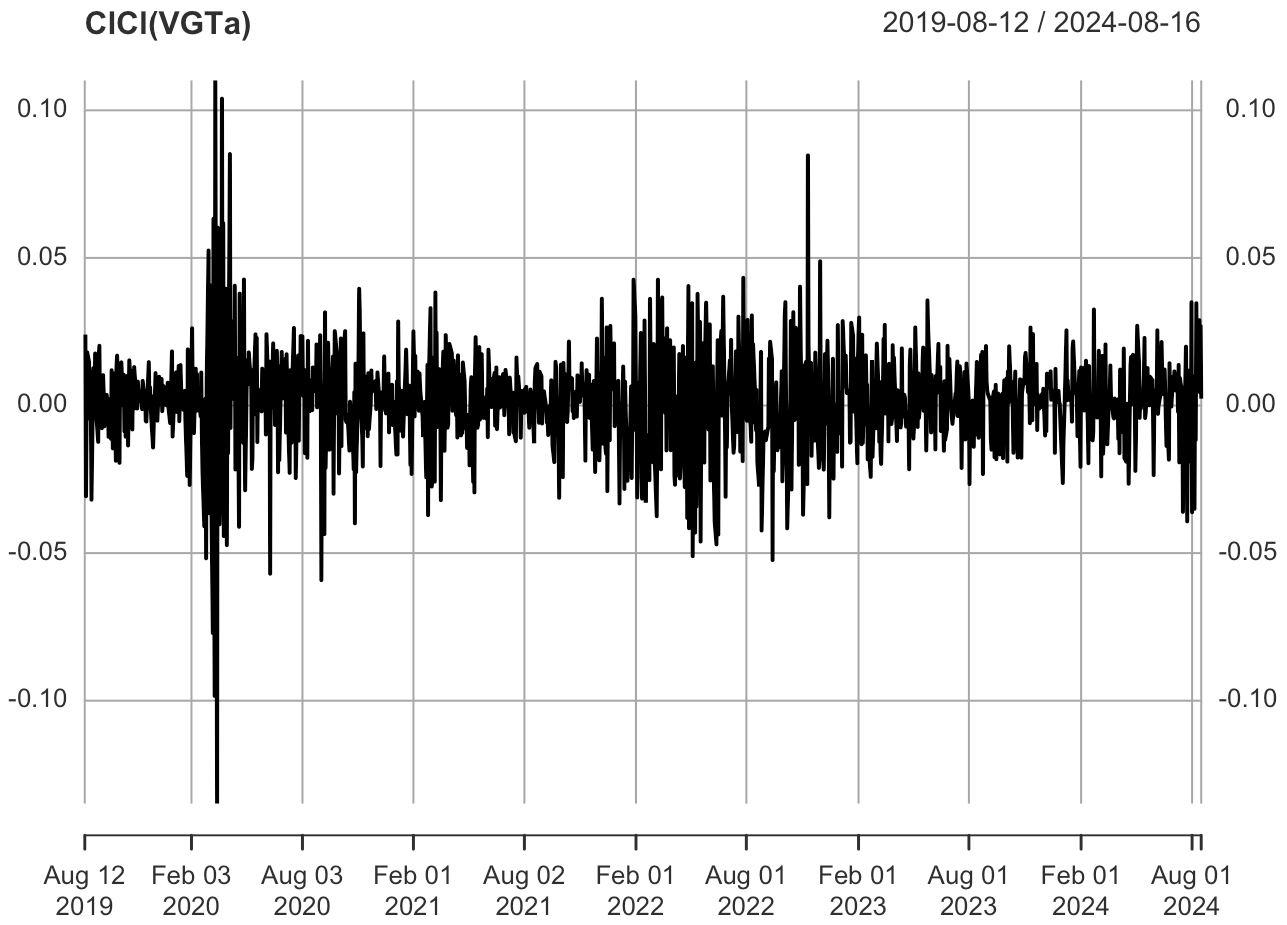

```
plot(ClCl(VUGa))
```



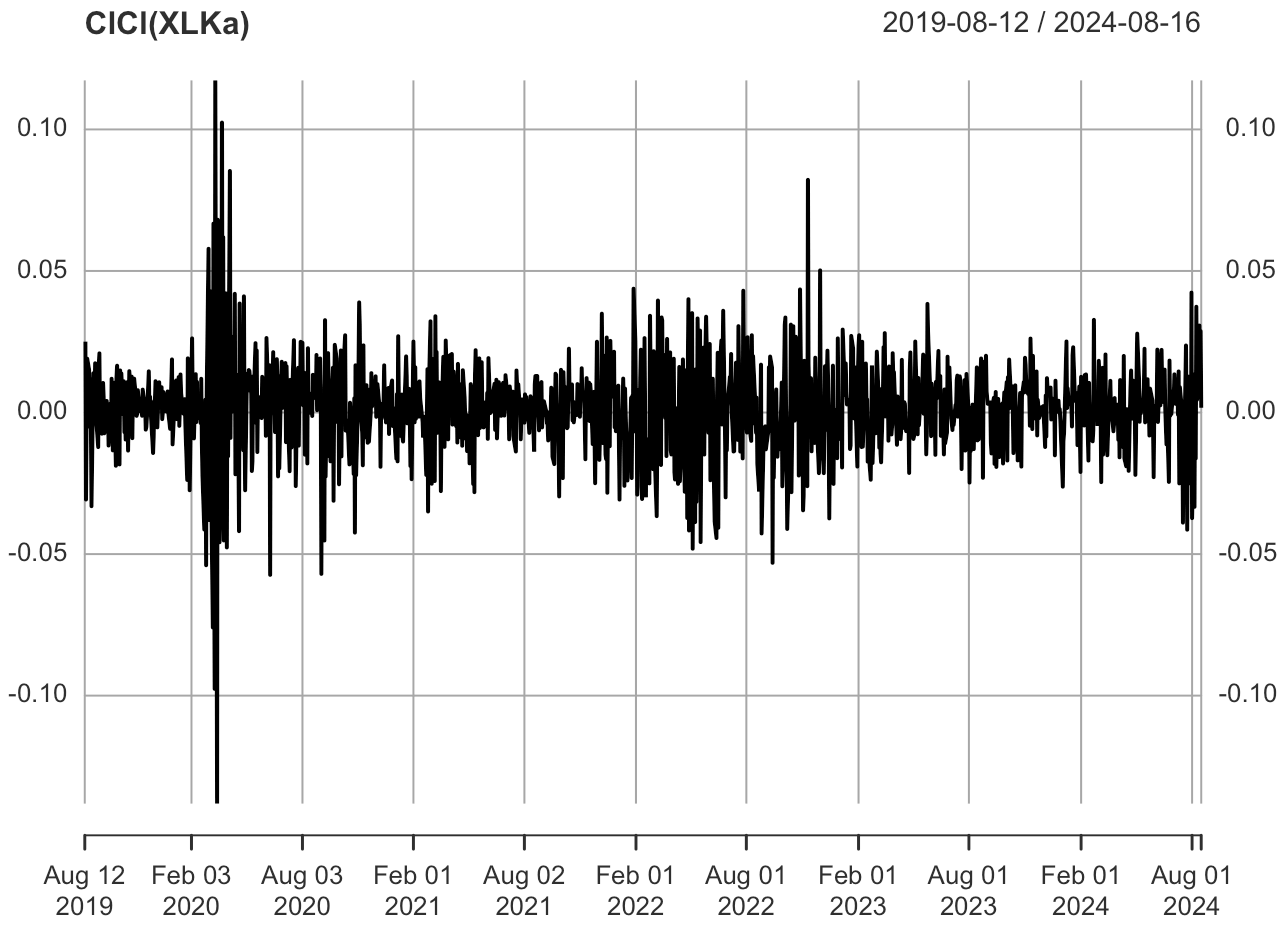

```
plot(ClCl(IWFa))
```



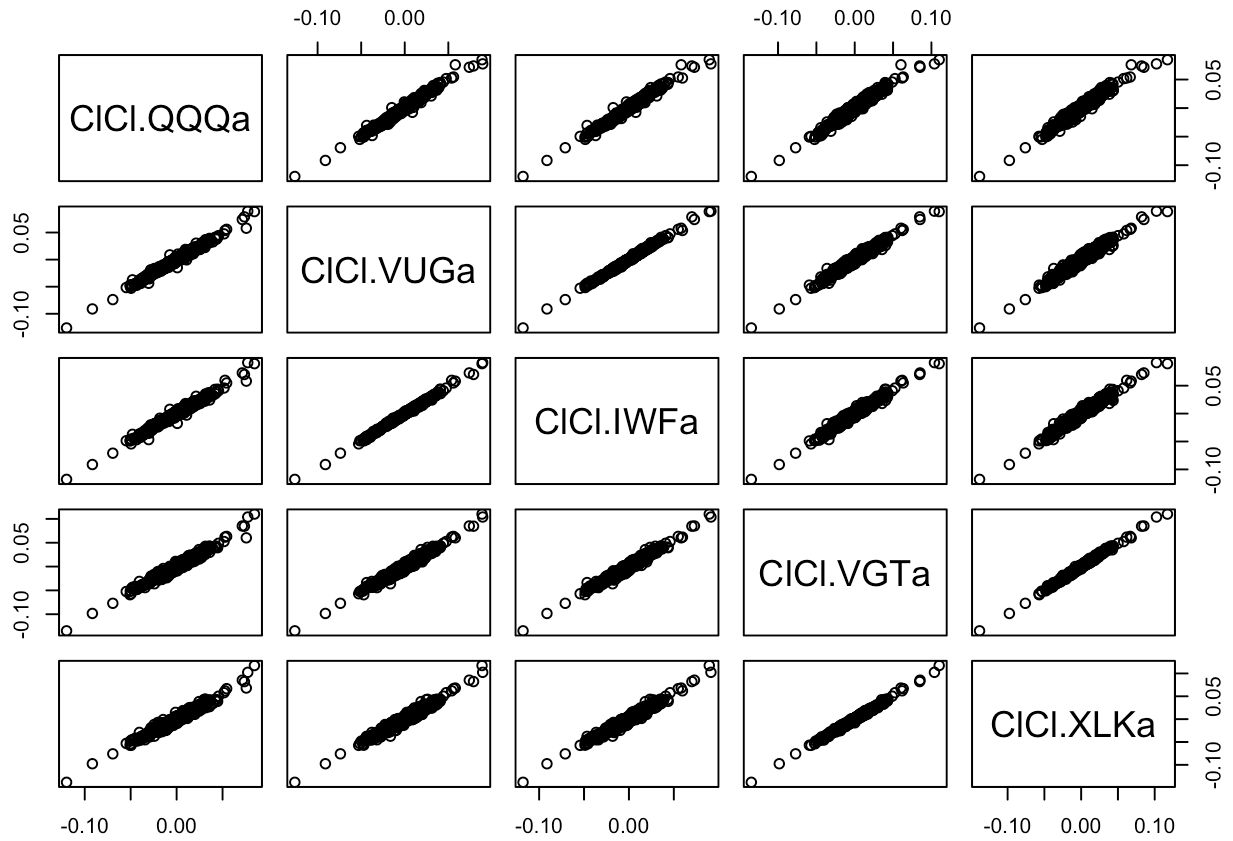

```
plot(ClCl(VGTa))
```




```
plot(ClCl(XLKa))
```



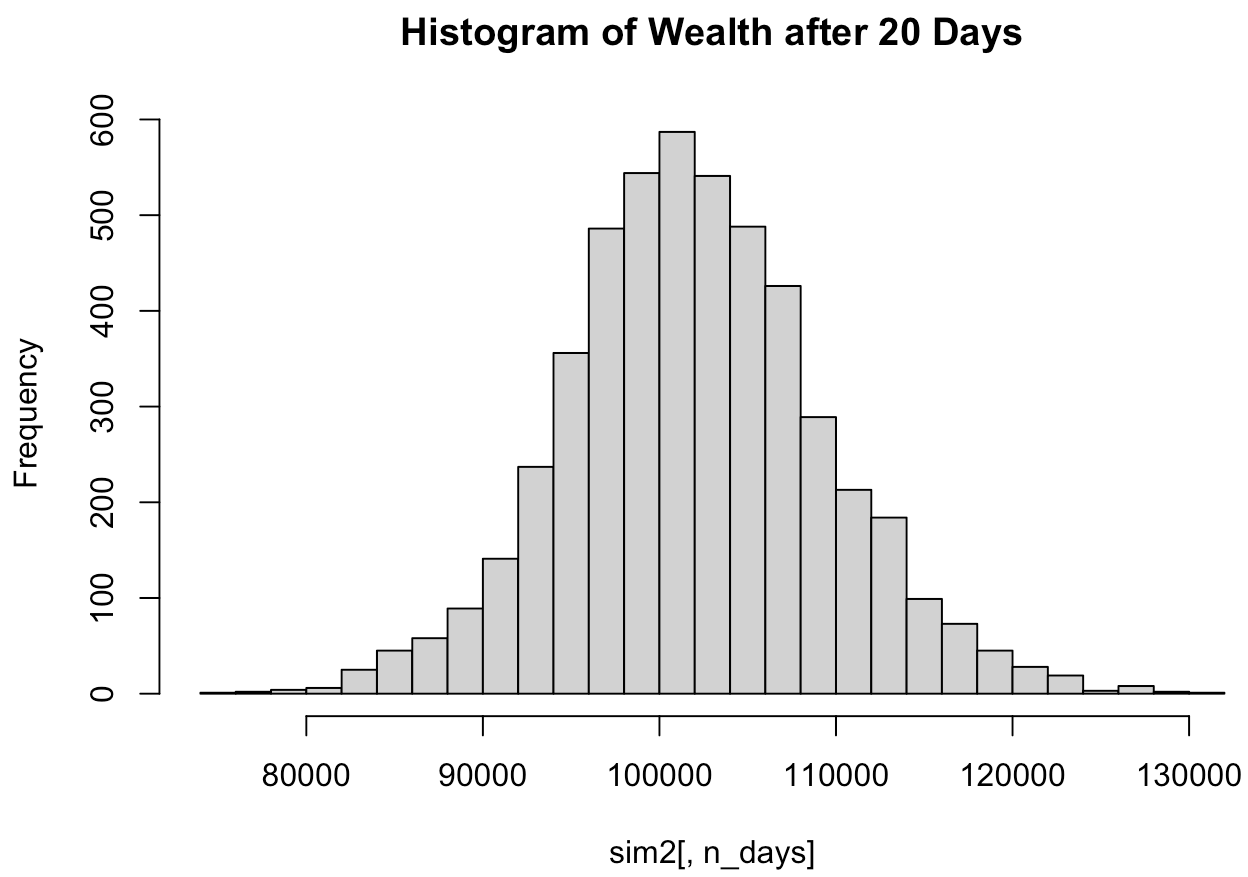

```
all_returns = cbind(ClCl(QQQa),ClCl(VUGa),ClCl(IWFa),ClCl(VGTa), ClCl(XLKa))
all_returns = as.matrix(na.omit(all_returns))

pairs(all_returns)
```




```
#simulates a random day
return.today = resample(all_returns, 1, orig.ids=FALSE)

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

initial_wealth = 100000
sim2 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.2, 0.2, 0.2, 0.2, 0.2)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}

head(sim2)
```

```
##              [,1]      [,2]     [,3]      [,4]      [,5]      [,6]      [,7]
## result.1 101848.38 102199.31 102426.4 104714.67 103766.02 104784.35 103238.57
## result.2  99856.36  99896.54 100573.1 101385.54 101123.70 101275.63 101547.66
## result.3  97051.50  96050.83  96680.5  97251.42  97842.40  98478.27  98270.13
## result.4 100110.32 101188.95 101832.0 101858.47 104106.81 101750.11 101120.99
## result.5 104595.72 101134.80 101691.3 101185.35  99922.05  97282.84  96018.06
## result.6 102592.51 101467.36 101502.2 100502.48 101613.01 103758.27 102693.81
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1 101765.81 103559.89 102465.91 102147.85 102912.11 103164.87 105395.28
## result.2 101572.13 103810.30 102805.23 103389.89 101604.34 101754.41 103291.25
## result.3  99545.85 101050.02 103349.64 103215.11 104660.08 104081.97 102269.91
## result.4 101568.56 101475.48 100430.42 100577.15  98594.22  98740.31  98557.50
## result.5  96086.90  95605.72  95894.42  93910.50  91846.06  90883.11  92556.77
## result.6 100816.74  96115.90  97626.67  97344.09  96558.57  98339.36 100156.98
##             [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1 106937.01 108864.80 108146.89 109091.82 110681.82 110581.27
## result.2 103776.41 106378.06 104207.48 104631.61 103786.23 104318.47
## result.3 102705.04 103514.78 102877.56 103286.82 104021.21 105111.19
## result.4 100309.73 100970.26 102830.22 102752.95 101493.50 100574.62
## result.5  91838.92  92661.69  91299.32  92636.26  93389.74  92732.28
## result.6  99540.42  99958.04 103200.37 101478.97  99392.92  97319.47
```

```
hist(sim2[,n_days], 25,
     main = "Histogram of Wealth after 20 Days")
```



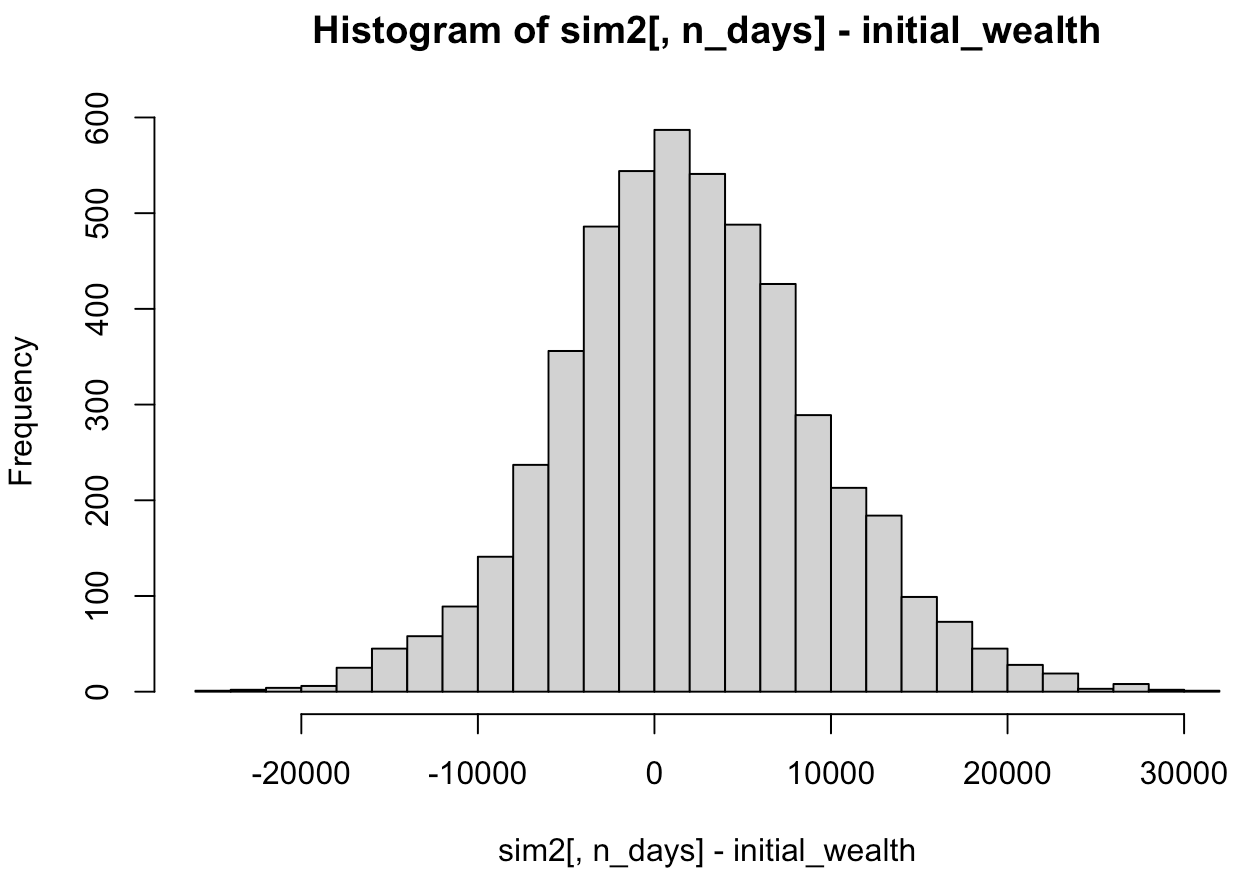
```
# final welath
mean(sim2[,n_days])
```

```
## [1] 102010.8
```

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

```
## [1] 2010.849
```

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



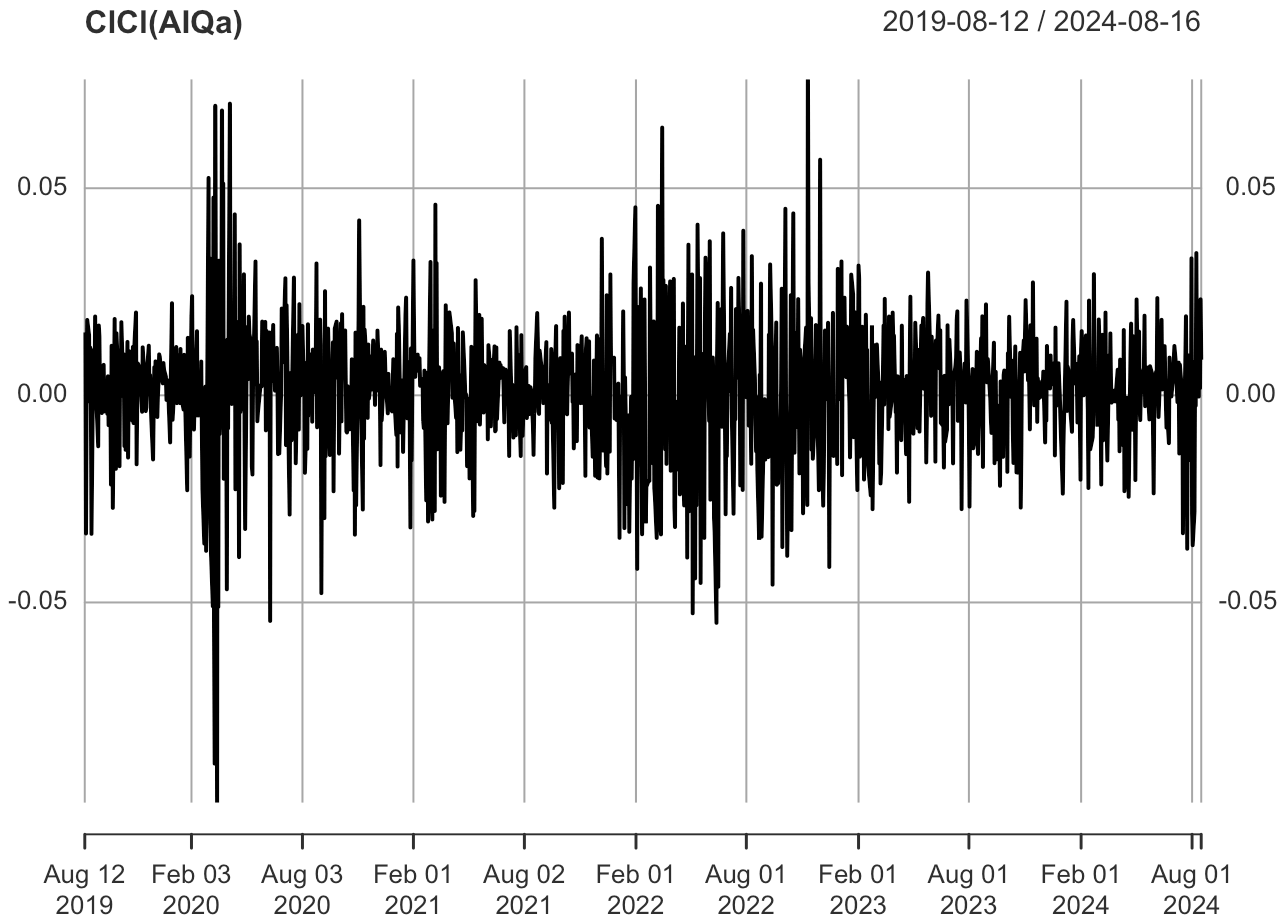

```
#5% VaR
quantile(sim2[,n_days]- initial_wealth, prob=0.05)
```

```
##        5%
## -9578.273
```

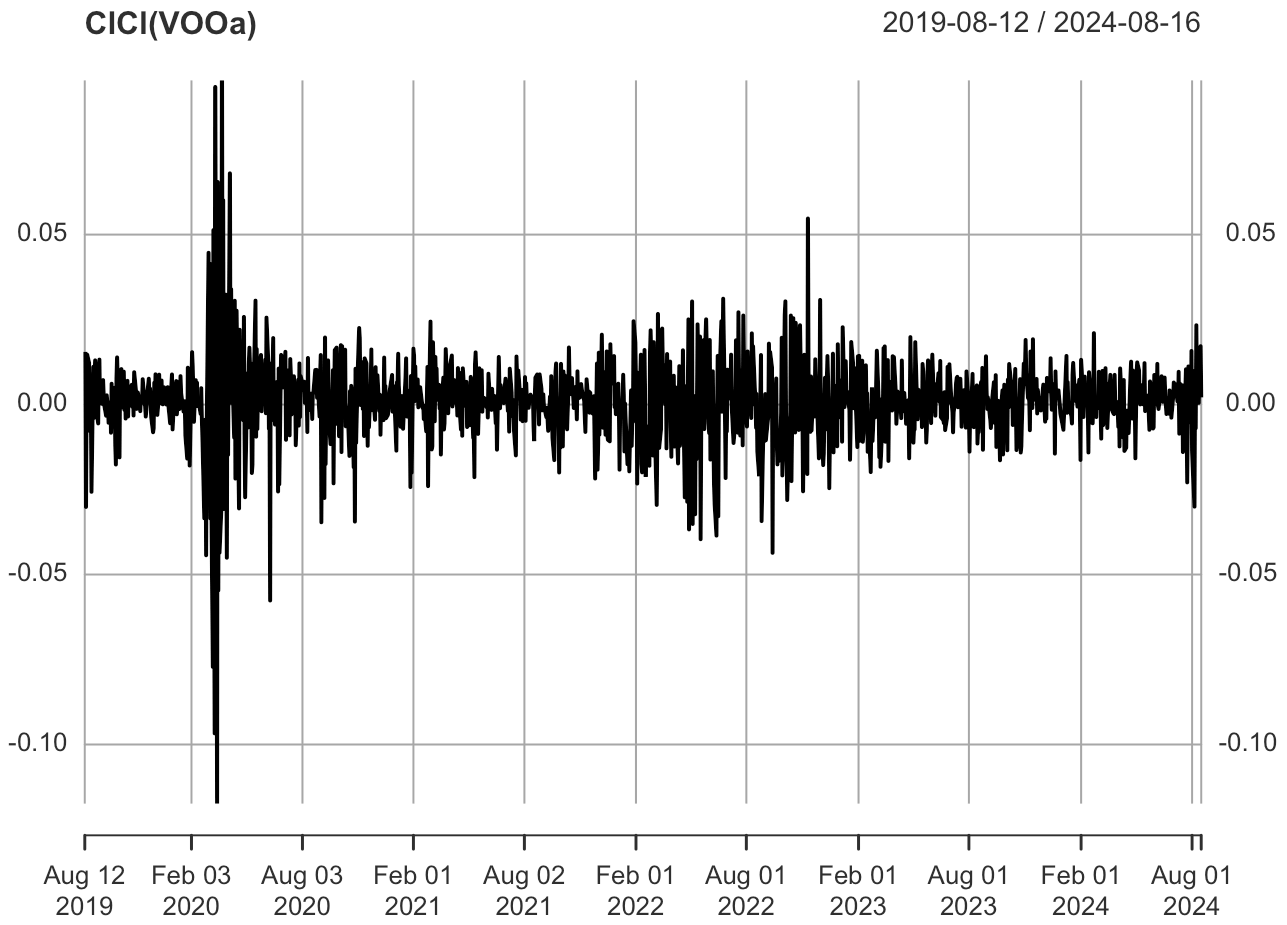
```
#safe and long run ETF


mystocks = c("AIQ", "VOO", "VIG","XLV")
getSymbols(mystocks, from="2019-08-12")
```

```
## [1] "AIQ" "VOO" "VIG" "XLV"
```

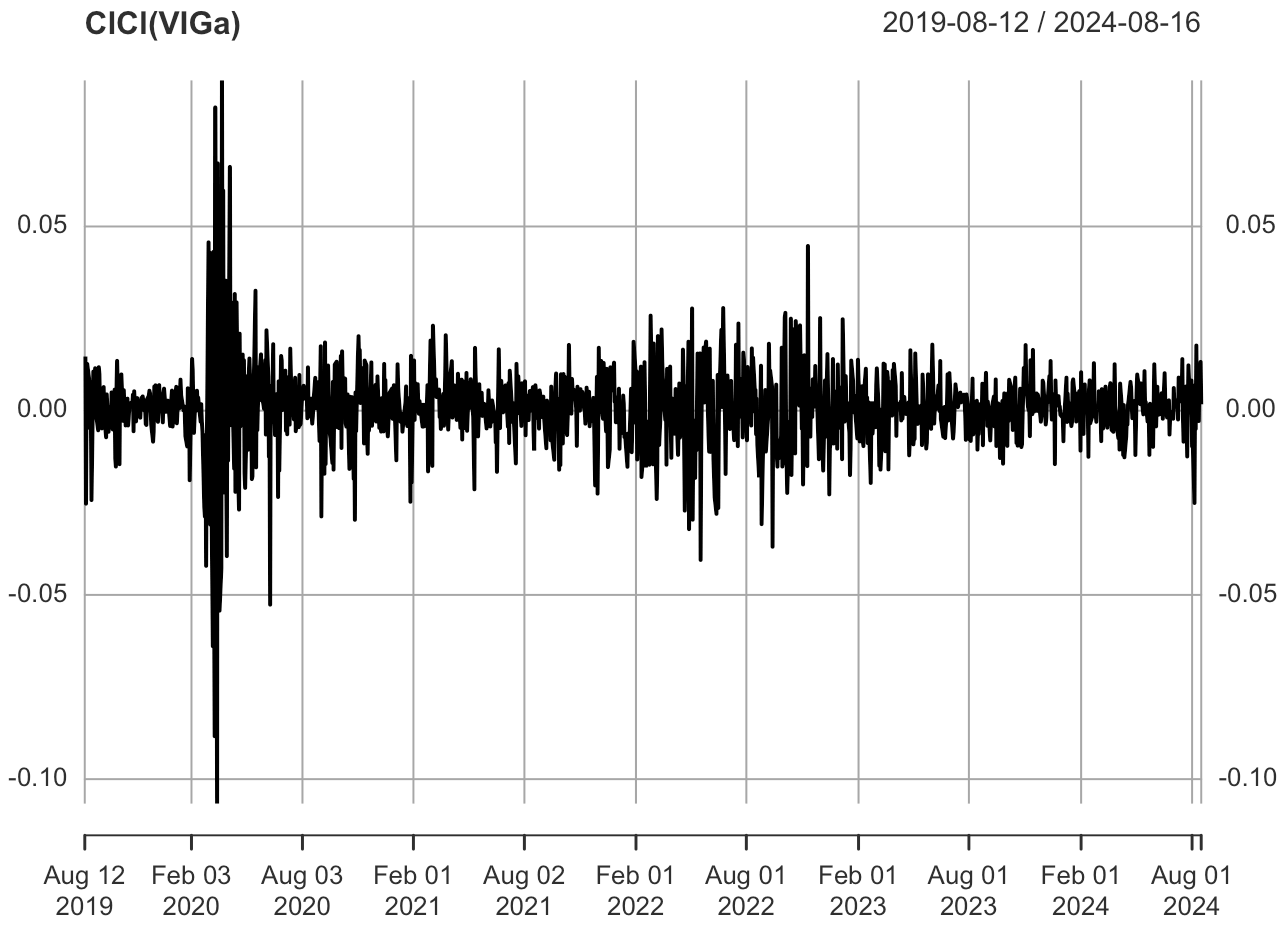
```
AIQa = adjustOHLC(AIQ)
VOOa = adjustOHLC(VOO)
VIGa = adjustOHLC(VIG)
SCHAa = adjustOHLC(XLV)


for(ticker in mystocks) {
    expr = paste0(ticker, "a = adjustOHLC(", ticker, ")")
    eval(parse(text=expr))
}
plot(ClCl(AIQa))
```



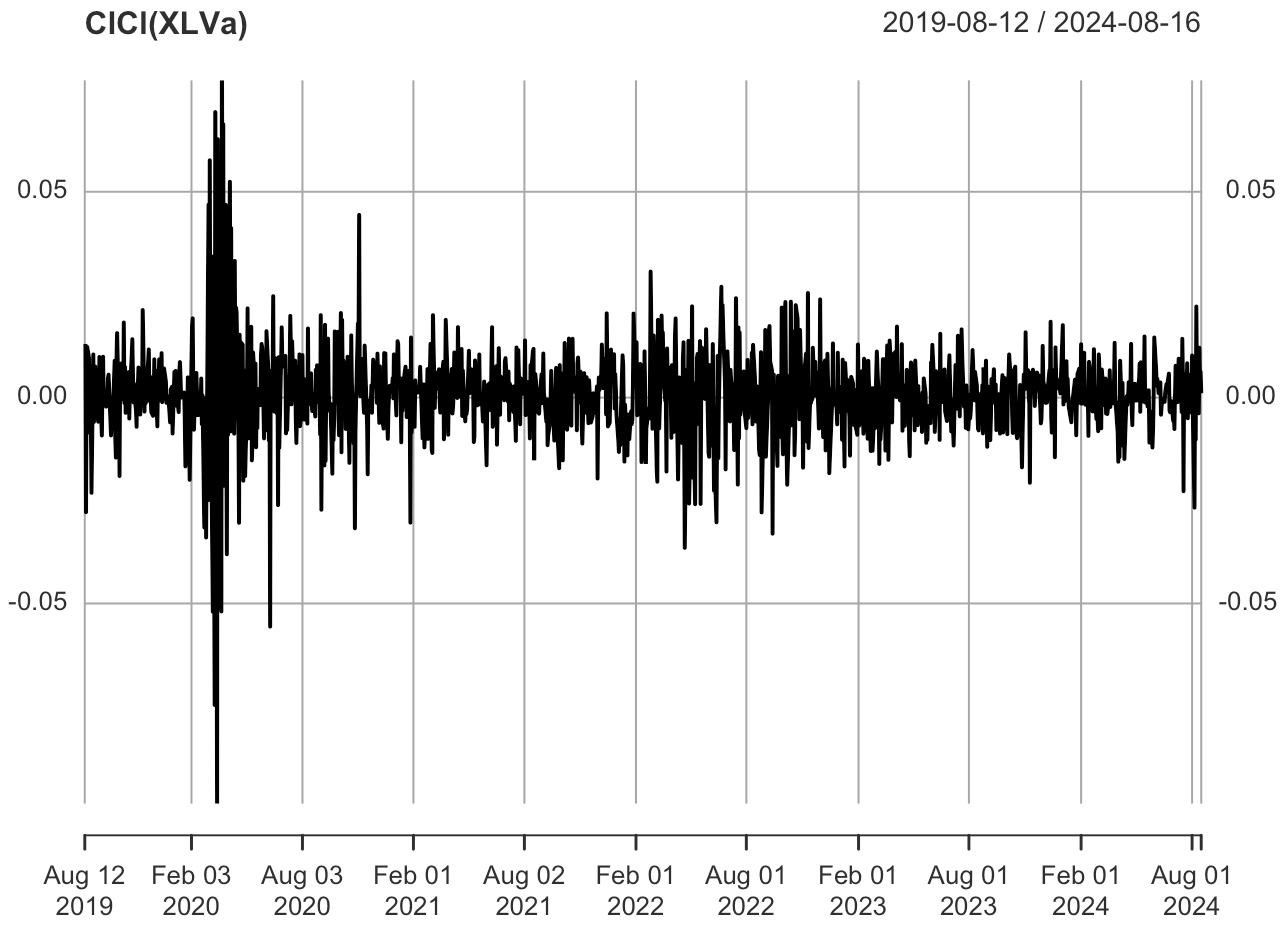

```
plot(ClCl(VOOa))
```



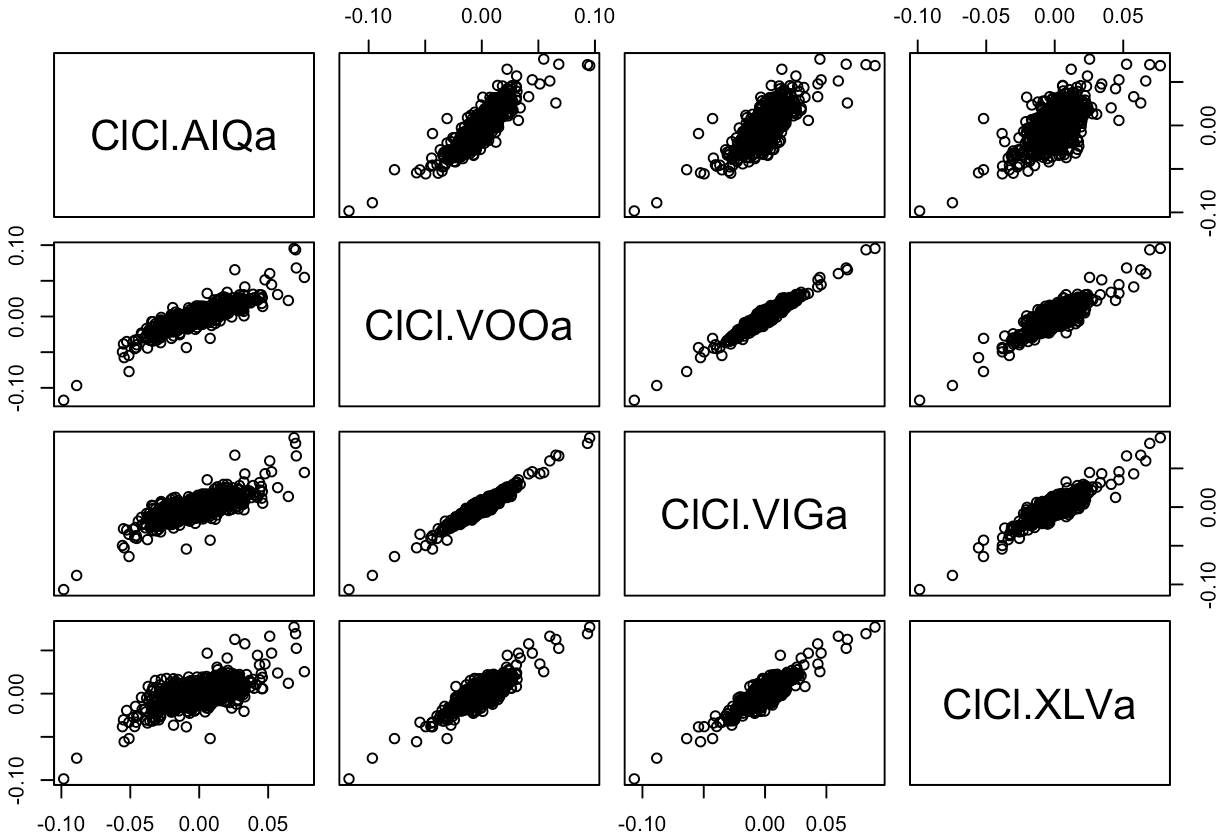

```
plot(ClCl(VIGa))
```




```
plot(ClCl(XLVa))
```



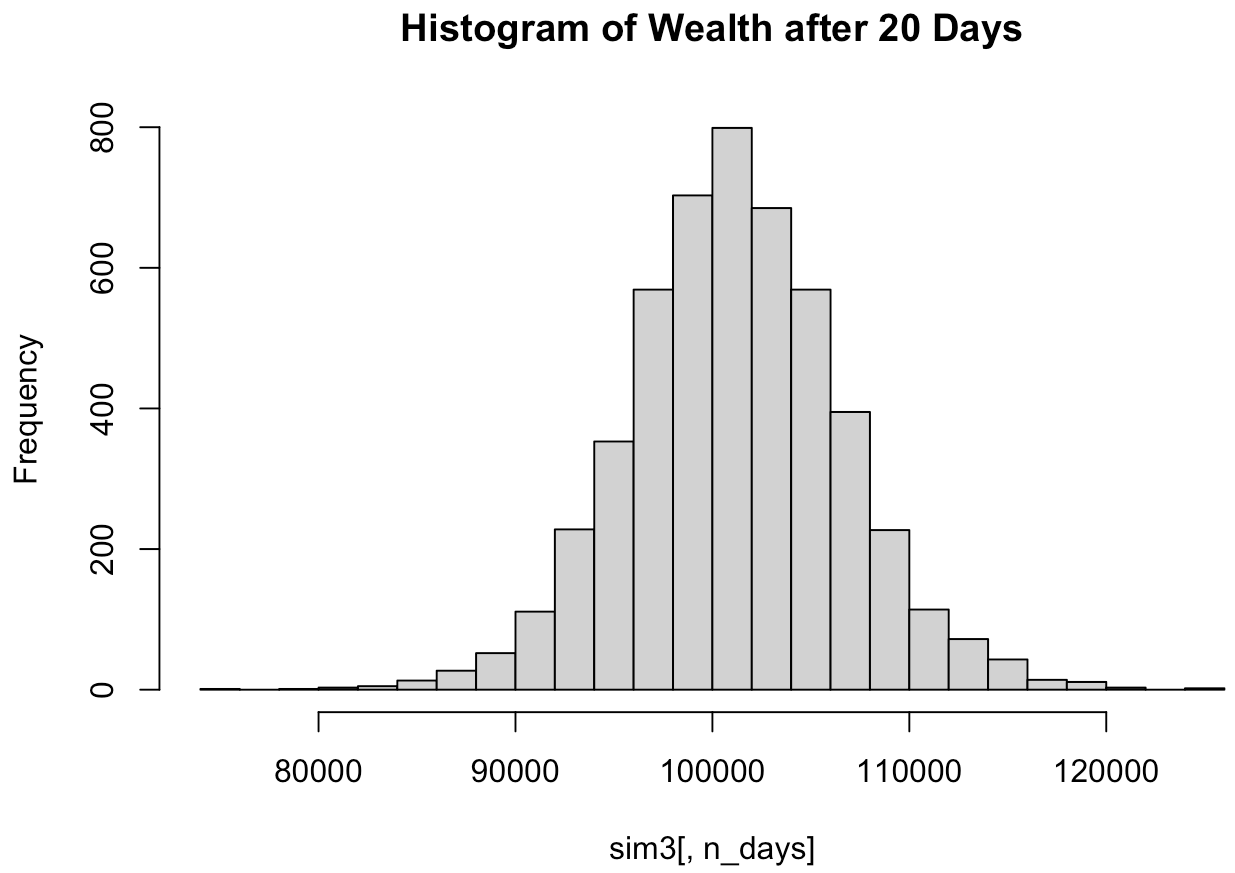

```
all_returns = cbind(ClCl(AIQa),ClCl(VOOa),ClCl(VIGa),ClCl(XLVa))
all_returns = as.matrix(na.omit(all_returns))

pairs(all_returns)
```




```
#simulates a random day
return.today = resample(all_returns, 1, orig.ids=FALSE)

total_wealth = 100000
my_weights = c(0.2, 0.2, 0.3, 0.3)
holdings = total_wealth*my_weights
holdings = holdings*(1 + return.today)

initial_wealth = 100000
sim3 = foreach(i=1:5000, .combine='rbind') %do% {
    total_wealth = initial_wealth
    weights = c(0.2, 0.2, 0.3, 0.3)
    holdings = weights * total_wealth
    n_days = 20
    wealthtracker = rep(0, n_days)
    for(today in 1:n_days) {
        return.today = resample(all_returns, 1, orig.ids=FALSE)
        holdings = holdings + holdings*return.today
        total_wealth = sum(holdings)
        wealthtracker[today] = total_wealth
    }
    wealthtracker
}

head(sim3)
```

```
##              [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## result.1  97809.07  98398.31  99336.57  99650.51  99503.32  96664.78  99103.24
## result.2  99974.35  99978.40  99441.77  99298.61  98565.69  97304.34  98073.58
## result.3  99419.23 100034.49 100026.42 100646.80 102085.91 102076.53 103114.63
## result.4 100791.10 100621.00 101019.22  97956.33  98127.59  98262.51  98654.90
## result.5  99924.95 100079.37  98685.04  98447.03  98198.86 104413.96 104039.77
## result.6 100098.82  99412.52  99417.34 100123.35  99906.85  98517.24  99517.17
##              [,8]      [,9]     [,10]     [,11]     [,12]     [,13]     [,14]
## result.1  98655.11  99084.53  97999.33  97515.36  98066.12  97402.40  97383.13
## result.2  98850.05  97940.65  97274.77  97220.56  97494.82  97277.27  94505.79
## result.3 102907.69 103123.10 104529.18 105464.03 105808.72 105754.26 106351.49
## result.4  97157.15  96777.82  96495.84  95067.37  94717.32  96370.44  96039.71
## result.5 103024.23 104927.58 104787.96 104477.63 104646.70 104189.16 104495.69
## result.6  99962.65 100989.98 100449.38 100099.96 100452.32  99796.07 101260.02
##             [,15]     [,16]     [,17]     [,18]     [,19]     [,20]
## result.1  98650.40  99362.59  96251.73  97594.12  96743.89  97599.80
## result.2  93458.84  93877.59  94209.58  92946.93  92477.91  92009.19
## result.3 106628.60 106549.03 107263.24 106535.90 107460.56 107971.69
## result.4  95604.42  96942.85  97859.20  98514.59  98304.12  99391.63
## result.5 105795.15 108142.06 108482.67 108145.67 107733.36 109855.64
## result.6 101634.21 100480.85 100179.70 101296.94 101990.01 102932.26
```

```
hist(sim3[,n_days], 25,
     main = "Histogram of Wealth after 20 Days")
```



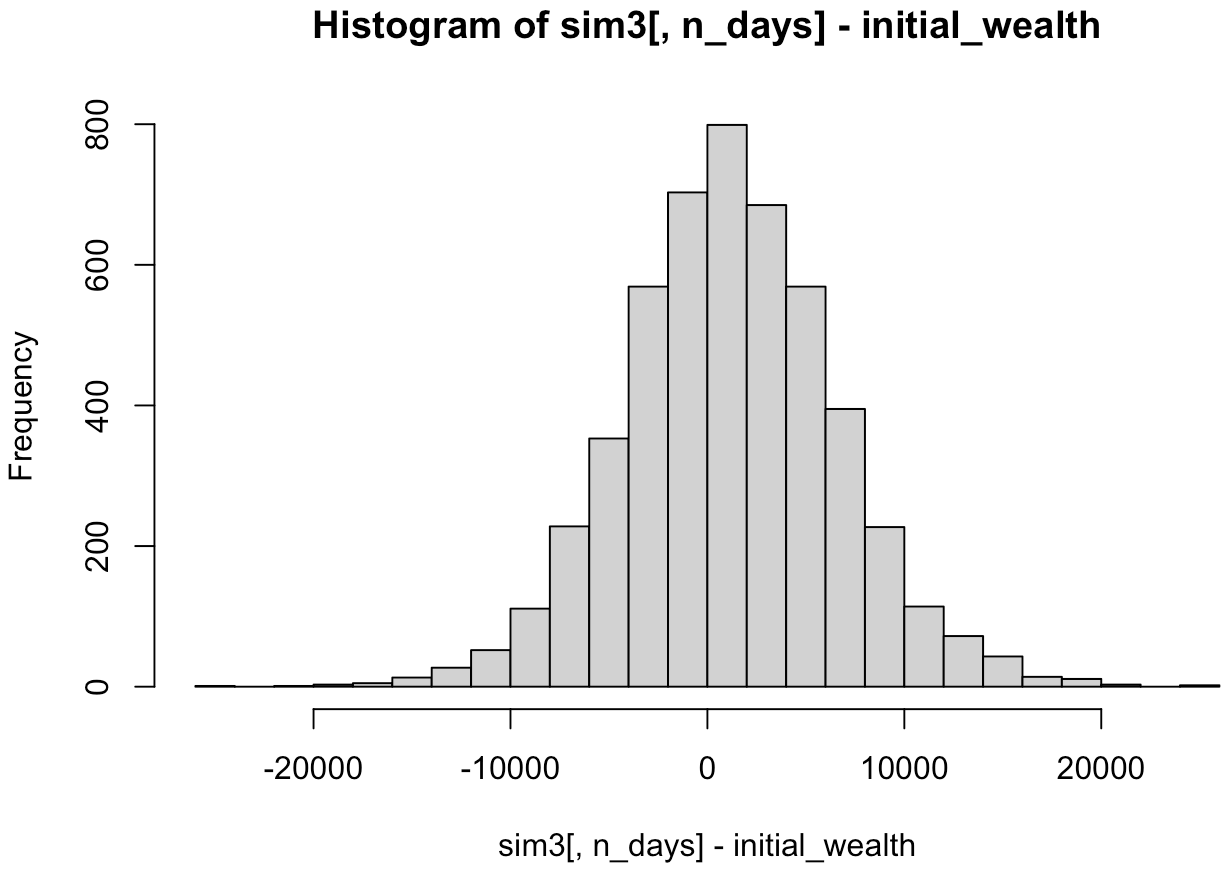
```
# final wealth
mean(sim3[,n_days])
```

```
## [1] 101162.5
```

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

```
## [1] 1162.493
```

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

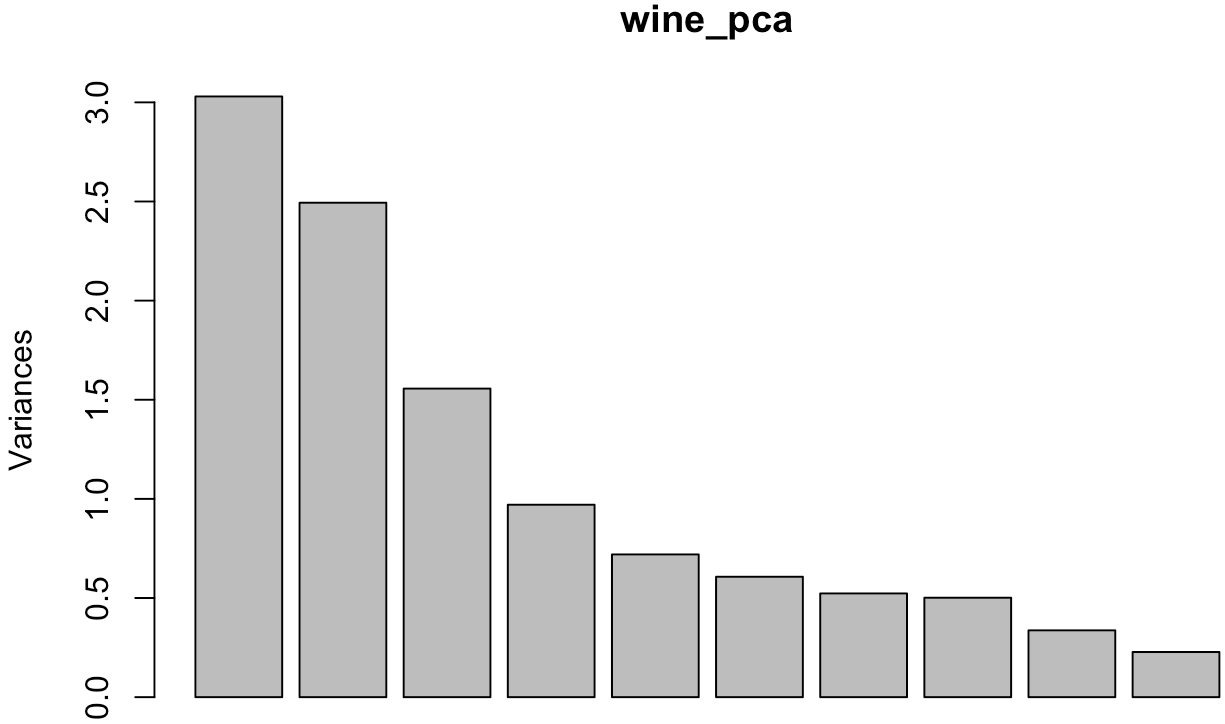



```
#5% VaR
quantile(sim3[,n_days]- initial_wealth, prob=0.05)
```

```
##        5%
## -7552.535
```

Clustering and dimensionality reduction

```
library(Rtsne)
wine=read.csv("/Users/sarahlee/Downloads/wine.csv")

#PCA
winescaled <- scale(wine[, 1:11],center=TRUE, scale=TRUE)
wine_pca=prcomp(winescaled,center=TRUE,scale. = TRUE)
plot(wine_pca)
```



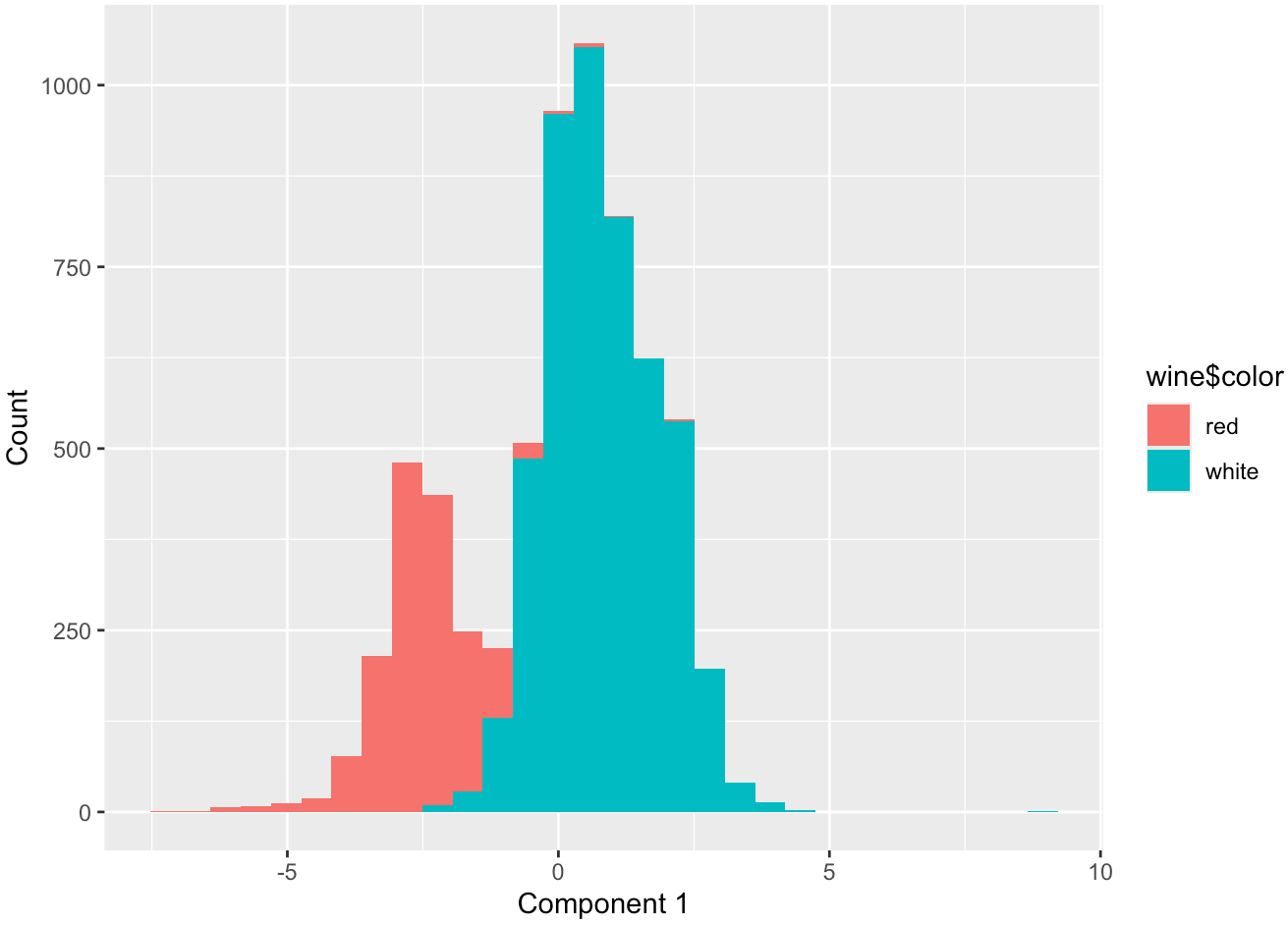

```
summary(wine_pca)
```

```
## Importance of components:
##                           PC1    PC2    PC3     PC4     PC5     PC6     PC7
## Standard deviation     1.7407 1.5792 1.2475 0.98517 0.84845 0.77930 0.72330
## Proportion of Variance 0.2754 0.2267 0.1415 0.08823 0.06544 0.05521 0.04756
## Cumulative Proportion  0.2754 0.5021 0.6436 0.73187 0.79732 0.85253 0.90009
##                            PC8     PC9   PC10    PC11
## Standard deviation     0.70817 0.58054 0.4772 0.18119
## Proportion of Variance 0.04559 0.03064 0.0207 0.00298
## Cumulative Proportion  0.94568 0.97632 0.9970 1.00000
```

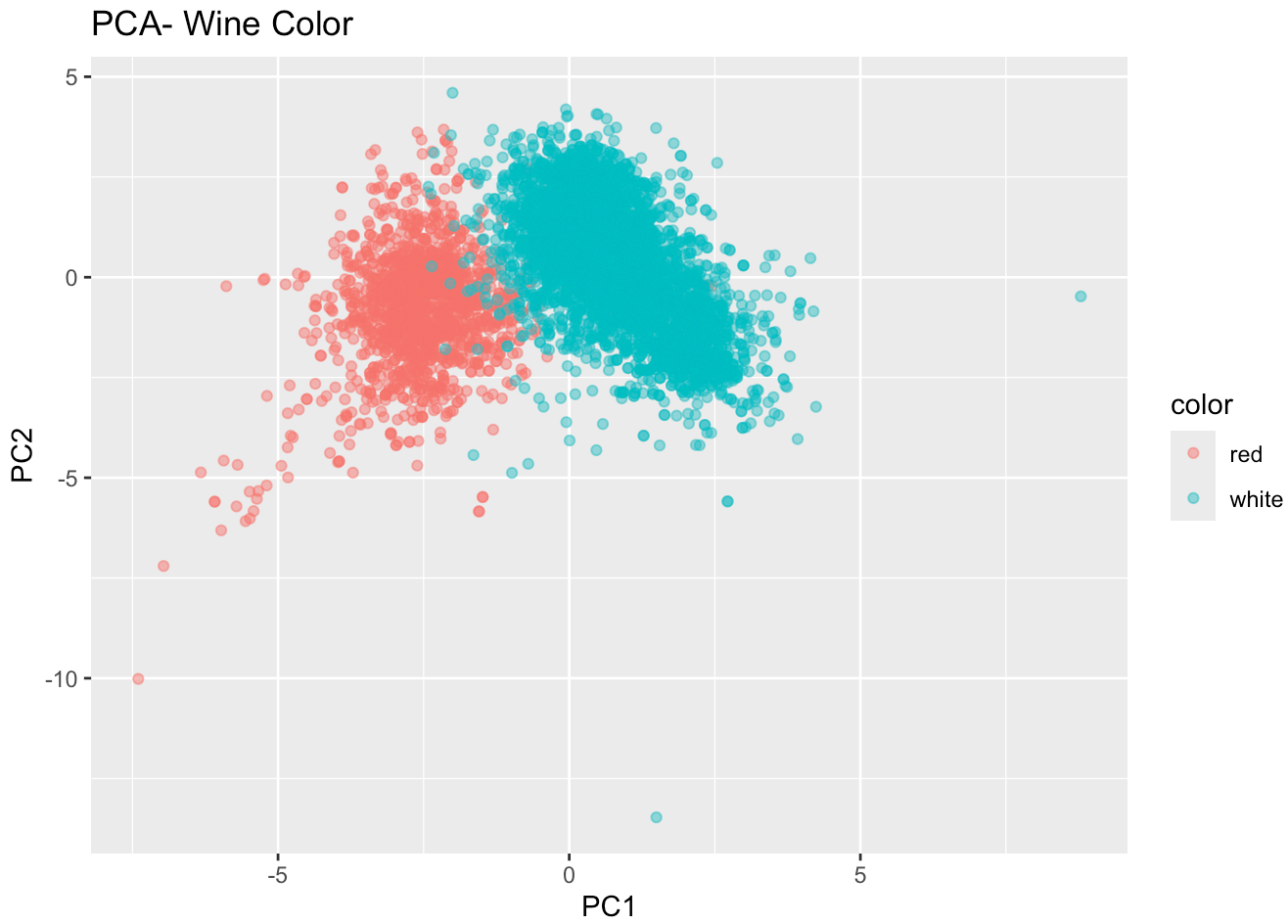
```
loadings = wine_pca$rotation
scores = wine_pca$x

qplot(scores[,1], fill=wine$color, xlab='Component 1', ylab='Count')
```

```
## Warning: `qplot()` was deprecated in ggplot2 3.4.0.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

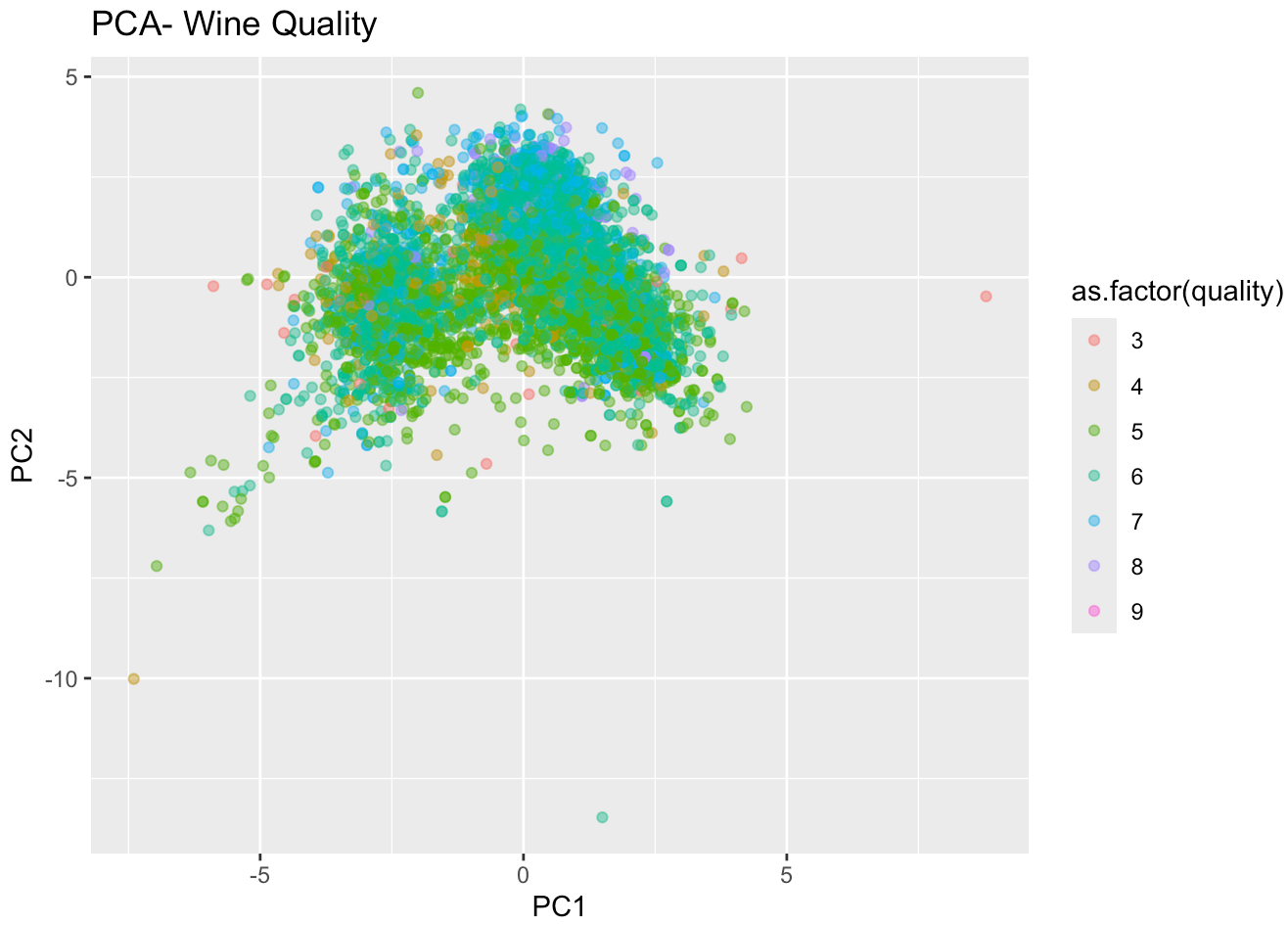
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
pca_df <- data.frame(wine_pca$x, wine[, 12:13]) #wine color and quality
pca_df$color <- factor(wine$color)

ggplot(pca_df, aes(x=PC1,y=PC2, color=color))+
  geom_point(alpha=0.5)+
  labs(title = "PCA- Wine Color")
```

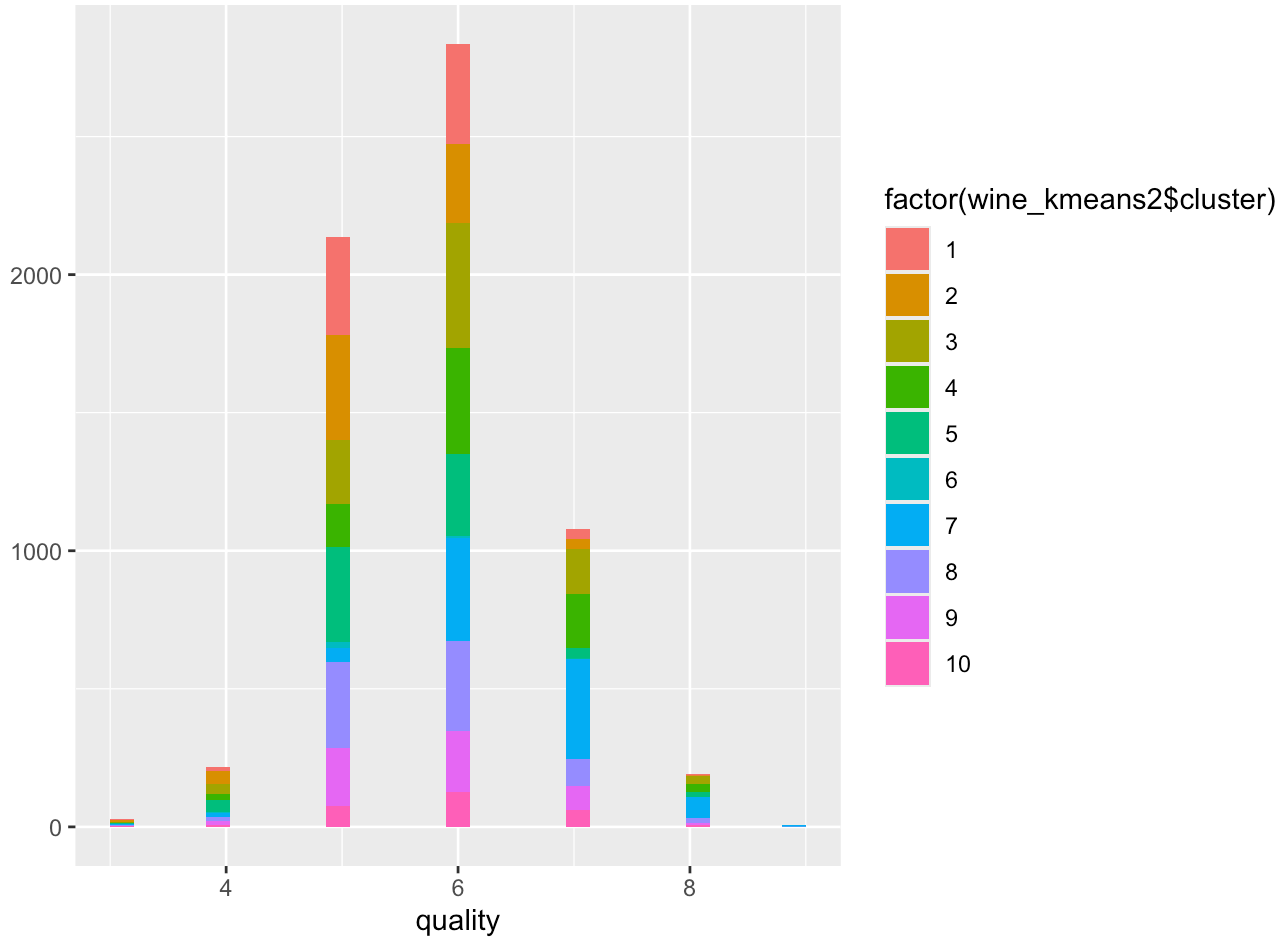



```
ggplot(pca_df, aes(x=PC1,y=PC2, color=as.factor(quality)))+
  geom_point(alpha=0.5)+
  labs(title = "PCA- Wine Quality")
```




```
#KMEANS
set.seed(1)
wine_kmeans=kmeans(winescaled,centers=2,nstart=25)
kmeans_df=data.frame(winescaled, cluster = factor(wine_kmeans$cluster), wine_color = fac
tor(wine$color),wine_quality = factor(wine$quality))

# Apply K-Means clustering with 10 clusters (for different wine qualities)
wine_kmeans2 = kmeans(winescaled, 10, nstart=50)
# Plot wine quality, filled by K-Means cluster assignments (10 clusters)
qplot(quality, fill=factor(wine_kmeans2$cluster), data = wine)
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
table(wine$color, wine_kmeans$cluster) # Compare K-Means clusters with wine color
```

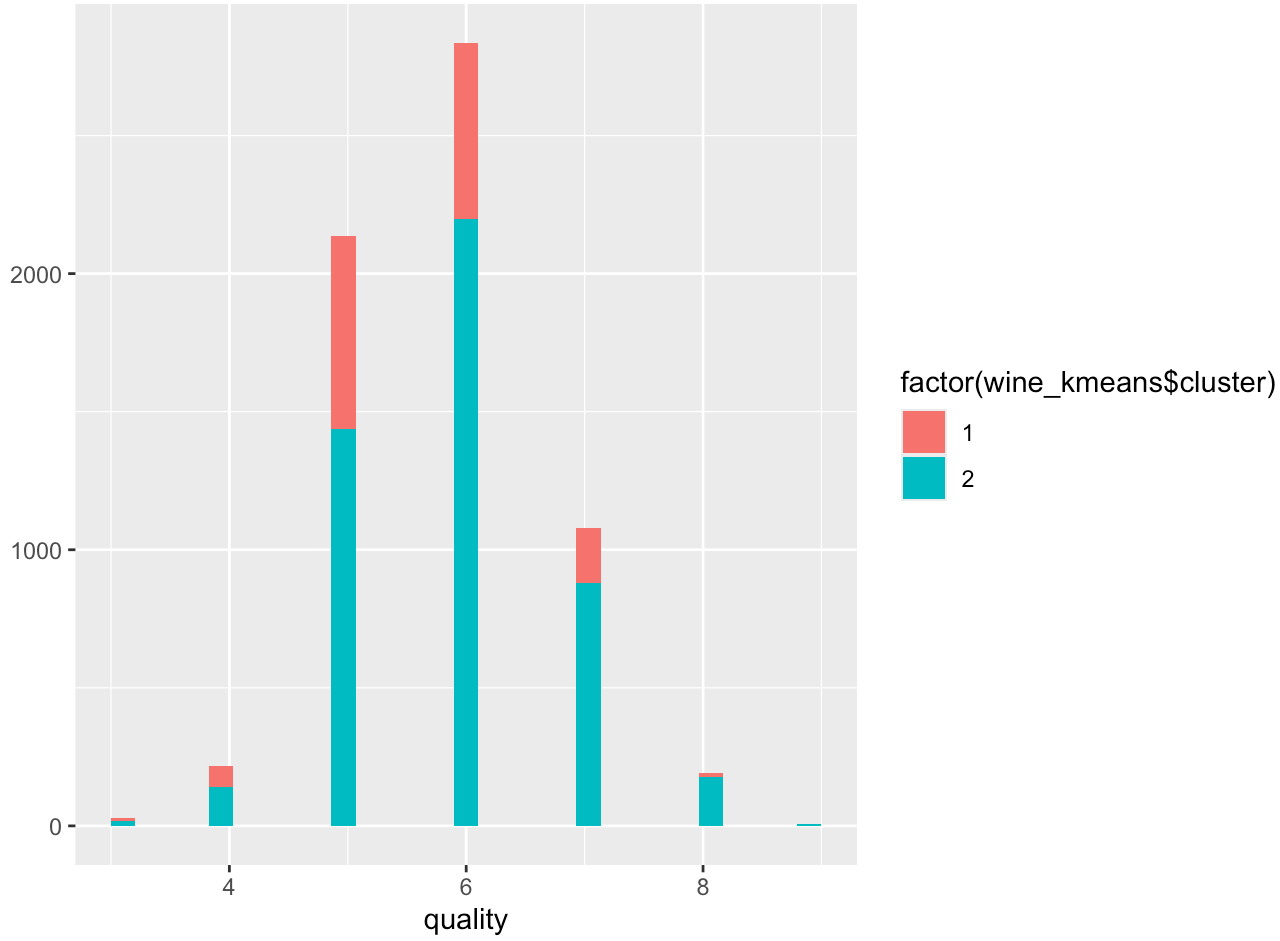
```
##
##            1    2
##   red   1575   24
##   white   68 4830
```

```
table(wine$quality, wine_kmeans$cluster) # Compare K-Means clusters with wine quality
```

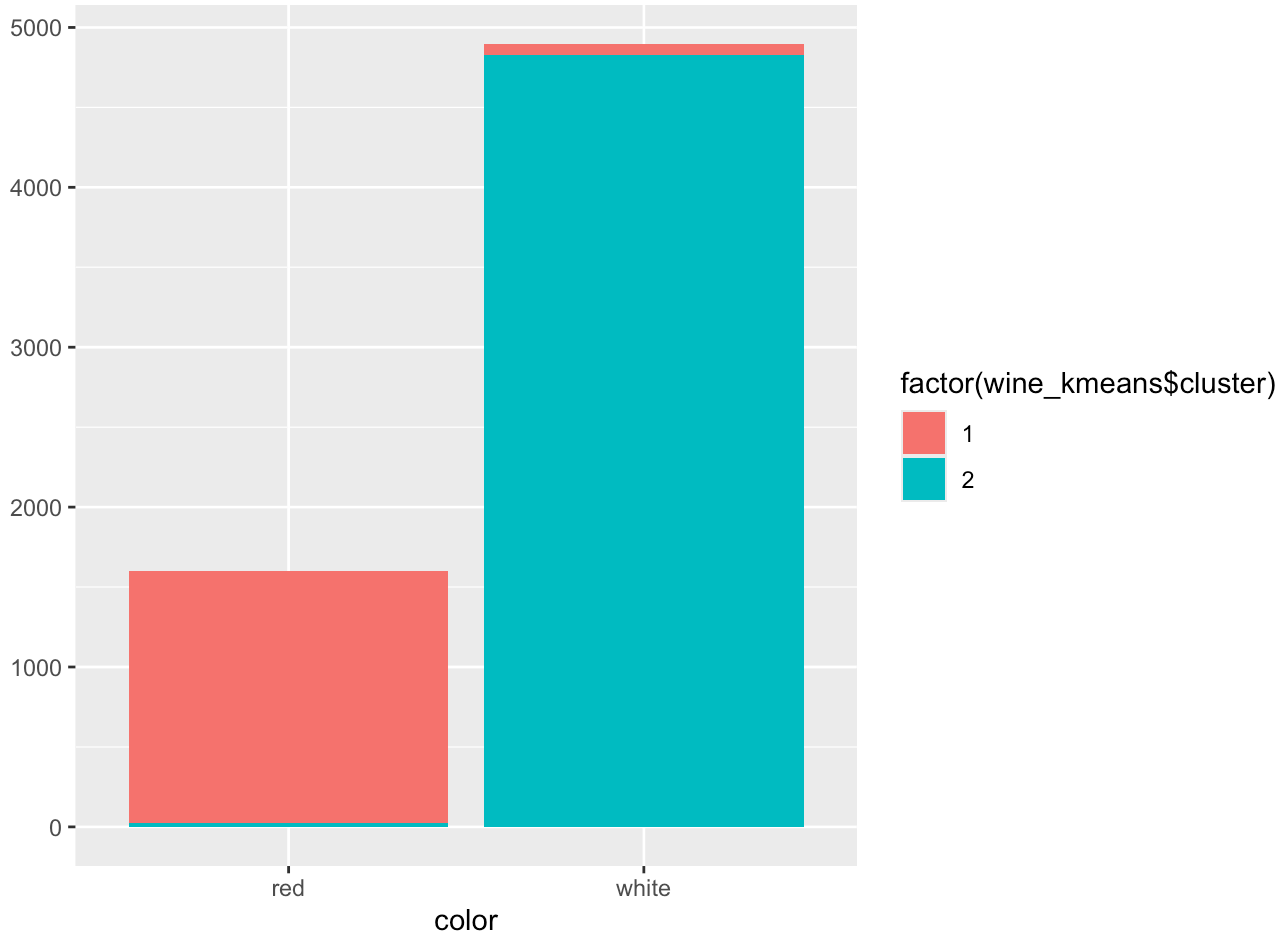
```
##
##        1    2
##   3   12   18
##   4   74  142
##   5  702 1436
##   6  640 2196
##   7  198  881
##   8   17  176
##   9    0    5
```

```
qplot(quality, fill=factor(wine_kmeans$cluster), data = wine) #Plot wine quality, filled
by K-Means cluster
```

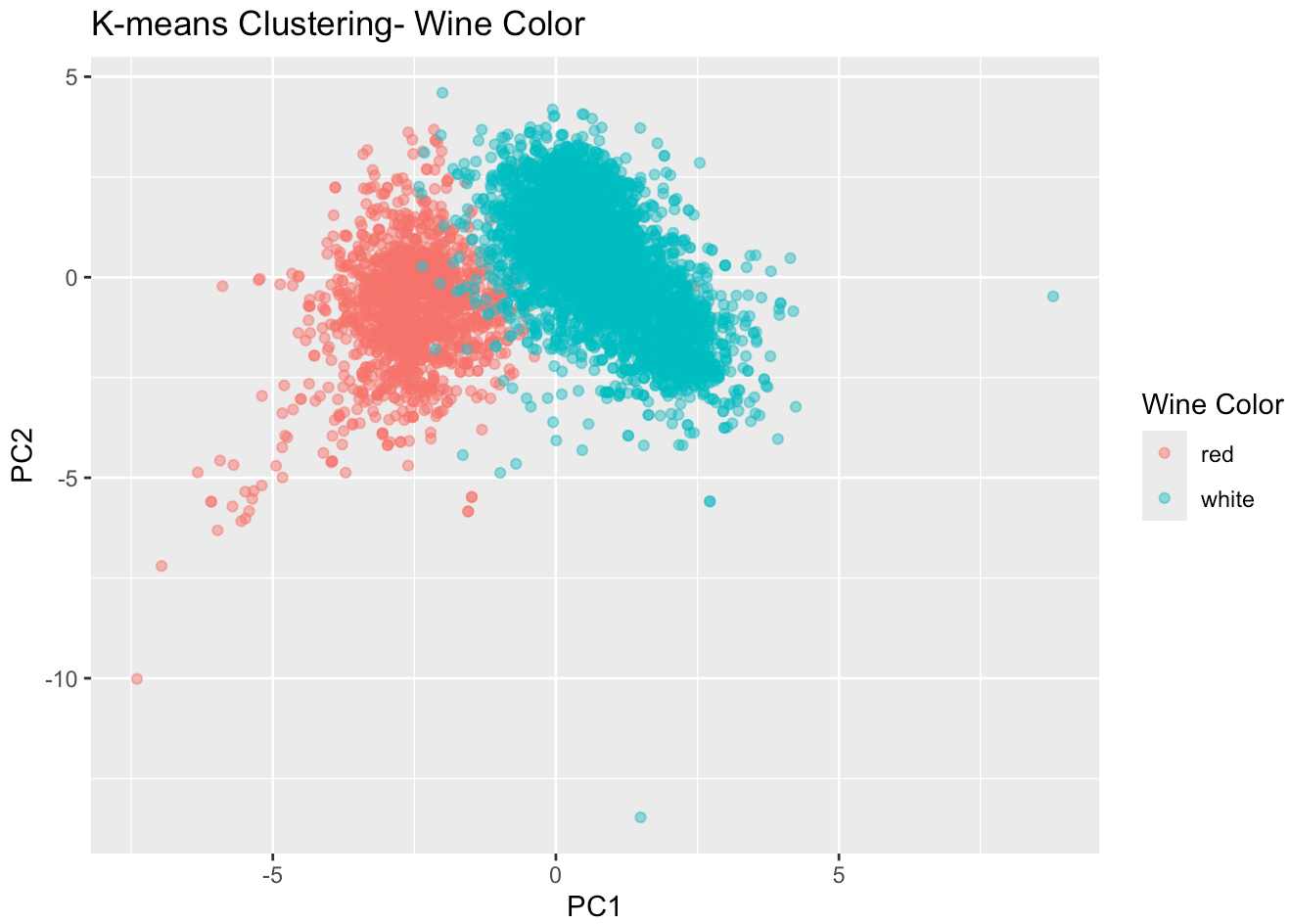
```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```




```
qplot(color, fill=factor(wine_kmeans$cluster), data = wine) #Plot wine color, filled by K-Means cluster
```

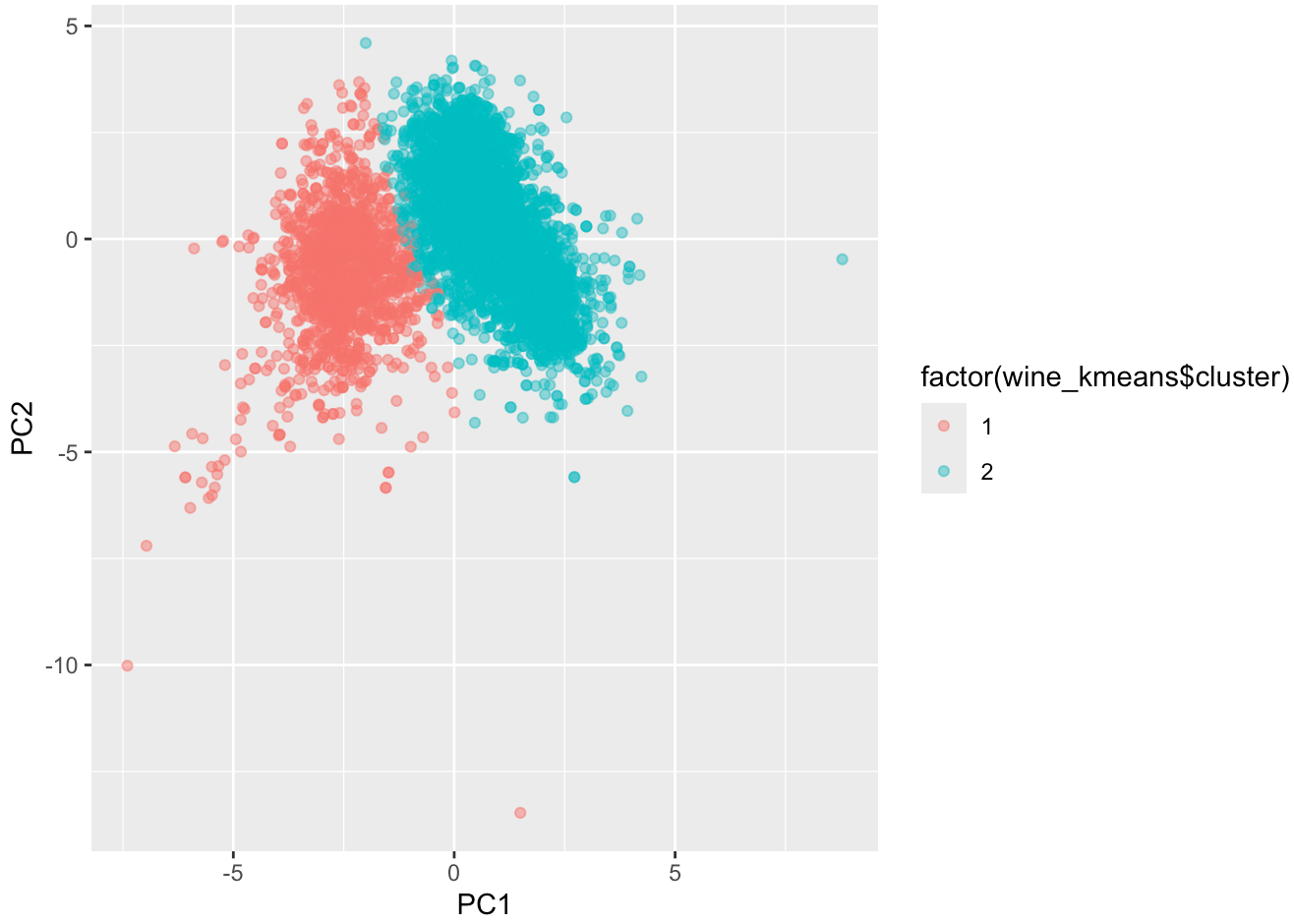



```
ggplot(kmeans_df, aes(x = wine_pca$x[,1], y = wine_pca$x[,2], color = wine_color)) +
  geom_point(alpha = 0.5) +
  labs(title = "K-means Clustering- Wine Color", x = "PC1", y = "PC2", color = "Wine Col
or")
```




```
ggplot(kmeans_df, aes(x = wine_pca$x[,1], y = wine_pca$x[,2], color = wine_quality)) +
  geom_point(alpha = 0.5) +
  labs(title = "K-means Clustering- Wine Quality", x = "PC1", y = "PC2", color = "Wine Q
uality")
```


```
#K-Means cluster assignments on the first two principal components
ggplot(pca_df, aes(x=PC1,y=PC2,color=factor(wine_kmeans$cluster)))+
  geom_point(alpha=0.5) #best distinguishes the two wine colors (very few overlaps)
```




```
#tSNE
set.seed(1)
winescaled_unique <- winescaled[!duplicated(winescaled), ]
tsne_results <- Rtsne(winescaled_unique, dims = 2, perplexity = 30)

wine_unique <- wine[!duplicated(winescaled), ]
tsne_df <- data.frame(tsne_results$Y, wine_unique[, 12:13])
colnames(tsne_df) <- c("Dim1", "Dim2","quality")
tsne_df$color <- factor(wine_unique$color)

#Plot t-SNE results, colored by wine color
ggplot(tsne_df, aes(x = Dim1, y = Dim2, color = color)) +
  geom_point(alpha = 0.7) +
  labs(title = "t-SNE- Wine Color")
```


```
#Plot t-SNE results, colored by wine quality
ggplot(tsne_df, aes(x = Dim1, y = Dim2, color = quality)) +
  geom_point(alpha = 0.7) +
  labs(title = "t-SNE- Wine Quality")
```

After analyzing the results of the following dimension reducing techniques: PCA, K-means clustering, and tSNE, tSNE best distinguishes wine color. From the plot above, we see a distinct separation of wine colors and thus suggests that the chemical property are informative in understanding the relationship of wine color. However, when trying to capture information about wine quality, none of these methods successfully distinguishes higher quality wines from lower quality wines.

Market segmentation

```
library(factoextra)
```

```
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WB
a
```

```
library(corrplot)
```

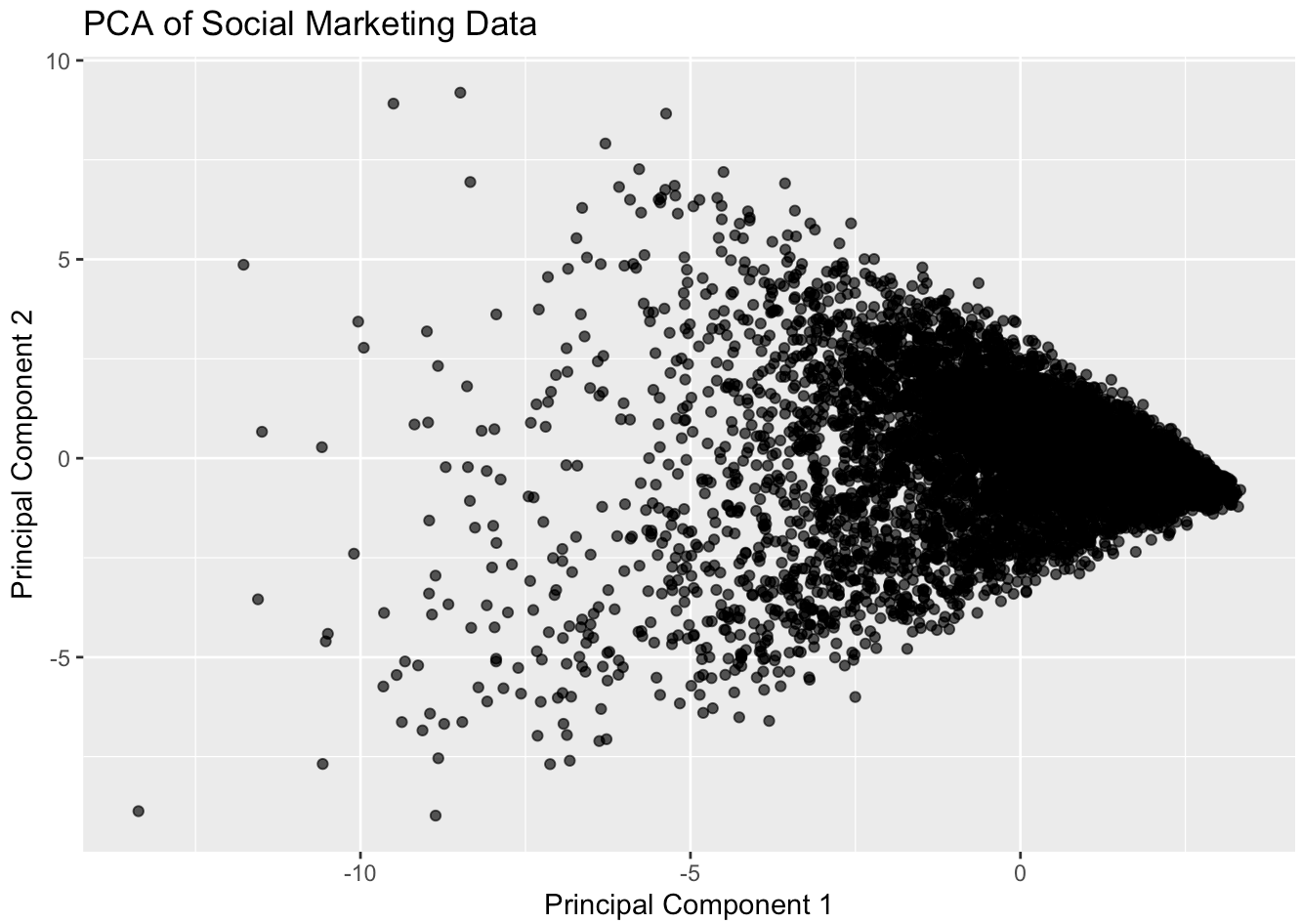
```
## corrplot 0.92 loaded
```

```
sns=read.csv("/Users/sarahlee/Downloads/social_marketing.csv")

summary(sns)
```

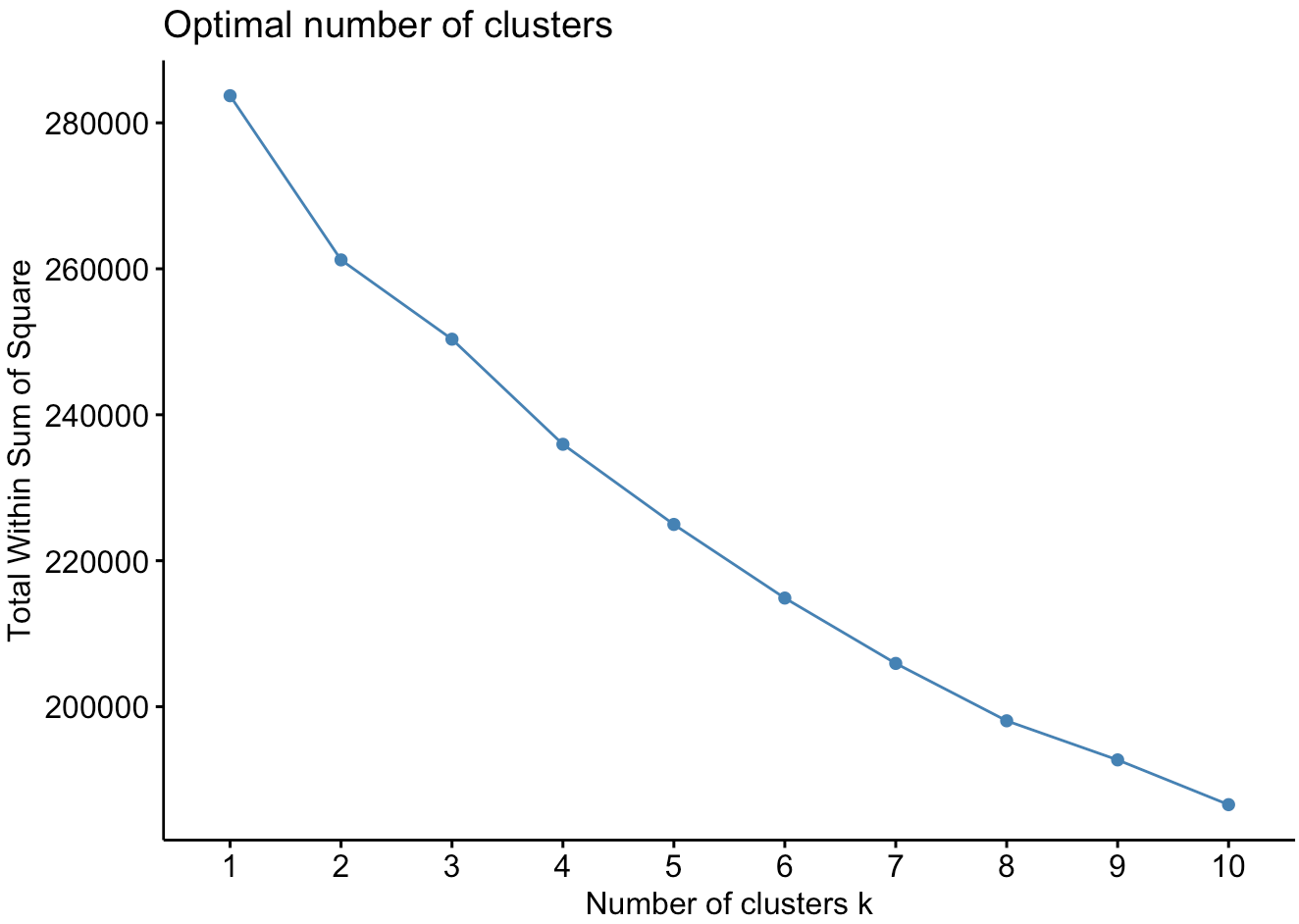
```
##       X                chatter       current_events      travel
##  Length:7882        Min.   : 0.000   Min.   :0.000   Min.   : 0.000
##  Class :character   1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 0.000
##  Mode  :character   Median : 3.000   Median :1.000   Median : 1.000
##              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

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

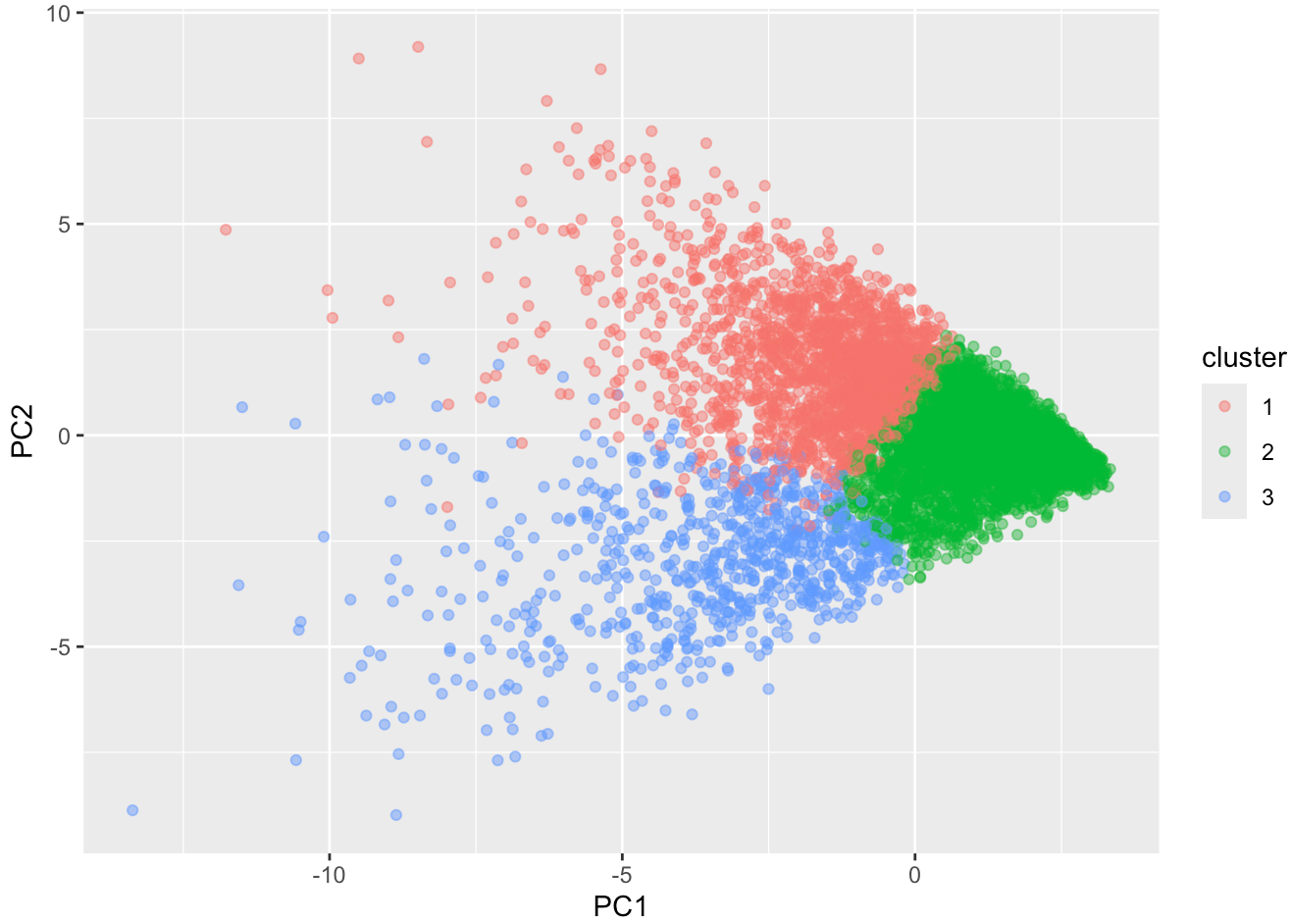
```
sns[is.na(sns)]=0
sns_normalized=as.data.frame(scale(sns[,-1]))
sns_normalized$X=sns$X

sns_pca=prcomp(sns_normalized[,-ncol(sns_normalized)],center=T,scale. = TRUE)
sns_pca_df <- data.frame(sns_pca$x)
sns_pca_df$X <- sns_normalized$X

#visualization
ggplot(sns_pca_df, aes(x = PC1, y = PC2)) +
  geom_point(alpha = 0.7) +
  labs(title = "PCA of Social Marketing Data",
       x = "Principal Component 1",
       y = "Principal Component 2")
```




```
# plots the first two principal components to capture the most variance in data

fviz_nbclust(sns_normalized[, -ncol(sns_normalized)], kmeans, method = "wss")
```



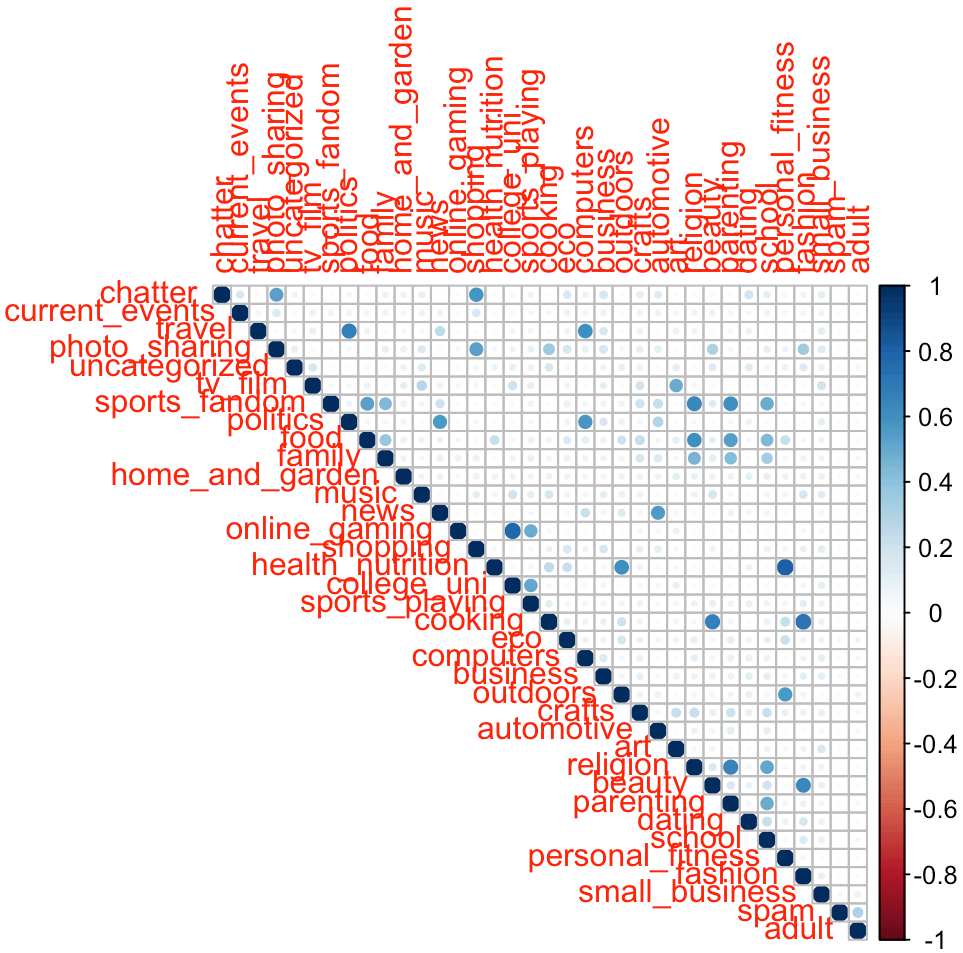

```
#optimal number of clusters at 3 using the elbow method
set.seed(1)
sns_kmeans=kmeans(sns_normalized[,-ncol(sns_normalized)],center=3,nstart=25)
sns_pca_df$cluster <- factor(sns_kmeans$cluster)

ggplot(sns_pca_df, aes(x=PC1, y=PC2, color=cluster))+
  geom_point(alpha=0.5)
```




```
# This shows the 3 distinct clusters with the twitter followers as each clusters represe
nt the market segment based on the their interests
```

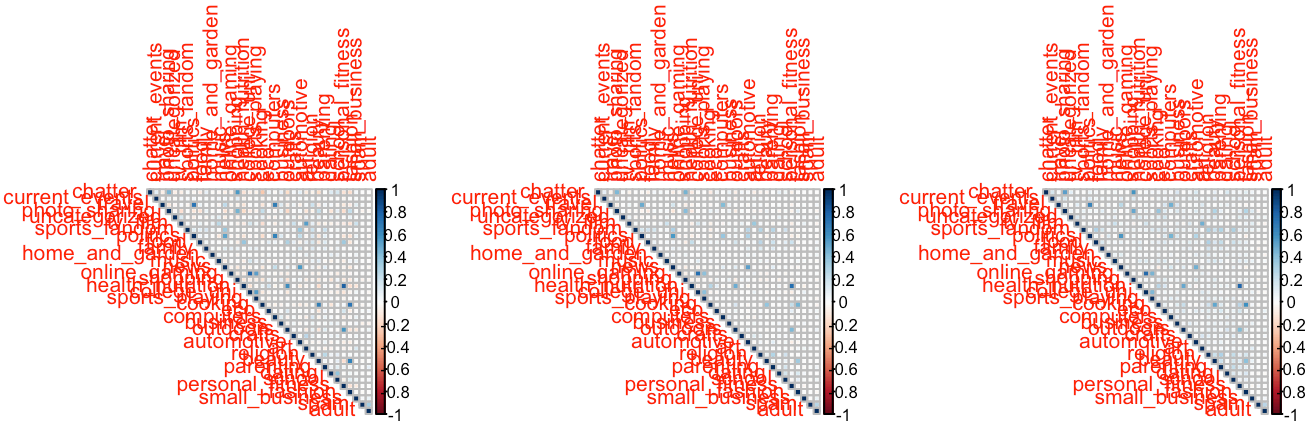
The PCA plot shows how the data points are distributed in the first two principal components. The subsequent clustering analysis suggests that the Twitter followers can be divided into 3 distinct segments (or market segments). The clustering reflects the 3 different groups of followers with different sets of interests.

```
# Calculate the correlation matrix
cor_matrix <- cor(sns_normalized[,-ncol(sns_normalized)])

# Visualize the correlation matrix
corrplot(cor_matrix, type = "upper")
```




```
sns_normalized$cluster <- sns_kmeans$cluster
plot_cluster_correlation <- function(cluster_number) {
  cluster_data <- sns_normalized %>% filter(cluster == cluster_number) %>% select(-X, -c
luster)
  cor_matrix <- cor(cluster_data)
  corrplot(cor_matrix, method = "circle", type = "upper", title = paste("Correlation Mat
rix for Cluster", cluster_number), mar=c(0,0,1,0))
}

# Plot correlation matrices for each cluster
par(mfrow=c(1,3))  # Adjust layout to fit all plots in one window
plot_cluster_correlation(1)
plot_cluster_correlation(2)
plot_cluster_correlation(3)
```

Correlation Matrix for Cluster 1 **Correlation Matrix for Cluster 2** **Correlation Matrix for Cluster 3**



```
show_top_correlations <- function(cluster_number, top_n = 5) {
  cluster_data <- sns_normalized %>% filter(cluster == cluster_number) %>% select(-X, -c
luster)
  cor_matrix <- cor(cluster_data)

  # Get the upper triangle of the correlation matrix
  upper_tri <- cor_matrix
  upper_tri[lower.tri(upper_tri, diag = TRUE)] <- NA

  cor_long <- as.data.frame(as.table(upper_tri))

  # Filter for the top n most correlated pairs
  top_correlations <- cor_long %>%
    filter(!is.na(Freq)) %>%
    arrange(desc(abs(Freq))) %>%
    head(top_n)

  print(paste("Top", top_n, "correlated variable pairs in Cluster", cluster_number))
  print(top_correlations)

  return(top_correlations)
}

top_correlations_cluster_1 <- show_top_correlations(1, top_n = 5)
```

```
## [1] "Top 5 correlated variable pairs in Cluster 1"
##               Var1             Var2      Freq
## 1 health_nutrition personal_fitness 0.8181738
## 2    online_gaming      college_uni 0.7914516
## 3           travel         politics 0.7278171
## 4          cooking          fashion 0.7190307
## 5          cooking           beauty 0.6874663
```

```
top_correlations_cluster_2 <- show_top_correlations(2, top_n = 5)
```

```
## [1] "Top 5 correlated variable pairs in Cluster 2"
##               Var1             Var2      Freq
## 1 health_nutrition personal_fitness 0.7377869
## 2    online_gaming      college_uni 0.7359980
## 3         politics             news 0.5610353
## 4             news       automotive 0.5321555
## 5           travel         politics 0.4935320
```

```
top_correlations_cluster_3 <- show_top_correlations(3, top_n = 5)
```

```
## [1] "Top 5 correlated variable pairs in Cluster 3"
##               Var1             Var2      Freq
## 1 health_nutrition personal_fitness 0.7876688
## 2    online_gaming      college_uni 0.7615645
## 3           travel         politics 0.6530600
## 4          cooking          fashion 0.6088459
## 5          chatter         shopping 0.5891064
```

Overall, health_nutrition and personal_fitness with online_gaming and college_uni all represented each of the clusters' interests as it was the highest correlated variables for all three segments/clusters. This suggests that these interests tend to co-occur frequently across the segments, potentially indicating key areas of focus for marketing strategies.

The Reuters corpus

Question: Can we identify distinct clusters of writing styles in Benjamin Kang Lim's articles using PCA and hierarchical clustering?

Approach: 1. Pre Processing text data 2. TFIDF 3. PCA (dimension reducing) 4. Clustering

```
library(tm)
```

```
## Loading required package: NLP
```

```
##
## Attaching package: 'NLP'
```

```
## The following object is masked from 'package:ggplot2':
##
##     annotate
```

```
##
## Attaching package: 'tm'
```

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

```
library(tidyverse)
```

```
## ── Attaching core tidyverse packages ──────────────────────── tidyverse 2.0.0 ──
## ✔ forcats   1.0.0     ✔ readr     2.1.5
## ✔ lubridate 1.9.3     ✔ stringr   1.5.1
## ✔ purrr     1.0.2     ✔ tibble    3.2.1
```

```
## ── Conflicts ────────────────────────────────────────── tidyverse_conflicts() ──
## ✖ purrr::accumulate() masks foreach::accumulate()
## ✖ NLP::annotate()     masks ggplot2::annotate()
## ✖ mosaic::count()     masks dplyr::count()
## ✖ purrr::cross()      masks mosaic::cross()
## ✖ mosaic::do()        masks dplyr::do()
## ✖ Matrix::expand()    masks tidyr::expand()
## ✖ dplyr::filter()     masks stats::filter()
## ✖ xts::first()        masks dplyr::first()
## ✖ dplyr::lag()        masks stats::lag()
## ✖ xts::last()         masks dplyr::last()
## ✖ Matrix::pack()      masks tidyr::pack()
## ✖ mosaic::stat()      masks ggplot2::stat()
## ✖ mosaic::tally()     masks dplyr::tally()
## ✖ Matrix::unpack()    masks tidyr::unpack()
## ✖ purrr::when()       masks foreach::when()
## ℹ Use the conflicted package (<http://conflicted.r-lib.org/>) to force all conflicts
to become errors
```

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

```
##
## Attaching package: 'proxy'
##
## The following object is masked from 'package:Matrix':
##
##     as.matrix
##
## The following objects are masked from 'package:stats':
##
##     as.dist, dist
##
## The following object is masked from 'package:base':
##
##     as.matrix
```

```
library(RCurl)
```

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

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

file_list = Sys.glob('/Users/sarahlee/Downloads/STA380-master/data/ReutersC50/C50train/B
enjaminKangLim/*.txt')
ben = lapply(file_list, readerPlain)

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

mynames
```

```
##  [1] "BenjaminKangLim102444newsML.txt" "BenjaminKangLim106762newsML.txt"
##  [3] "BenjaminKangLim110733newsML.txt" "BenjaminKangLim112125newsML.txt"
##  [5] "BenjaminKangLim114204newsML.txt" "BenjaminKangLim115482newsML.txt"
##  [7] "BenjaminKangLim118614newsML.txt" "BenjaminKangLim118687newsML.txt"
##  [9] "BenjaminKangLim12228newsML.txt"  "BenjaminKangLim129162newsML.txt"
## [11] "BenjaminKangLim133520newsML.txt" "BenjaminKangLim135232newsML.txt"
## [13] "BenjaminKangLim14439newsML.txt"  "BenjaminKangLim145148newsML.txt"
## [15] "BenjaminKangLim145736newsML.txt" "BenjaminKangLim146225newsML.txt"
## [17] "BenjaminKangLim150365newsML.txt" "BenjaminKangLim151860newsML.txt"
## [19] "BenjaminKangLim155059newsML.txt" "BenjaminKangLim155439newsML.txt"
## [21] "BenjaminKangLim15741newsML.txt"  "BenjaminKangLim166435newsML.txt"
## [23] "BenjaminKangLim174581newsML.txt" "BenjaminKangLim177874newsML.txt"
## [25] "BenjaminKangLim178518newsML.txt" "BenjaminKangLim182490newsML.txt"
## [27] "BenjaminKangLim18363newsML.txt"  "BenjaminKangLim186173newsML.txt"
## [29] "BenjaminKangLim186199newsML.txt" "BenjaminKangLim187423newsML.txt"
## [31] "BenjaminKangLim188955newsML.txt" "BenjaminKangLim188988newsML.txt"
## [33] "BenjaminKangLim189031newsML.txt" "BenjaminKangLim190439newsML.txt"
## [35] "BenjaminKangLim192394newsML.txt" "BenjaminKangLim195350newsML.txt"
## [37] "BenjaminKangLim198232newsML.txt" "BenjaminKangLim201374newsML.txt"
## [39] "BenjaminKangLim204845newsML.txt" "BenjaminKangLim211064newsML.txt"
## [41] "BenjaminKangLim214183newsML.txt" "BenjaminKangLim21575newsML.txt"
## [43] "BenjaminKangLim221670newsML.txt" "BenjaminKangLim222816newsML.txt"
## [45] "BenjaminKangLim231106newsML.txt" "BenjaminKangLim232863newsML.txt"
## [47] "BenjaminKangLim235646newsML.txt" "BenjaminKangLim241295newsML.txt"
## [49] "BenjaminKangLim241719newsML.txt" "BenjaminKangLim24300newsML.txt"
```

```
names(ben) = mynames
documents_raw = Corpus(VectorSource(ben))

#preprocessing text data
my_documents = documents_raw
my_documents = tm_map(my_documents, content_transformer(removePunctuation))
```

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

```
my_documents = tm_map(my_documents, content_transformer(tolower))
```

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

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

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

```
my_documents = tm_map(my_documents, content_transformer(removeNumbers))
```

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

```
my_documents = tm_map(my_documents, content_transformer(stripWhitespace))
```

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

```
DTM_ben = DocumentTermMatrix(my_documents)

inspect(DTM_ben[1:10, 1:20]) #frequency of terms in the first 10 documents and first 20
terms
```

```
## <<DocumentTermMatrix (documents: 10, terms: 20)>>
## Non-/sparse entries: 37/163
## Sparsity           : 82%
## Maximal term length: 11
## Weighting          : term frequency (tf)
## Sample             :
##     Terms
## Docs abandon accused adding ago allows army august authorities ban banned
##   1        1       1      1   1      1    1      1           1   6      1
##   10       0       0      0   0      0    2      0           0   0      0
##   2        0       1      0   1      0    0      0           0   0      0
##   3        0       0      0   0      0    0      0           0   1      1
##   4        0       0      0   0      0    0      0           0   0      0
##   5        0       2      1   0      0    1      0           0   0      0
##   6        0       2      1   0      0    1      0           0   0      0
##   7        0       1      0   0      1    1      0           0   0      0
##   8        0       0      1   0      0    0      2           2   0      0
##   9        0       0      0   0      0    0      0           0   0      0
```

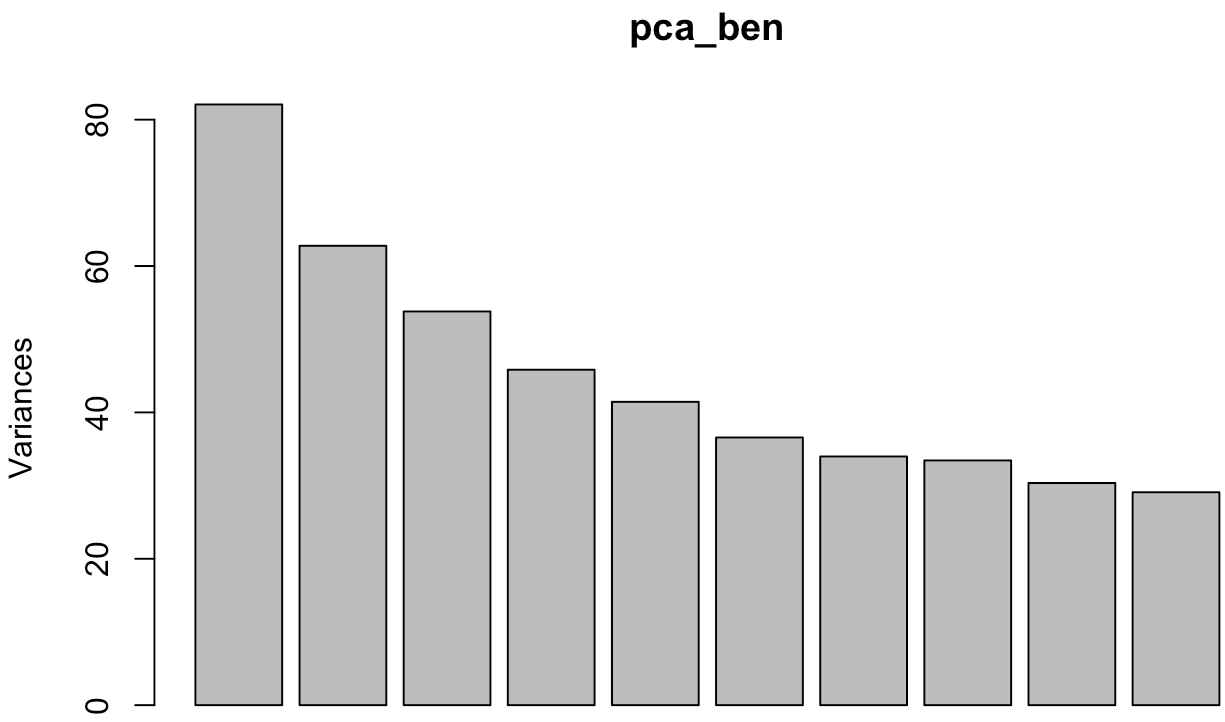
```
findAssocs(DTM_ben, "ban", .5) #terms highly associated with 'ban'
```

```
## $ban
##          lift reunification      shipping      advocate         allay
##          0.91          0.87          0.86          0.83          0.83
##     animosity     cbeijings    clamouring       declare          dock
##          0.83          0.83          0.83          0.83          0.83
##           eat       enclave       glaring         kongs     landslide
##          0.83          0.83          0.83          0.83          0.83
##         lifts         logic        losses         macau        macaus
##          0.83          0.83          0.83          0.83          0.83
##    negotiator      patience    portuguese      resisted       resolve
##          0.83          0.83          0.83          0.83          0.83
##    ridiculous          rock      scrapped     scrapping         ships
##          0.83          0.83          0.83          0.83          0.83
##        shubei         spell     squashing       straits       strange
##          0.83          0.83          0.83          0.83          0.83
##       swallow        direct         third    decadesold     conducted
##          0.83          0.81          0.81          0.78          0.77
##       hostage          lest         links       profile      indirect
##          0.77          0.77          0.77          0.77          0.76
##          pave   regulations     elections        invade        needed
##          0.76          0.76          0.73          0.73          0.73
##  presidential       reverts       zedongs     isolation        taiwan
##          0.73          0.73          0.73          0.72          0.72
##    introduced           lee        poured        banned      landmark
##          0.71          0.71          0.71          0.70          0.70
##     obstacles        sought          tang       taiwans        island
##          0.69          0.69          0.69          0.68          0.66
##  unilaterally          dont      kaisheks           mid       tenghui
##          0.66          0.65          0.64          0.64          0.64
##  vicechairman     transport           way        allows       private
##          0.64          0.63          0.63          0.61          0.61
##         tests        called   nationalist         talks        across
##          0.61          0.59          0.59          0.59          0.58
##          hold       abandon      aircraft       attempt      concerns
##          0.58          0.56          0.56          0.56          0.56
##       greater        nearby       renewed      resolved       reunify
##          0.56          0.56          0.56          0.56          0.56
##           run          seek     timetable        return        troops
##          0.56          0.56          0.56          0.54          0.54
##     reluctant         china         offer     political    threatened
##          0.53          0.52          0.52          0.52          0.52
##      creating           end           saw        series
##          0.51          0.51          0.51          0.51
```

```
# Drop terms that only occur in one or two documents
DTM_ben = removeSparseTerms(DTM_ben, 0.95)
tfidf_ben = weightTfIdf(DTM_ben)

# PCA on term frequencies
X = as.matrix(tfidf_ben) # DTM to matrix
summary(colSums(X))
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## 0.00000 0.05727 0.07828 0.09637 0.11402 0.64106
```

```
scrub_cols = which(colSums(X) == 0)
X = X[,-scrub_cols]

pca_ben = prcomp(X, rank=2, scale=TRUE)
plot(pca_ben)
```



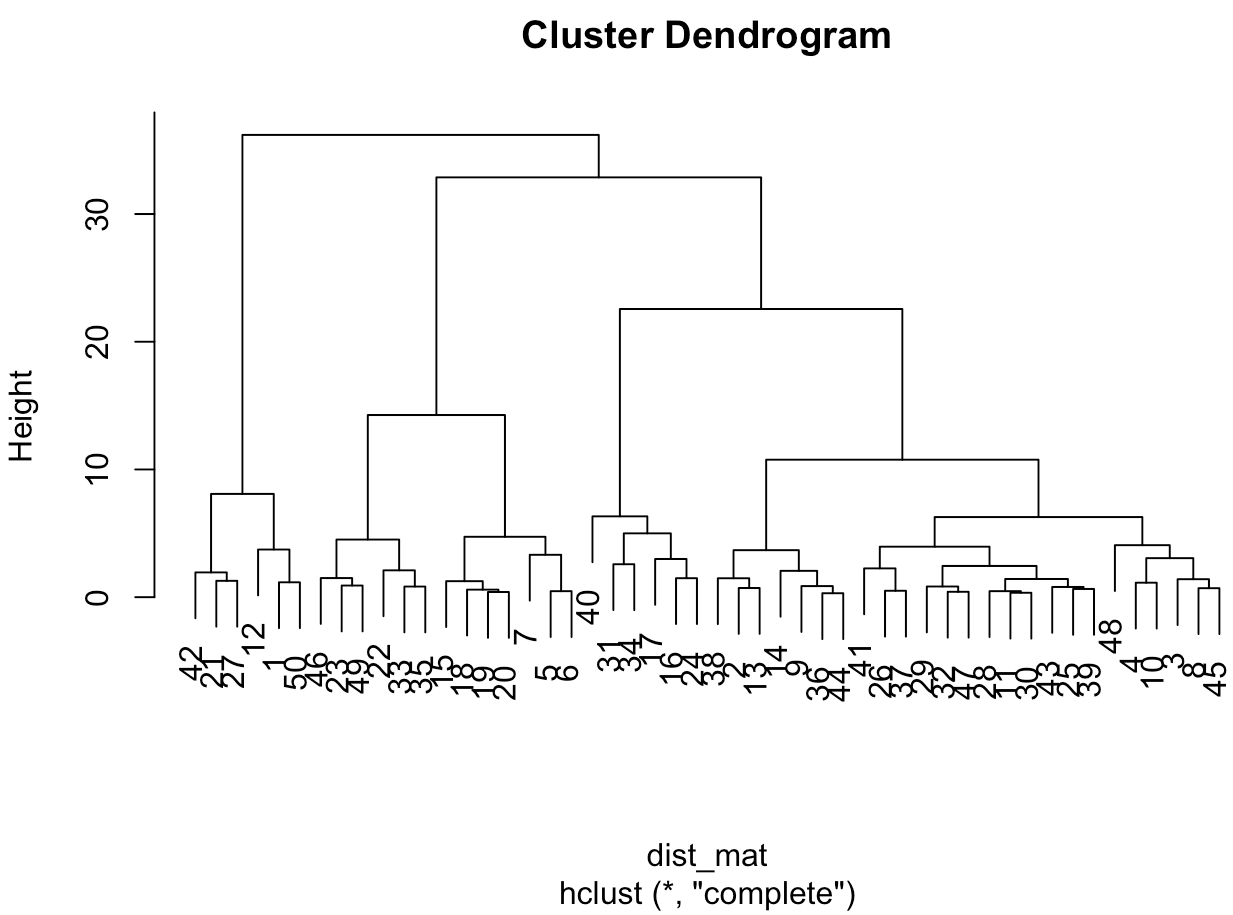

```
# top 25 loadings for the first two principal components
pca_ben$rotation[order(abs(pca_ben$rotation[,1]),decreasing=TRUE),1][1:25]
```

```
##         served        crushed      overthrow         charge     subversion
##     0.09696902     0.09667856     0.09219762     0.09219083     0.09193588
##     background         daring       defying        maximum         museum
##     0.09189404     0.09189404     0.09189404     0.09189404     0.09189404
##      jingsheng      dissident         family          heavy   prodemocracy
##     0.09097314     0.08987086     0.08851471     0.08819839     0.08620124
##       plotting         mother            dan demonstrations        student
##     0.08579277     0.08520757     0.08510588     0.08485271     0.08484461
##          trial   surveillance        willing        lingyun           join
##     0.08433636     0.08406755     0.08355653     0.08327425     0.08281053
```

```
pca_ben$rotation[order(abs(pca_ben$rotation[,2]),decreasing=TRUE),2][1:25]
```

```
##    comeback  eliminated   virtually       enjoy princelings        wide
## -0.08821156 -0.08777765 -0.08657235 -0.08487186 -0.08487186 -0.08487186
##  corruption      guanwu        iron      staged    indirect        pave
## -0.08400069 -0.08042645 -0.08042645 -0.08004075  0.07966690  0.07966690
## regulations       cases      direct       party       talks    resigned
##  0.07966690 -0.07952644  0.07851628 -0.07831324  0.07794316 -0.07771268
##       april         lee    landmark   taiwanese     taiwans     reforms
## -0.07724816  0.07724347  0.07679678  0.07674367  0.07668220 -0.07634686
##      poured
##  0.07625332
```

```
#Hierarchical Clustering
dist_mat = dist(pca_ben$x)
tree_ben = hclust(dist_mat)
plot(tree_ben) # closer together in the dendrogram = more similar
```




```
#Cut the tree to form 5 clusters
clust5 = cutree(tree_ben, k=5)

#two specific documents (5&19) in Cluster 3
which(clust5 == 3)
```

```
##  5  6  7 15 18 19 20
##  5  6  7 15 18 19 20
```

```
content(ben[[5]])
```

```
##  [1] "The mother of detained Chinese dissident Wang Dan said on Monday she would defe
nd her son against the capital charge of plotting to overthrow the government and that h
e was prepared for a heavy sentence."
##  [2] "\"Two defence counsels are allowed...I will be one of them,\" Wang Lingyun, a 6
1-year-old researcher at a museum who has no background in law, told Reuters in an inter
view."
##  [3] "\"Wang Dan also wants me to defend him,\" she said. Chinese laws allow accused
to be defended by family members."
##  [4] "A court spokeswoman confirmed Wang Dan, 26, had been charged with plotting to o
verthrow the government but declined to give further details."
##  [5] "Wang Dan's mother said the court had yet to inform her of the trial date, but s
aid it could come as early as this week."
##  [6] "The dissident, detained without charge since May 1995, met his mother for the f
irst time under police surveillance at a detention centre in Beijing, the Hong Kong-base
d Information Centre of Human Rights and Democratic Movement in China said."
##  [7] "\"Wang Dan...was very calm and psychologically prepared for a heavy sentence,\"
the group quoted the mother as saying. She added that she was worried about her son's he
alth."
##  [8] "She told Reuters earlier that she would defend her son against the charge of co
llaborating with overseas subversive forces. \"This charge does not stand up,\" she sai
d, adding that Wang Dan had only taken a correspondence course at the University of Cali
fornia, Berkeley, in the United States."
##  [9] "\"It has nothing to do with politics. I was the one who wanted him to attend th
e course...It has nothing to do with overthrowing the government,\" she said."
## [10] "Asked to comment on the chances of winning the case, she said: \"I'm not optimi
stic. But I must say it for the record. This will become history.\""
## [11] "Relatives said last week they had found a lawyer willing to defend Wang after b
eing given one day to find one."
## [12] "The lawyer retained by the family would defend the dissident against other char
ges in the indictment."
## [13] "Wang Dan, former leader of the 1989 pro-democracy demonstrations, has been char
ged with the capital offence of plotting to subvert the government, based on evidence su
ch as writings critical of the state and accepting funds from abroad."
## [14] "He was detained by police in a raid on his home in May 1995, but was not formal
ly arrested or charged until last week."
## [15] "On Friday, the family obtained a copy of the bill of indictment, which accused
Wang Dan of plotting to subvert the government, a crime that carries a maximum penalty o
f death. The minimum sentence is 10 years, although the court can show leniency if it fi
nds extenuating circumstances."
## [16] "Wang has already served four years in prison for counter-revolutionary crimes,
or subversion, for his role in the 1989 demonstrations centred in Beijing's Tiananmen Sq
uare, which were crushed by the army with heavy loss of life."
## [17] "Wang had been expected to face new charges since last December, when the court
that convicted veteran democracy activist Wei Jingsheng of plotting to overthrow the gov
ernment also implicated the former student leader."
## [18] "The court's verdict said Wei, who was jailed for 14 years, had links with peopl
e \"convicted of counter-revolutionary crimes, including Wang Dan\". It also referred to
a tape-recorded conversation between Wang and Wei, but gave no details."
## [19] "Wang had been active since his release from jail, defying persistent police sur
veillance and harassment to join in a daring appeal to communist leaders for the release
of all those still in prison for their part in the 1989 protests."
```

```
content(ben[[19]])
```

```
##  [1] "China began the trial of prominent dissident and former student leader Wang Dan
on Wednesday on the capital charge of plotting to overthrow the government, a court offi
cial said."
##  [2] "\"The trial started at 9.00 a.m. (0100 GMT),\" said an official of the Beijing
People's Intermediate Court reached by telephone. He declined to give further details."
##  [3] "Security was tight around the court building in western Beijing, with dozens of
police preventing the public from approaching the building and manning roadblocks to pre
vent access."
##  [4] "The father of the former student leader protested his son's innocence just hour
s before the start of the proceedings and said the family would not give in to the autho
rities."
##  [5] "\"He is definitely innocent,\" Wang Xianzeng told reporters as he left his home
with the dissident's mother, Wang Lingyun, to go to the court. \"It just depends on whet
her the government wants to convict him or not.\""
##  [6] "Wang's mother, a 61-year old museum researcher who has no background in law, wo
uld attend the trial as one of two defence lawyers. His father and a sister would be all
owed to sit in."
##  [7] "\"I'm not optimistic about the results,\" Wang Xianzeng said. \"We will definit
ely appeal... We will not give in."
##  [8] "\"Speech can't overthrow the government,\" he said."
##  [9] "The court indictment against Wang, 27, includes the charge that his writings in
foreign publications were evidence of his plot to try to overthrow the government."
## [10] "He is also accused of accepting foreign funds, of colluding with subversives li
ving overseas and of conspiring with domestic plotters to organise the overthrow of the
government."
## [11] "Wang Lingyun said on Tuesday she expected Wang to receive a harsh sentence alth
ough he would plead not guilty."
## [12] "The former student leader faces a minimum 10-year sentence and a maximum penalt
y of death."
## [13] "The mother has said the dissident was calm and mentally prepared for a harsh se
ntence, although she has said his health had deteriorated since he vanished into detenti
on in May 1995."
## [14] "She said court officials told her the verdict could be delivered as early as We
dnesday or in two days. Family members had been under surveillance for several weeks, sh
e added."
## [15] "The New York-based Human Rights Watch said last week the chances of acquittal w
ere slim because Wang has not had adequate time to prepare a defence."
## [16] "Human Rights Watch attacked the trial on Tuesday as a sign of the Chinese leade
rship's increasing intolerance of dissent."
## [17] "\"The fact is that China's urban dissident movement...has in effect been compre
hensively smashed,\" it said in a statement."
## [18] "Wang's court appearance was expected to resemble the in-camera proceedings last
December against Wei Jingsheng, regarded as the father of China's tiny, struggling democ
racy movement."
## [19] "Wang has already served four years in prison for counter-revolutionary crimes,
or subversion, for his role in pro-democracy demonstrations in Beijing's Tiananmen Squar
e that were crushed by the army in June 1989 with heavy loss of life."
## [20] "He was politically active again after his parole in 1993, defying police survei
llance to join a daring appeal to communist leaders for the release of those still jaile
d for their part in the 1989 protests."
```

```
## [21] "China has recently cracked down on the few remaining dissidents who have not fl
ed into exile or been jailed."
```

Results: Through PCA, text data dimensionality was effectively reduced, retaining thematic information. According to the explained variance plot, the first few components capture the majority of the variance in the data. Furthermore, the top loadings showed that the text has the themes of political activism, opposition to governemt, and issues related to governance. Through clustering, documents were grouped based on their similiarities. Specifically, in cluster 3, text was mainly about Chinese political issues.

Conclusion: The analysis of Benjamin Kang Lim's identified thematic clusters within the author's texts. PCA was effective in reducing the dimensionality of text data while retaining significant information. The first two principal components showed the themes of political activism. Hierarchical clustering grouped text files based of similar themes. This not only highlights the diversity in author's themes but also suggests certain topics are recurrent and can be categorized. Overall, the analysis demonstrated patterns of Benjamin Kang Lim's textual data, enabling a deeper understanding of author's focus and writing style.

Association rule mining

```
library(tidyverse)
library(igraph)
```

```
##
## Attaching package: 'igraph'
```

```
## The following objects are masked from 'package:lubridate':
##
##     %--%, union
```

```
## The following objects are masked from 'package:purrr':
##
##     compose, simplify
```

```
## The following object is masked from 'package:tibble':
##
##     as_data_frame
```

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

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

```
## The following objects are masked from 'package:dplyr':
##
##     as_data_frame, groups, union
```

```
## The following objects are masked from 'package:stats':
##
##     decompose, spectrum
```

```
## The following object is masked from 'package:base':
##
##     union
```

```
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)

#support=proportion of transactions in the dataset that contain a particular itemset(Hig
her support indicates more frequent occurrence.)
#confidence=Confidence is the proportion of transactions containing item A that also con
tain item B.(Higher confidence indicates stronger association.)
#lift=ratio of the observed support to that expected if A and B were independent.

grocery=read.transactions("/Users/sarahlee/Downloads/groceries.txt", format="basket", se
p=",")
summary(grocery)
```

```
## transactions as itemMatrix in sparse format with
##  9835 rows (elements/itemsets/transactions) and
##  169 columns (items) and a density of 0.02609146
##
## most frequent items:
##       whole milk other vegetables       rolls/buns             soda
##             2513             1903             1809             1715
##           yogurt          (Other)
##             1372            34055
##
## element (itemset/transaction) length distribution:
## sizes
##    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
## 2159 1643 1299 1005  855  645  545  438  350  246  182  117   78   77   55   46
##   17   18   19   20   21   22   23   24   26   27   28   29   32
##   29   14   14    9   11    4    6    1    1    1    1    3    1
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   4.409   6.000  32.000
##
## includes extended item information - examples:
##             labels
## 1 abrasive cleaner
## 2 artif. sweetener
## 3   baby cosmetics
```

```
grocery_rules <- apriori(grocery, parameter = list(support = 0.01, confidence = 0.4, max
len = 4))
```

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.4    0.1    1 none FALSE            TRUE       5    0.01      1
##  maxlen target  ext
##       4  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
```

```
## Warning in apriori(grocery, parameter = list(support = 0.01, confidence = 0.4,
## : Mining stopped (maxlen reached). Only patterns up to a length of 4 returned!
```

```
##  done [0.00s].
## writing ... [62 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
#A low support parameter threshold of 0.001 to capture various discoveries of combinatio
ns
#I chose the confidence of 0.4 so that it captures a wider range of rules while filterin
g out weaker associations and not being overly restrictive

summary(grocery_rules)
```

```
## set of 62 rules
##
## rule length distribution (lhs + rhs):sizes
##  2  3
## 25 37
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   2.000   2.000   3.000   2.597   3.000   3.000
##
## summary of quality measures:
##     support          confidence        coverage            lift
##  Min.   :0.01007   Min.   :0.4016   Min.   :0.01729   Min.   :1.572
##  1st Qu.:0.01149   1st Qu.:0.4178   1st Qu.:0.02450   1st Qu.:1.729
##  Median :0.01388   Median :0.4502   Median :0.03005   Median :1.947
##  Mean   :0.01752   Mean   :0.4643   Mean   :0.03882   Mean   :1.993
##  3rd Qu.:0.01790   3rd Qu.:0.4975   3rd Qu.:0.04291   3rd Qu.:2.195
##  Max.   :0.05602   Max.   :0.5862   Max.   :0.13950   Max.   :3.030
##      count
##  Min.   : 99.0
##  1st Qu.:113.0
##  Median :136.5
##  Mean   :172.4
##  3rd Qu.:176.0
##  Max.   :551.0
##
## mining info:
##     data ntransactions support confidence
##  grocery          9835    0.01        0.4
##                                                                                  c
all
##  apriori(data = grocery, parameter = list(support = 0.01, confidence = 0.4, maxlen =
4))
```

```
class(grocery_rules)
```

```
## [1] "rules"
## attr(,"package")
## [1] "arules"
```

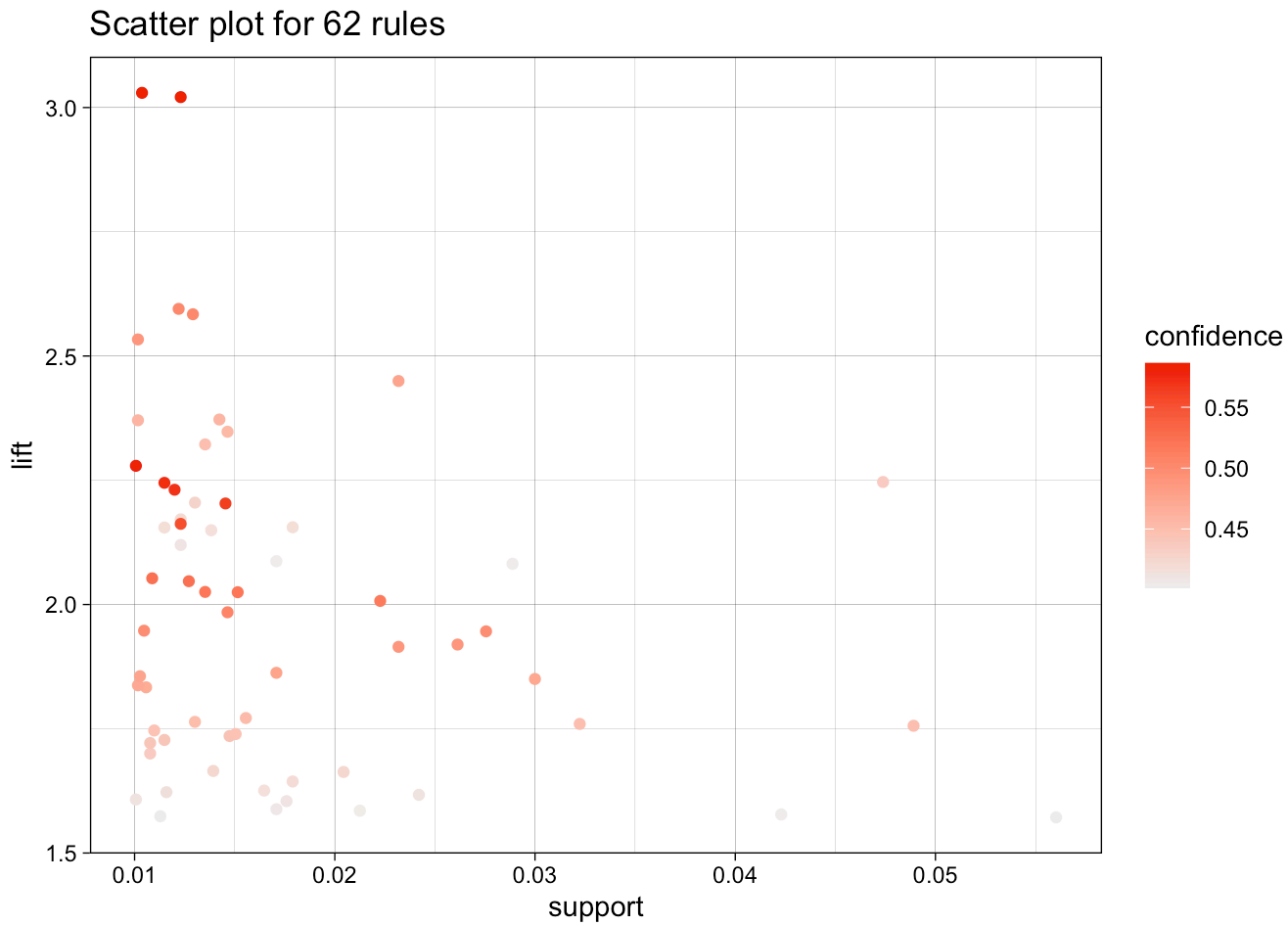
```
inspect(grocery_rules[1:10])
```

```
##      lhs                   rhs                support    confidence coverage
## [1]  {hard cheese}      => {whole milk}       0.01006609 0.4107884  0.02450432
## [2]  {butter milk}      => {whole milk}       0.01159126 0.4145455  0.02796136
## [3]  {ham}              => {whole milk}       0.01148958 0.4414062  0.02602949
## [4]  {sliced cheese}    => {whole milk}       0.01077783 0.4398340  0.02450432
## [5]  {oil}              => {whole milk}       0.01128622 0.4021739  0.02806304
## [6]  {onions}           => {other vegetables} 0.01423488 0.4590164  0.03101169
## [7]  {hamburger meat}   => {other vegetables} 0.01382816 0.4159021  0.03324860
## [8]  {hamburger meat}   => {whole milk}       0.01474326 0.4434251  0.03324860
## [9]  {sugar}            => {whole milk}       0.01504830 0.4444444  0.03385867
## [10] {cream cheese}     => {whole milk}       0.01647178 0.4153846  0.03965430
##      lift     count
## [1]  1.607682  99
## [2]  1.622385 114
## [3]  1.727509 113
## [4]  1.721356 106
## [5]  1.573968 111
## [6]  2.372268 140
## [7]  2.149447 136
## [8]  1.735410 145
## [9]  1.739400 148
## [10] 1.625670 162
```

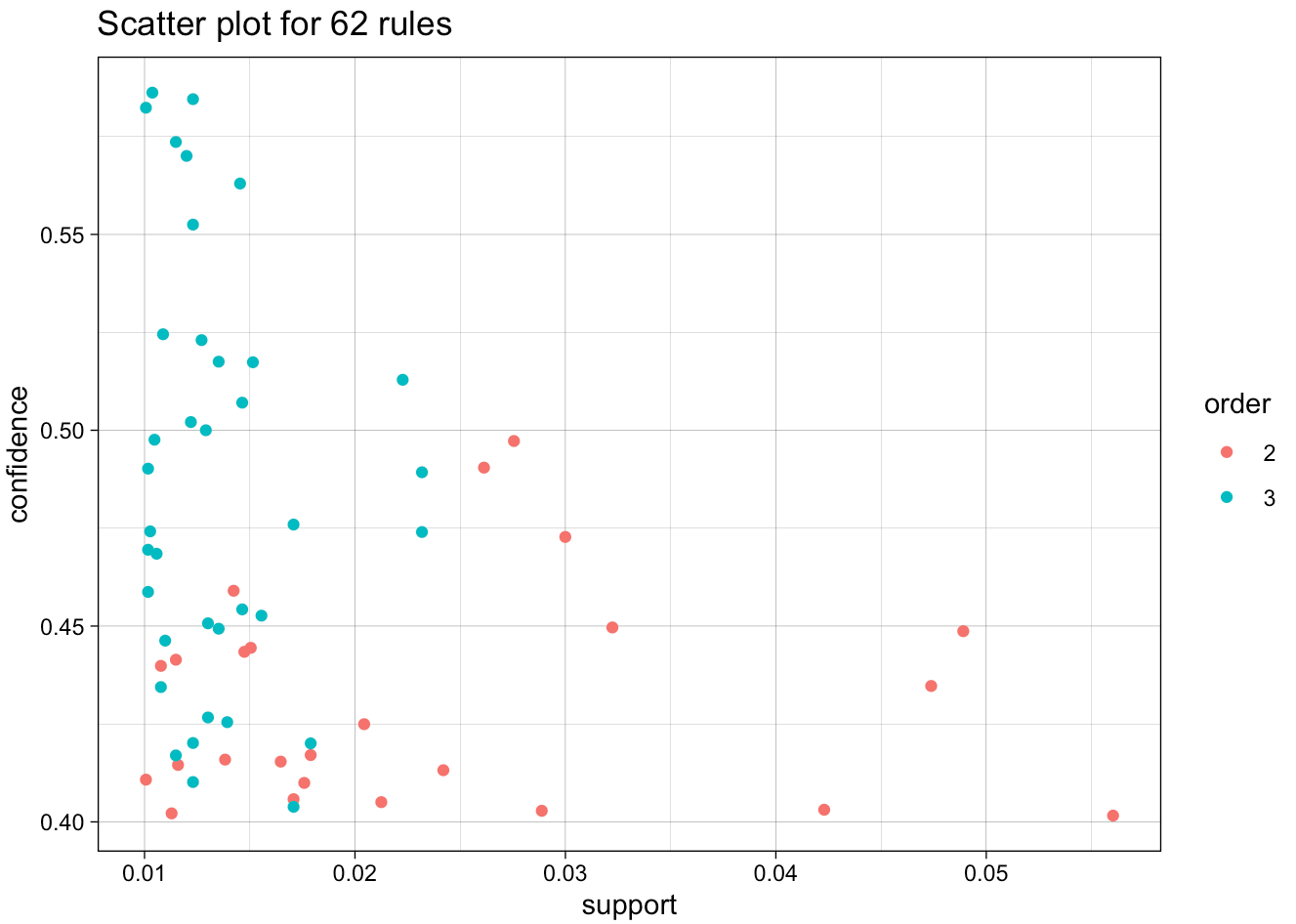
```
## top 10 baskets sorted by lift.
inspect(sort(grocery_rules, by = "lift")[1:10])
```

```
##      lhs                                rhs                support
## [1]  {citrus fruit, root vegetables}   => {other vegetables} 0.01037112
## [2]  {root vegetables, tropical fruit} => {other vegetables} 0.01230300
## [3]  {rolls/buns, root vegetables}     => {other vegetables} 0.01220132
## [4]  {root vegetables, yogurt}         => {other vegetables} 0.01291307
## [5]  {whipped/sour cream, yogurt}      => {other vegetables} 0.01016777
## [6]  {root vegetables, whole milk}     => {other vegetables} 0.02318251
## [7]  {onions}                          => {other vegetables} 0.01423488
## [8]  {pork, whole milk}                => {other vegetables} 0.01016777
## [9]  {whipped/sour cream, whole milk}  => {other vegetables} 0.01464159
## [10] {pip fruit, whole milk}           => {other vegetables} 0.01352313
##      confidence coverage   lift     count
## [1]  0.5862069  0.01769192 3.029608 102
## [2]  0.5845411  0.02104728 3.020999 121
## [3]  0.5020921  0.02430097 2.594890 120
## [4]  0.5000000  0.02582613 2.584078 127
## [5]  0.4901961  0.02074225 2.533410 100
## [6]  0.4740125  0.04890696 2.449770 228
## [7]  0.4590164  0.03101169 2.372268 140
## [8]  0.4587156  0.02216573 2.370714 100
## [9]  0.4542587  0.03223183 2.347679 144
## [10] 0.4493243  0.03009659 2.322178 133
```

```
#the highest lift value is at 21.49, meaning bottled beer,liquor strongly associates wit
h 'red/blush wine'. This makes sense as it is more likely for one to buy red/blush wine
when he or she buys bottled beer and liquor.


plot(grocery_rules, measure = c("support", "lift"), shading = "confidence",jitter = 0)
```




```
#Rules with higher lift values tend to be associated with lower support. The plot shows that there are several rules with very high lift (above 2.5) and moderate to high confidence (around 0.45 to 0.55), which suggests that these item combinations are strong and reliable, even if they don't occur very frequently

plot(grocery_rules, method='two-key plot',jitter = 0)
```




```
#Higher-order rules (order 3, in blue) tend to have slightly higher confidence than the
lower-order rules (order 2, in red). Higher support generally corresponds to lower confi
dence, which is typical because rules that apply to a large portion of the data (high su
pport) are often less specific.
```

Image classification with neural networks

in jupyter notebook