STA Part-2 Exercises

Chu Nie, Chang Li, Bolun Zhang, June Yuan

2022/8/6

Probability Practice

Part A

```
## [1] 0.7142857
```

So the fraction of people who are truthful clickers answered yes is 0.714.

Part B

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

So the probability that a person who tests positive have the disease is 0.1988824.

Wrangling the Billboard Top 100

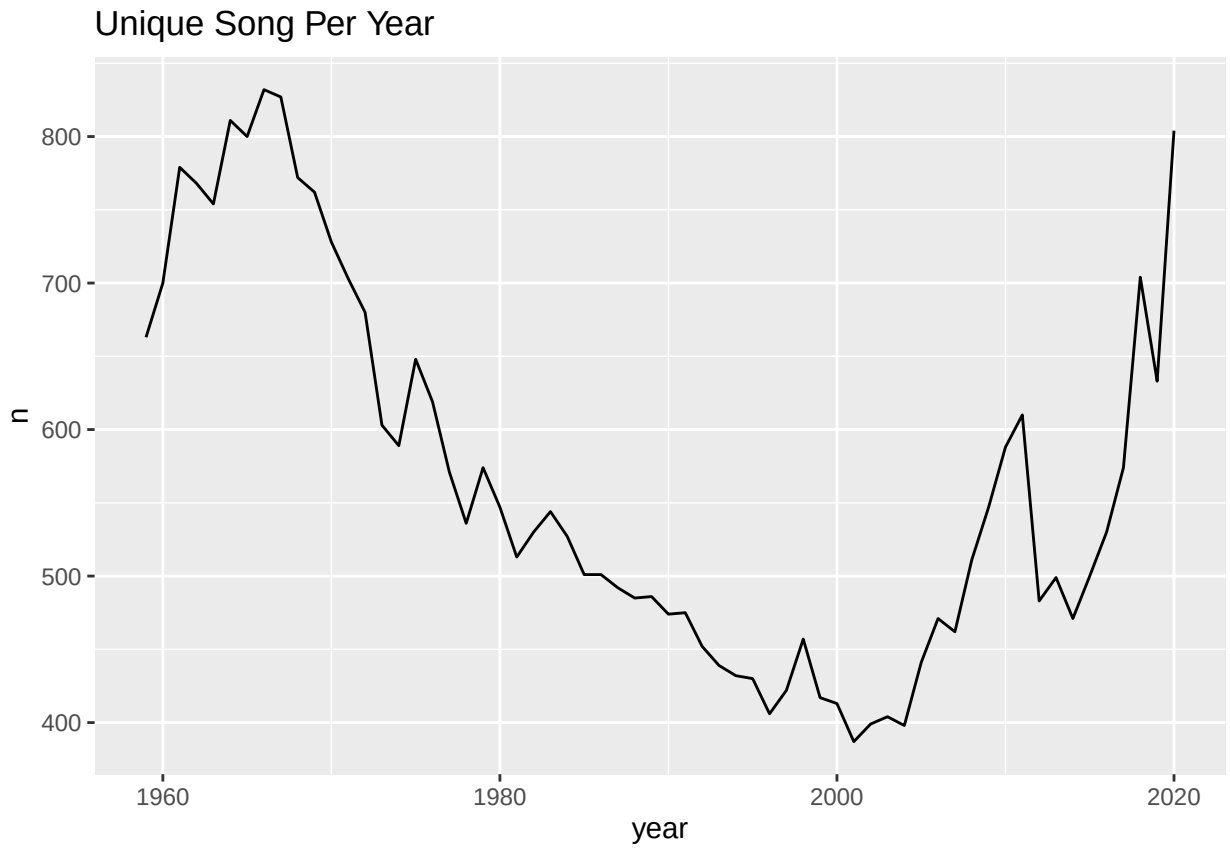
```
## New names:
## Rows: 327895 Columns: 13
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (5): url, week_id, song, performer, song_id dbl (8): ...1, week_position,
## instance, previous_week_position, peak_positio...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## * `` -> `...1`
```

```
## # A tibble: 6 x 13
##    ...1 url        week_id week_~1 song  perfo~2 song_id insta~3 previ~4 peak_~5
##   <dbl> <chr>      <chr>     <dbl> <chr> <chr>   <chr>     <dbl>   <dbl>   <dbl>
## 1     1 http://ww~ 7/17/1~      34 Don'~ Patty ~ Don't ~       1      45      34
## 2     2 http://ww~ 7/24/1~      22 Don'~ Patty ~ Don't ~       1      34      22
## 3     3 http://ww~ 7/31/1~      14 Don'~ Patty ~ Don't ~       1      22      14
## 4     4 http://ww~ 8/7/19~      10 Don'~ Patty ~ Don't ~       1      14      10
## 5     5 http://ww~ 8/14/1~       8 Don'~ Patty ~ Don't ~       1      10       8
## 6     6 http://ww~ 8/21/1~       8 Don'~ Patty ~ Don't ~       1       8       8
## # ... with 3 more variables: weeks_on_chart <dbl>, year <dbl>, week <dbl>, and
## #   abbreviated variable names 1: week_position, 2: performer, 3: instance,
## #   4: previous_week_position, 5: peak_position
## # i Use `colnames()` to see all variable names
```

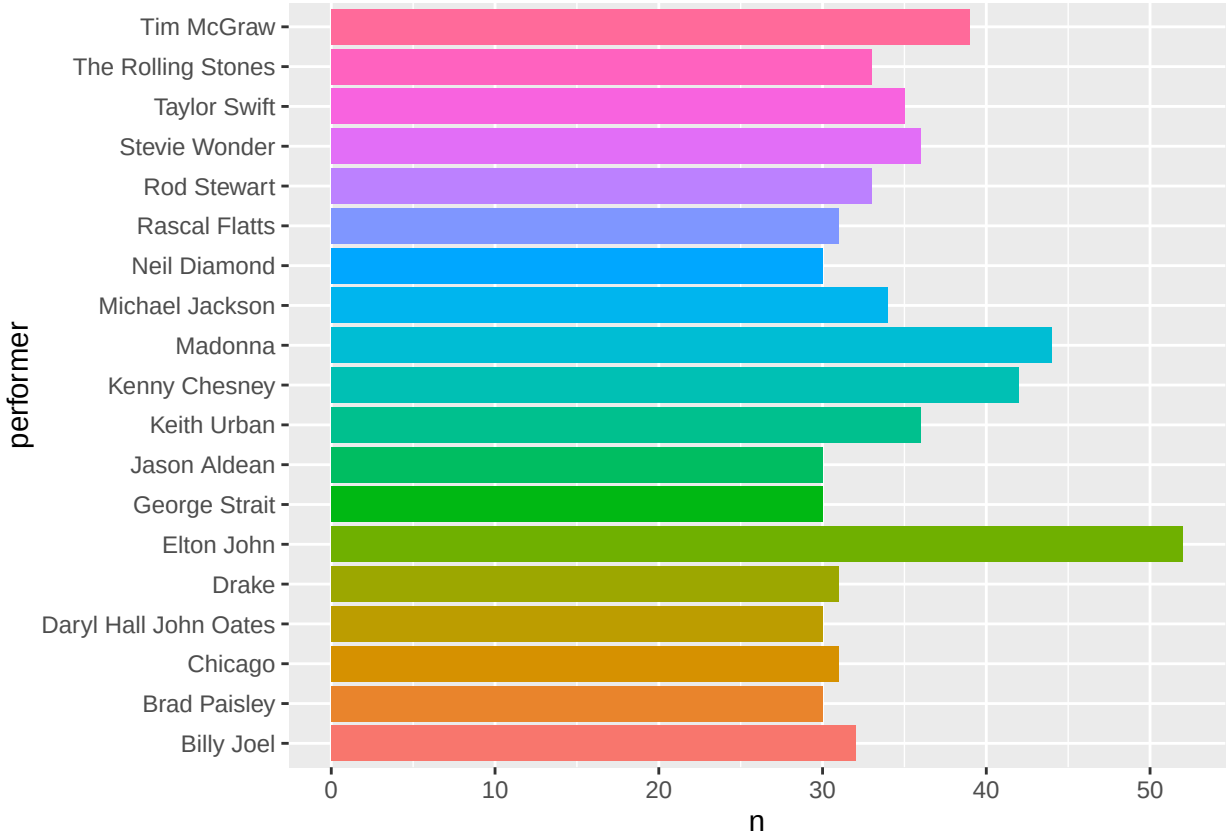
Part A

```
##
## 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

## ------------------------------------------------------------------------------

## You have loaded plyr after dplyr - this is likely to cause problems.
## If you need functions from both plyr and dplyr, please load plyr first, then dplyr:
## library(plyr); library(dplyr)

## ------------------------------------------------------------------------------

##
## Attaching package: 'plyr'

## The following objects are masked from 'package:dplyr':
##
##     arrange, count, desc, failwith, id, mutate, rename, summarise,
##     summarize

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

Part B


Unique Song Per Year
800
700
600
500
400
n
1960
1980
2000
2020
year

Part C



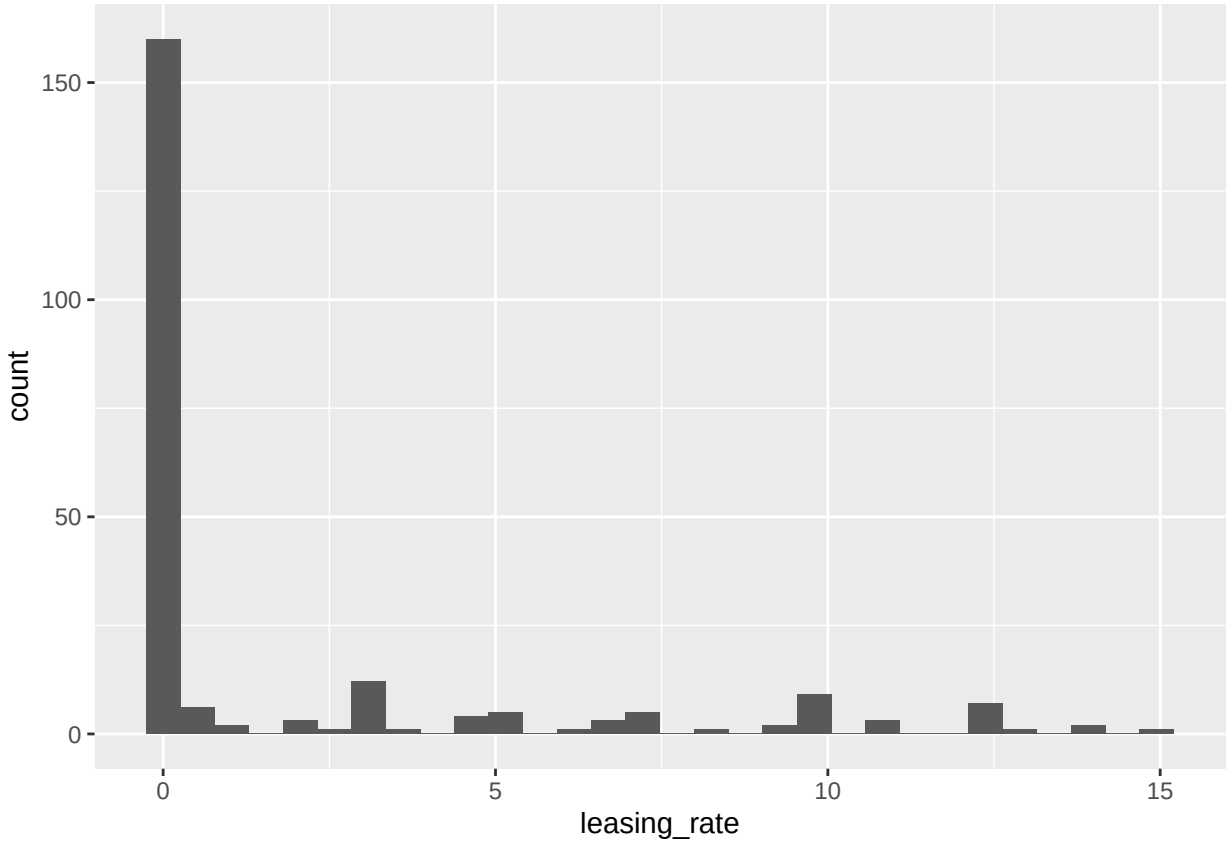
Visual story telling part 1: green buildings

```
## Rows: 7894 Columns: 23
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (23): CS_PropertyID, cluster, size, empl_gr, Rent, leasing_rate, stories...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 6 x 23
##   CS_Proper~1 cluster   size empl_gr  Rent leasi~2 stories   age renov~3 class_a
##         <dbl>   <dbl>  <dbl>   <dbl> <dbl>   <dbl>   <dbl> <dbl>   <dbl>   <dbl>
## 1      379105       1 260300    2.22  38.6    91.4      14    16       0       1
## 2      122151       1  67861    2.22  28.6    87.1       5    27       0       0
## 3      379839       1 164848    2.22  33.3    88.9      13    36       1       0
## 4       94614       1  93372    2.22  35      97.0      13    46       1       0
## 5      379285       1 174307    2.22  40.7    96.6      16     5       0       1
## 6       94765       1 231633    2.22  43.2    92.7      14    20       0       1
## # ... with 13 more variables: class_b <dbl>, LEED <dbl>, Energystar <dbl>,
## #   green_rating <dbl>, net <dbl>, amenities <dbl>, cd_total_07 <dbl>,
## #   hd_total07 <dbl>, total_dd_07 <dbl>, Precipitation <dbl>, Gas_Costs <dbl>,
```

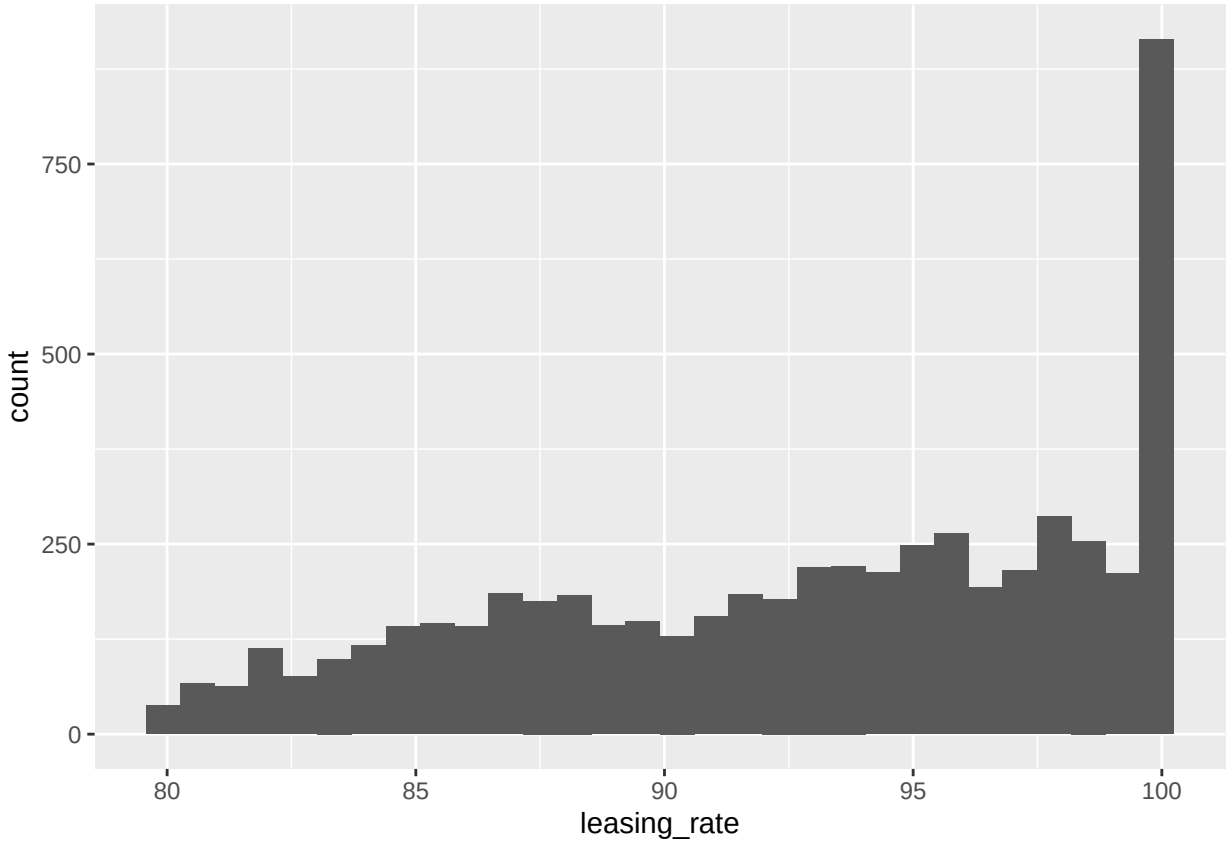
```
## #   Electricity_Costs <dbl>, cluster_rent <dbl>, and abbreviated variable names
## #   1: CS_PropertyID, 2: leasing_rate, 3: renovated
## # i Use `colnames()` to see all variable names
```

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



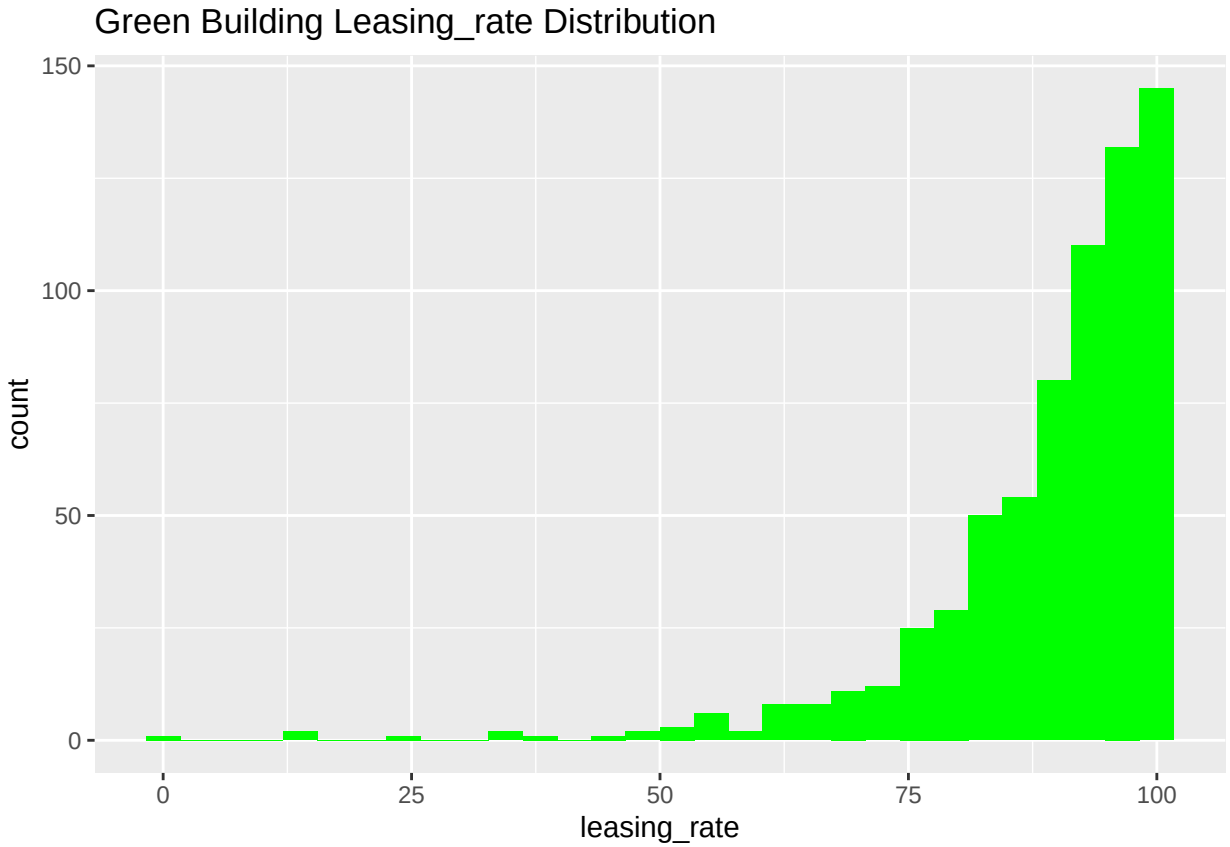
From this plot we know that there are more than 150 properties have 0 % leasing rate, which is abnormal, so I consider them as outlier. However, there exists significant amount of buildings have leasing_rate less than 10%, so dropping them entirely may cause bias.

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

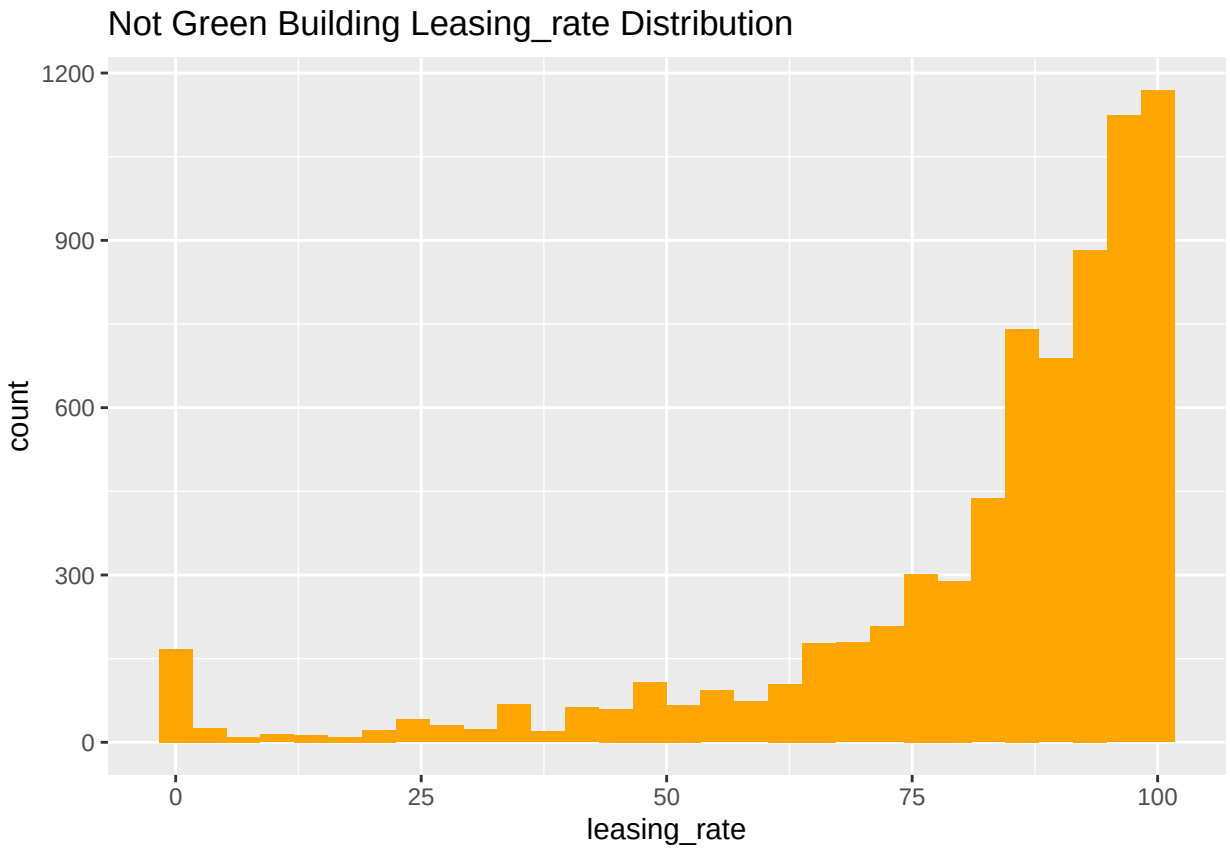


In the above plot, I plotted the distribustion of leasing rate larger than 80%. There are more than 750 of buildings have 100% leasing rate.

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



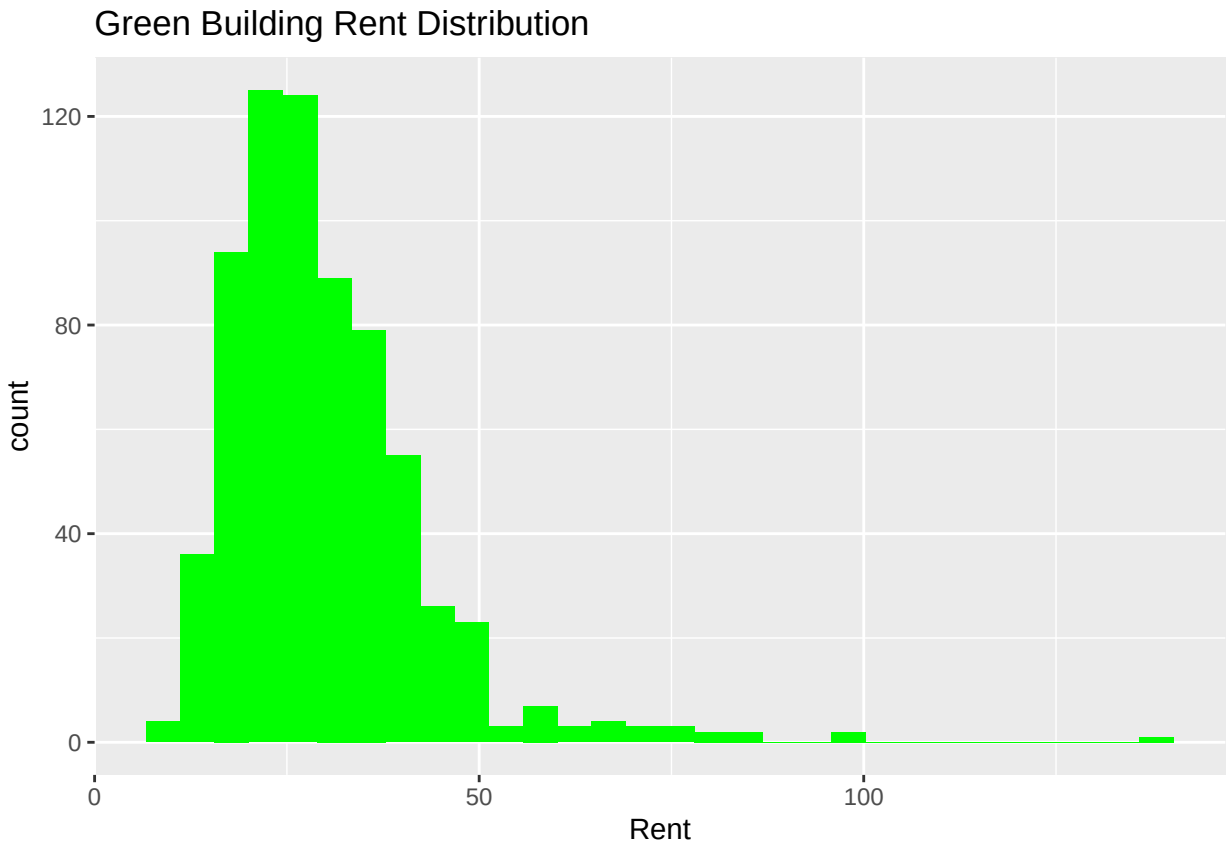

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



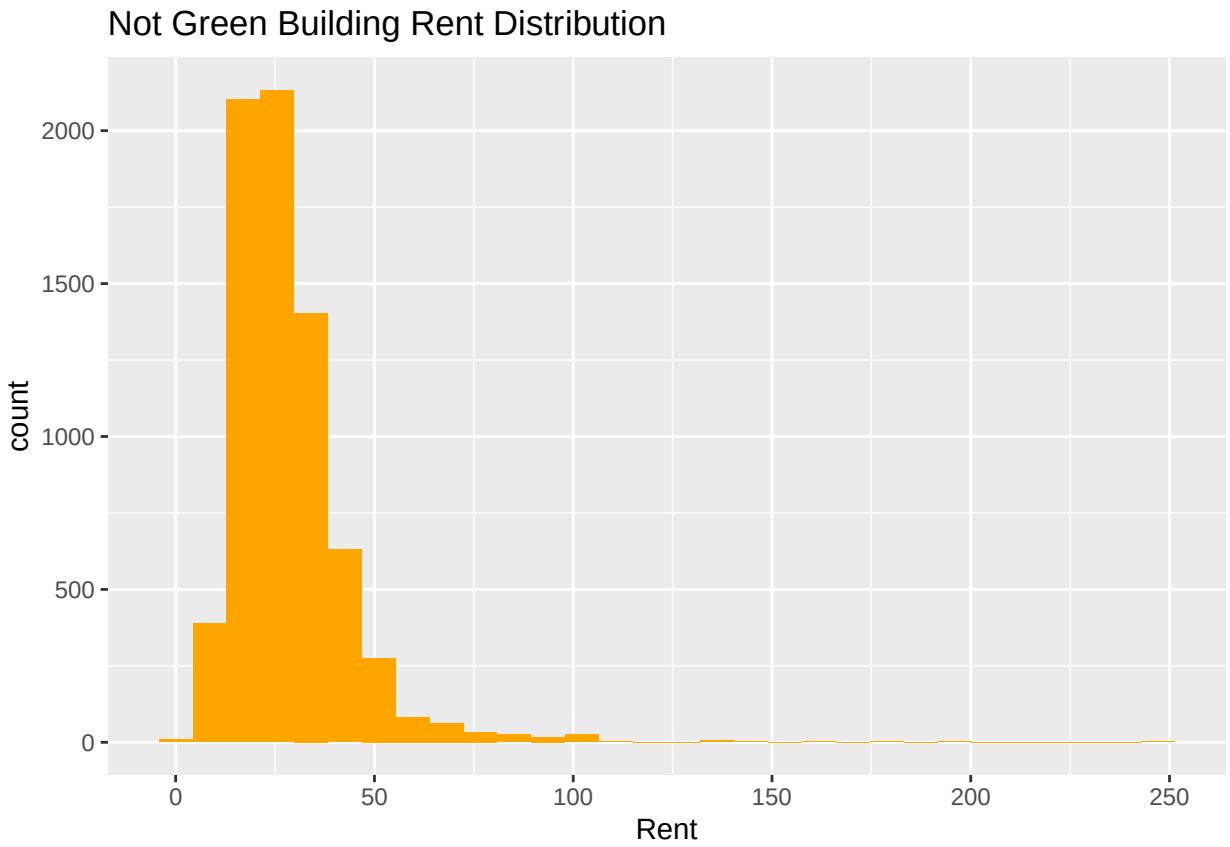
From the plot above, we get the information that most of the 0% leasing_rate comes from buildings that are not green.Then we could say most of the outliers come from not-green buildings.

Then, we take a look at the rent price. We first look at the distribution for both green and non-green buildings.

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

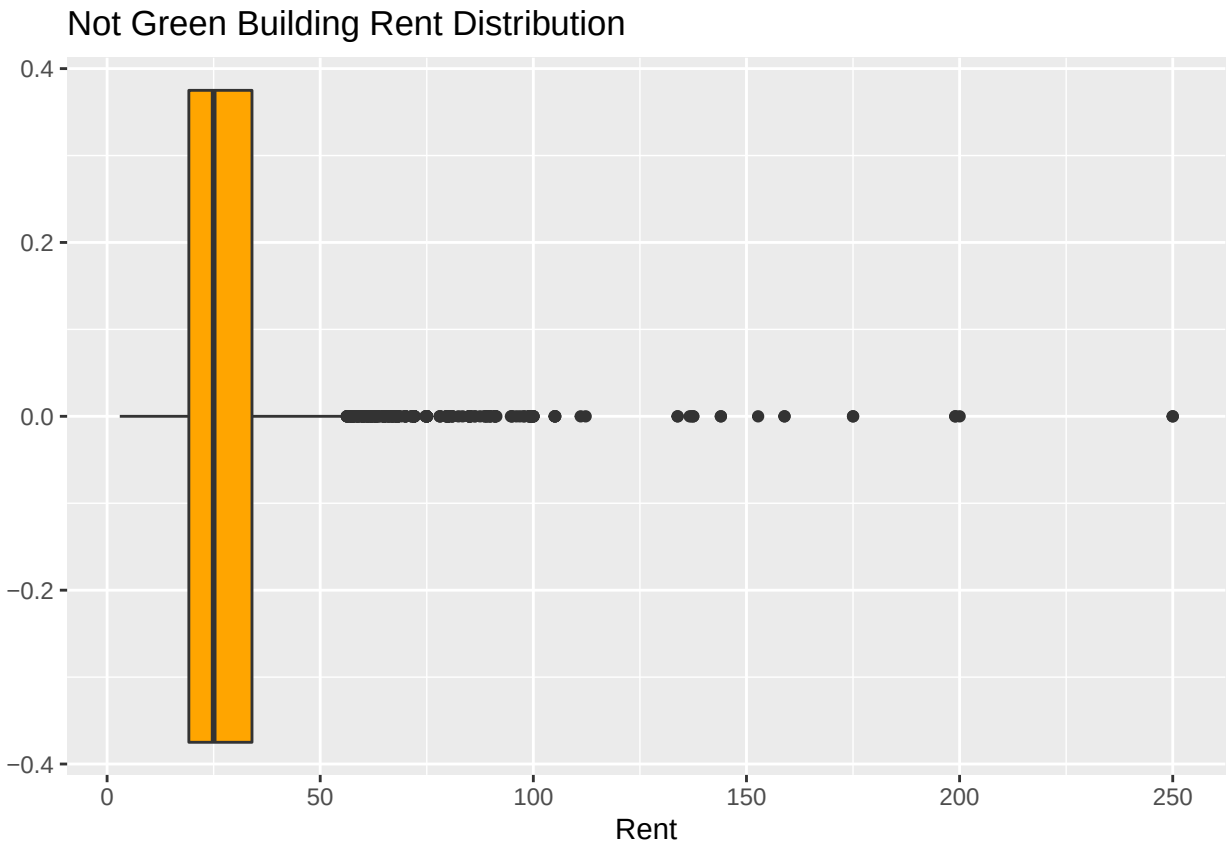



```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



According to the plots above, the non_green buildings have wider price range. Both green and non-green buildings have rent that accumulated around the range 0-100. Also, there exists some buildings that are pricing over 100 for both green and non-green buildings. Guru was right on this one so we proceed to look at me-

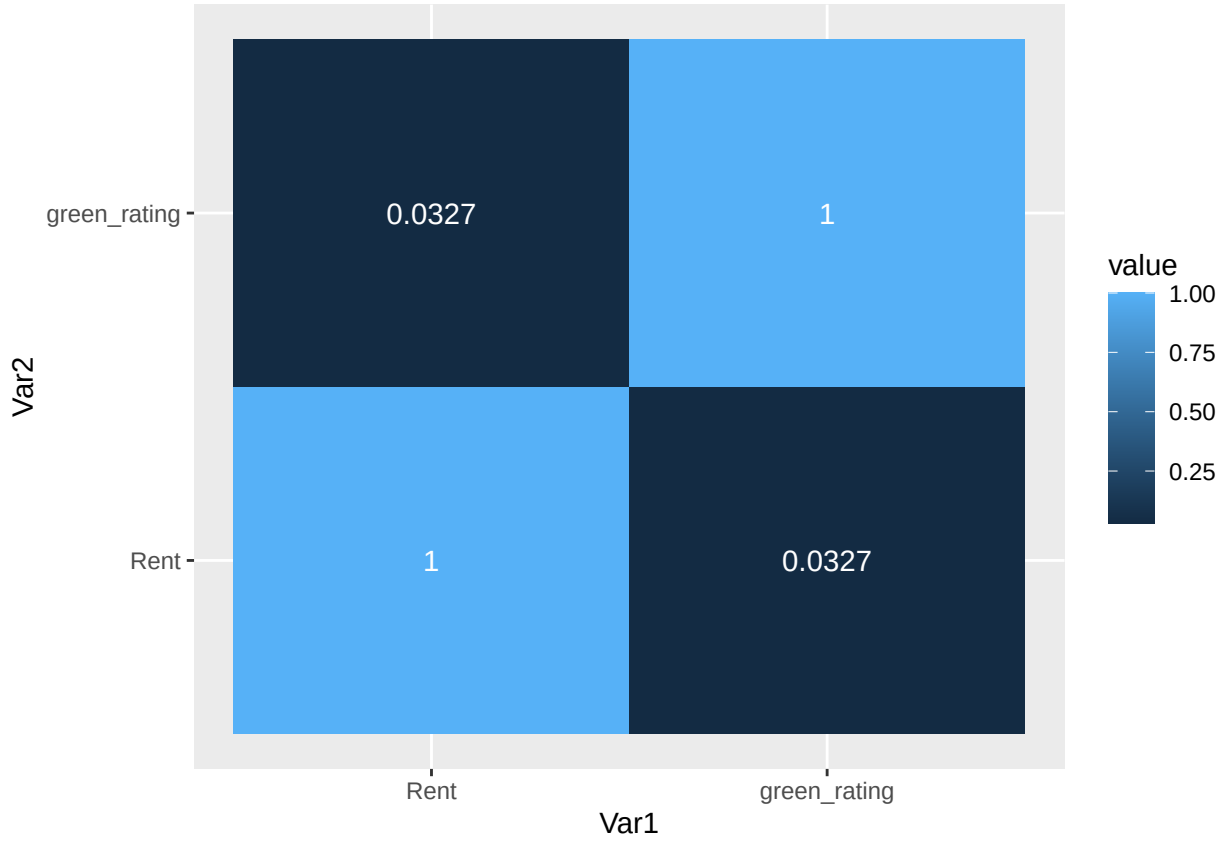
dian rent.




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

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

While green buildings have higher median rent, the non-green buildings have wider distribution range and more high rent outliers. It is very shallow for Guru to claim that green buildings have higher economic values, since other factors, like location, could affect the rent price as well.



According to the correlation heatmap, we can tell that the green_rating has weak correlation with Rent.

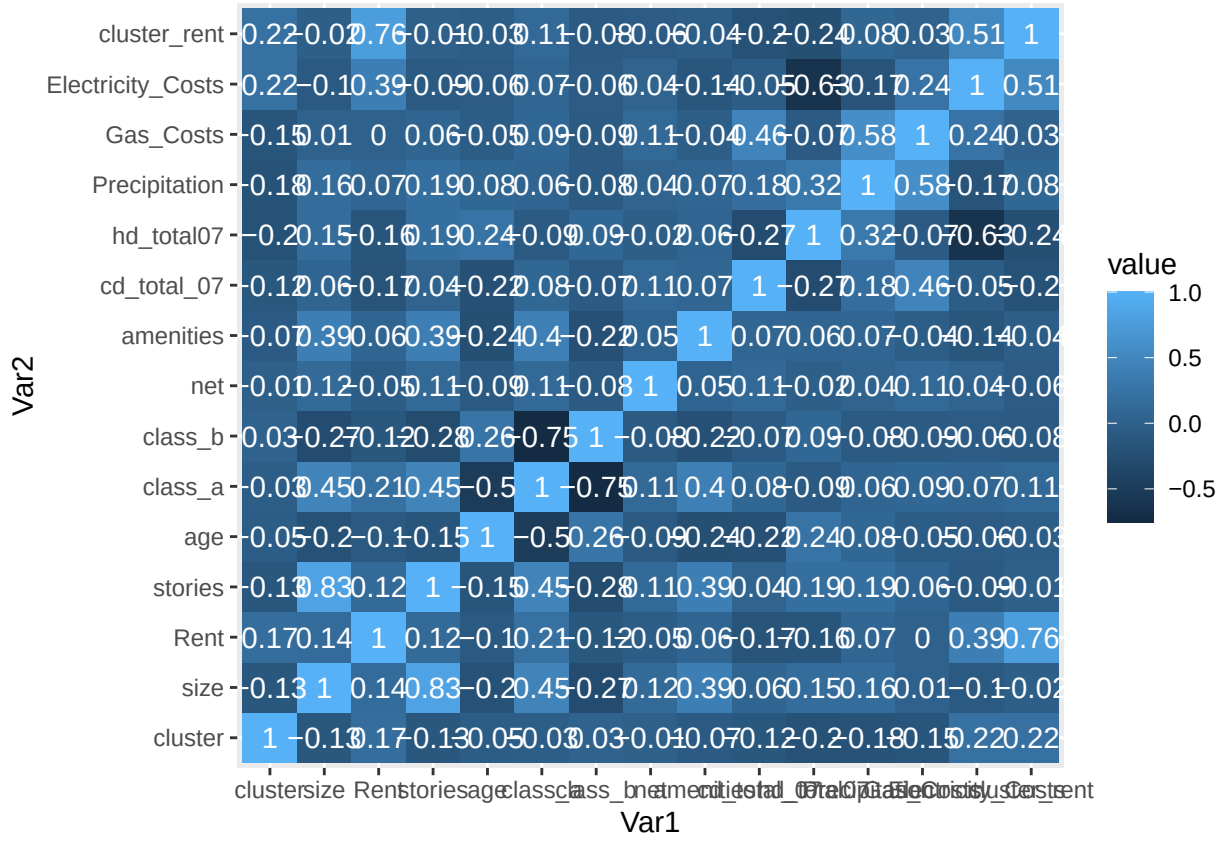
At this point, we know that guru was not right. To find out what affects rent, we need to take further analysis on this.

We further look at linear regression to gain some insights.

```
##
## Call:
## lm(formula = Rent ~ ., data = greenbuildings)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -53.753  -3.581  -0.526   2.491 173.916
##
## Coefficients: (1 not defined because of singularities)
##                    Estimate Std. Error t value Pr(>|t|)
## (Intercept)       -8.315e+00  1.018e+00  -8.167 3.67e-16 ***
## CS_PropertyID      2.959e-07  1.574e-07   1.879 0.060241 .
## cluster            7.532e-04  2.840e-04   2.653 0.008006 **
## size               6.741e-06  6.561e-07  10.276  < 2e-16 ***
## empl_gr            6.450e-02  1.700e-02   3.794 0.000149 ***
## leasing_rate       9.454e-03  5.332e-03   1.773 0.076247 .
## stories           -3.472e-02  1.617e-02  -2.147 0.031823 *
## age               -1.249e-02  4.717e-03  -2.649 0.008096 **
## renovated         -1.425e-01  2.586e-01  -0.551 0.581681
## class_a            2.872e+00  4.377e-01   6.563 5.63e-11 ***
## class_b            1.186e+00  3.427e-01   3.462 0.000539 ***
```

```
## LEED                1.877e+00  3.582e+00   0.524 0.600318
## Energystar         -2.127e-01  3.818e+00  -0.056 0.955572
## green_rating        6.969e-01  3.839e+00   0.182 0.855929
## net                -2.559e+00  5.929e-01  -4.316 1.61e-05 ***
## amenities           6.703e-01  2.519e-01   2.661 0.007802 **
## cd_total_07        -1.248e-04  1.464e-04  -0.852 0.394005
## hd_total07          5.354e-04  8.972e-05   5.967 2.52e-09 ***
## total_dd_07                NA         NA      NA       NA
## Precipitation       4.830e-02  1.611e-02   2.997 0.002735 **
## Gas_Costs          -3.559e+02  7.842e+01  -4.538 5.76e-06 ***
## Electricity_Costs   1.886e+02  2.493e+01   7.563 4.38e-14 ***
## cluster_rent        1.008e+00  1.421e-02  70.949  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 9.413 on 7798 degrees of freedom
##   (74 observations deleted due to missingness)
## Multiple R-squared:  0.6126, Adjusted R-squared:  0.6116
## F-statistic: 587.2 on 21 and 7798 DF,  p-value: < 2.2e-16
```

According to the result above, green_rating's p-val is larger than 0.1, thus, we fail to reject the null hypothesis. The coefficient of green_rating is statistically insignificant, which means green_rating does not play an important role in affecting rent, while cluster rent,electricity_cost, age, cluster, and som other factors have significant relationship with rent. The result of the regression further strengthen our idea that guru's analysis contains flaws. To improve guru's analysis, we could include some of the variables that are statistically significant in linear regression. See the correlation heatmap below.

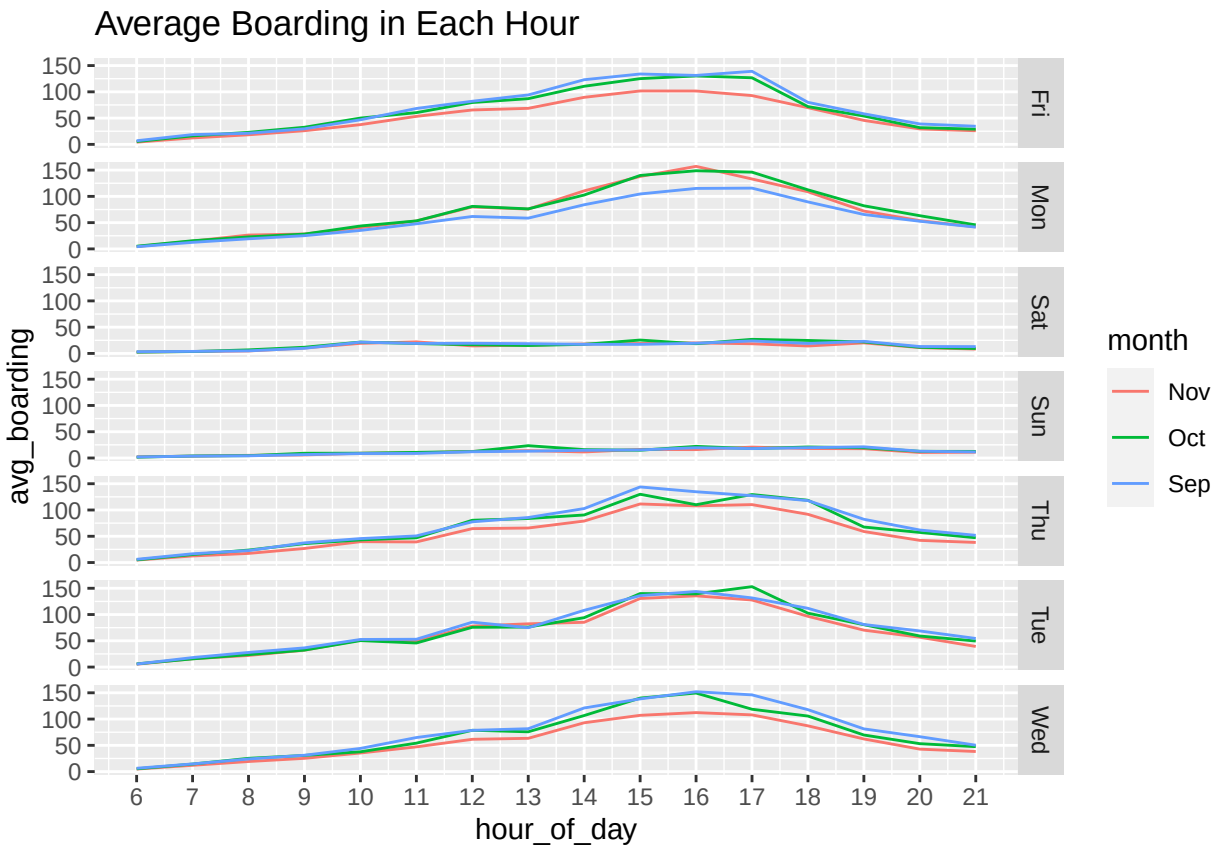


Visual story telling part 2: Capital Metro data

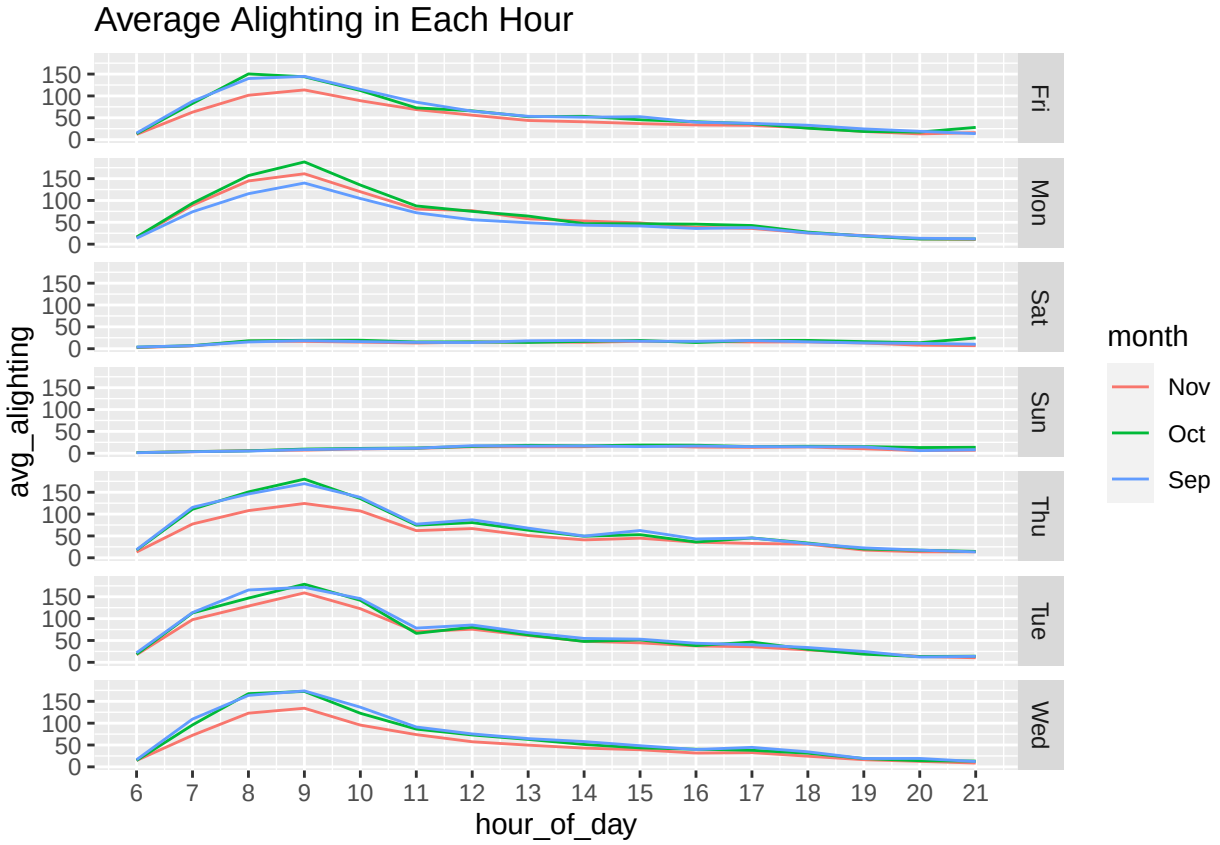
```
## Rows: 5824 Columns: 8
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (3): day_of_week, month, weekend
## dbl  (4): boarding, alighting, temperature, hour_of_day
## dttm (1): timestamp
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 6 x 8
##   timestamp           boarding alighting day_of_~1 tempe~2 hour_~3 month weekend
##   <dttm>                 <dbl>     <dbl> <chr>       <dbl>   <dbl> <chr> <chr>
## 1 2018-09-01 06:00:00        0         1 Sat          74.8       6 Sep   weekend
## 2 2018-09-01 06:15:00        2         1 Sat          74.8       6 Sep   weekend
## 3 2018-09-01 06:30:00        3         4 Sat          74.8       6 Sep   weekend
## 4 2018-09-01 06:45:00        3         4 Sat          74.8       6 Sep   weekend
## 5 2018-09-01 07:00:00        2         4 Sat          74.4       7 Sep   weekend
## 6 2018-09-01 07:15:00        4         4 Sat          74.4       7 Sep   weekend
## # ... with abbreviated variable names 1: day_of_week, 2: temperature,
## #   3: hour_of_day
```

In the plot below, we have our average boarding in each hour of the day, facetting by day_of_week. There are three lines in each facet. Each line represents a month (see legend to learn boarding trend of respective month).



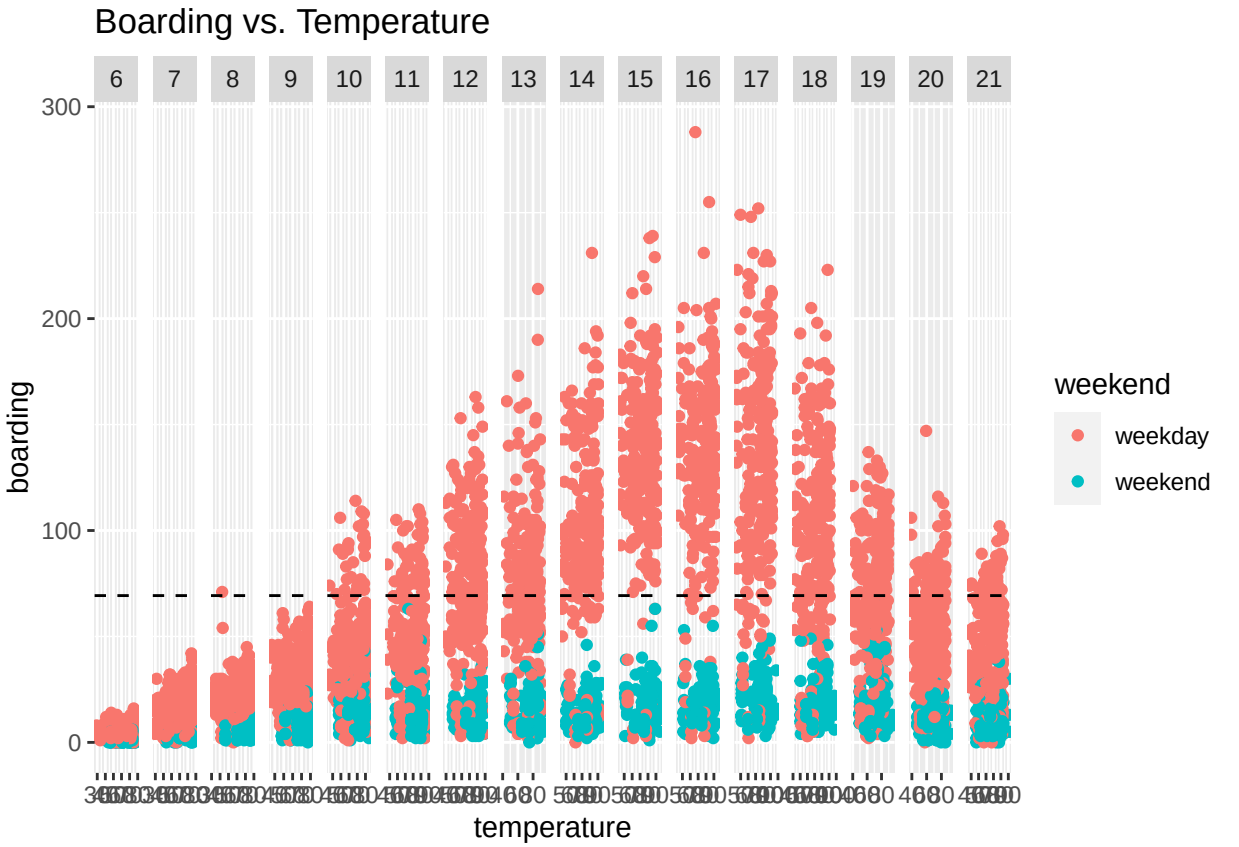
In the above plot, the hour of peak boardings are approximately the same for all weekdays, while weekends do not have a significant hour of peak boardings. The avg_boarding lines are relatively flat in weekends, meaning less traveling to school. The peak boarding hour of weekdays are between 16:00-18:00, illustrating that people tends to take bus/metro to transport back home. The average boarding on Monday in September is slightly lower than the other two months. It could be due to the fact that Labor Day falls on the first Monday of September. November has lower average boarding on Wed, Thur, Fri than other months, since Thanks Giving Holiday starts from the forth Thursday in November to the following Sunday. People tends to travel one day ahead of the holiday, so Wed in November also has lower average boarding.



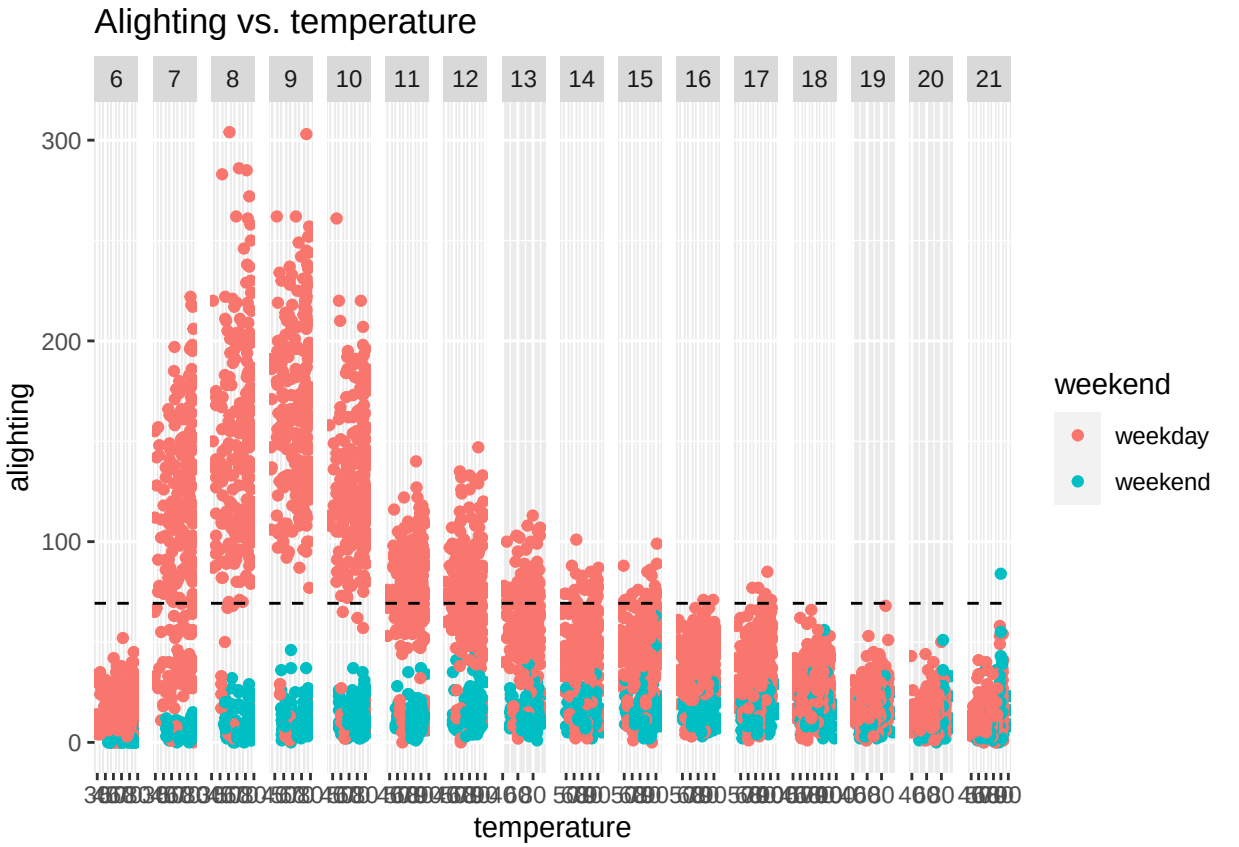
The plot above, we have our average alighting in each hour of the day, facetting by day_of_week. There are three lines in each facet. Each line represents a month (see legend to learn alighting trend of respective month). The avg_alighting plot is approximately a mirror flip of avg_boarding plot.

In the above plot, the hour of peak alignting are approximately the same for all weekdays, while weekends do not have a significant hour of peak alightings. The avg_alighting lines are relatively flat in weekends, meaning less traveling to school. The peak alighting hour of weekdays are between 8:0-10:00, illustrating that it is the morning peak period for people to transport to school/workplace. The average alighting on Monday in September is slightly lower than the other two months for same reason in avg_boarding plot. November has lower average alighting on Wed, Thur, Fri than other months, for identical reason in avg_boarding.

In the plot below, boarding and temperature in each 15-minute window are plotted, facetting by hour_of_day and coloring by weekend/weekday.The horizontal line is the average boarding.



In the plot above, we can tell weekdays tends to have significantly higher boarding amount in all the hours than weekends do. The boarding count is larger during the noon and afternoon time. In general, higher temperature will result in higher boarding count.



In the plot above, alighting and temperature in each 15-minute window are plotted, facetting by hour_of_day and coloring by weekend/weekday.The horizontal line is the average alighting.

We can also tell weekdays tends to have significantly higher alighting amount in all the hours than weekends do. The alighting count is larger during the morning peak time. In general, higher temperature will result in higher boarding count no matter which hour.

Portfolio modeling

Porfolio 1

25% SPY. SPY is the largest ETF and has a huge amount of fluid, which is one of the best ETFs to invest. 25% QQQ. QQQ tracks the Nasdaq-100, an index of the Nasdaq Stock Market's 100 largest nonfinancial members. 25% VUG. VUG concentrates on large-cap U.S. based growth equities. 25% VTI. VTI tracks the CRSP US total market Index's performance. its growth and value styles are appeared in large, mid, and small cap equity.

```
## 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 object is masked from 'package:plyr':
##
##     count

## 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

## 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
```

```
## [1] "SPY" "QQQ" "VUG" "VTI"
```

```
## 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/SPY?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/SPY?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/QQQ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/QQQ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

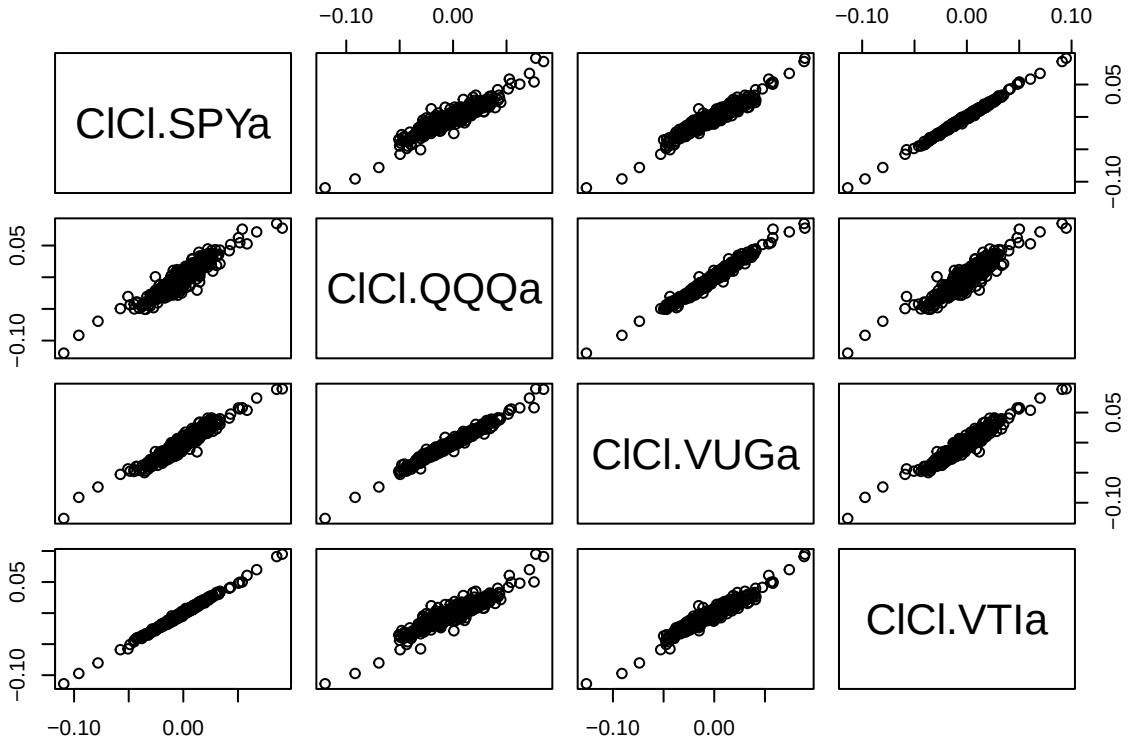
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VUG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/VUG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/VTI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

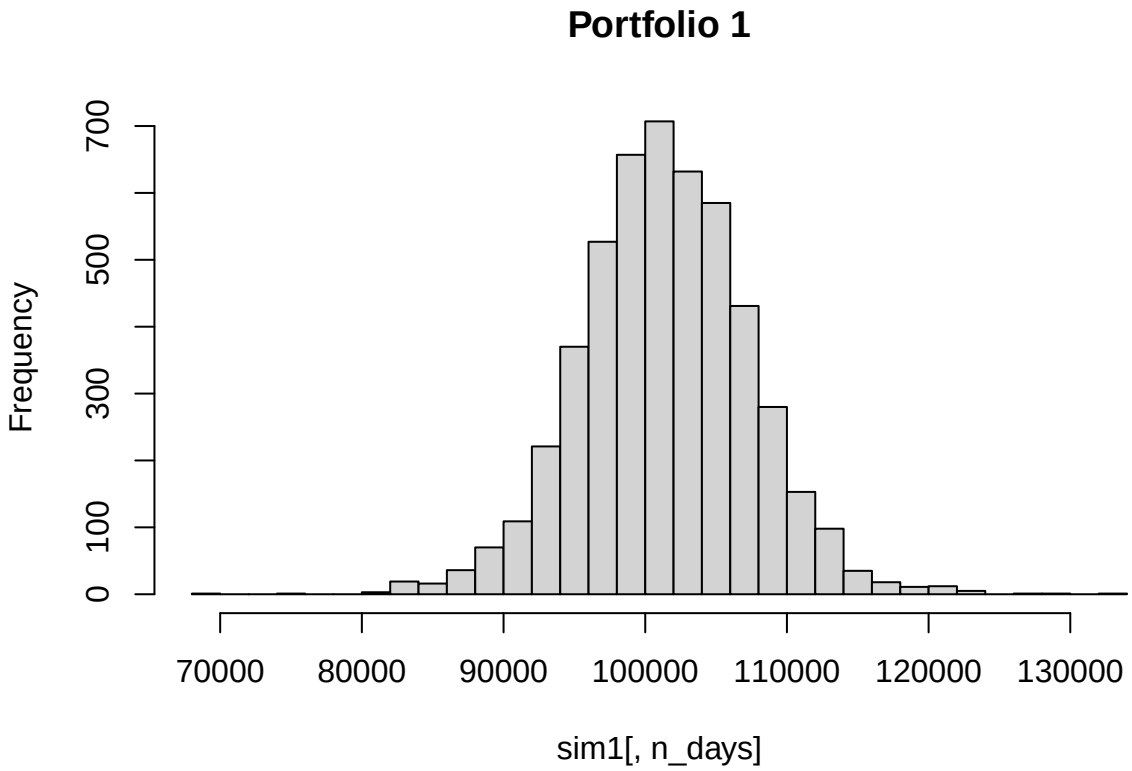
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/VTI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
##            SPY.Open SPY.High  SPY.Low SPY.Close SPY.Volume SPY.Adjusted
## 2016-08-15 196.6976 197.2457 196.6886  196.8773   49813500     196.8772
## 2016-08-16 196.4370 196.5088 195.8619  195.8619   53213600     195.8618
## 2016-08-17 195.8978 196.3741 195.0172  196.2303   75134300     196.2302
## 2016-08-18 196.2033 196.7065 196.0865  196.6706   52989300     196.6706
## 2016-08-19 196.1764 196.5718 195.6642  196.3830   75443000     196.3831
## 2016-08-22 196.1314 196.6167 195.7450  196.3741   61368800     196.3740
```

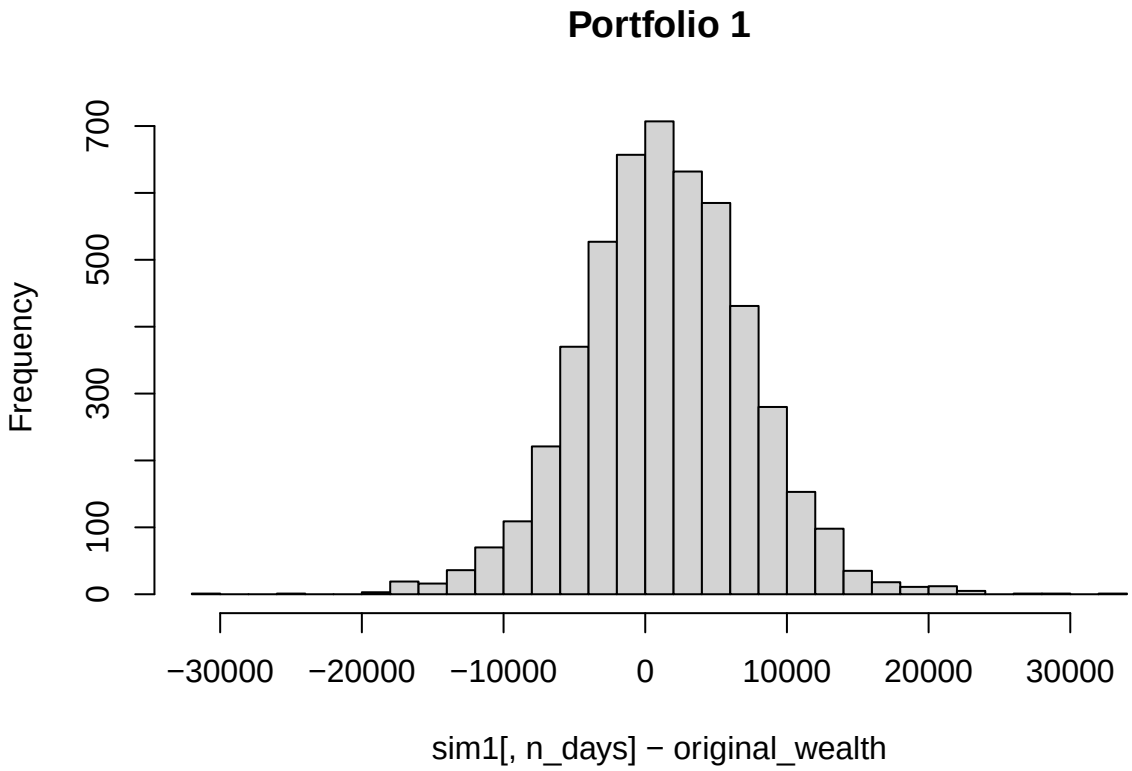


According to the graph, the earnings of these ETFs are highly connected. When one of the ETFs going up, the others will also going up, and when one of the ETFs going down, the others will also going down. This means all ETFs are moving with market trends.

Estimate the 20-day value at risk.


Portfolio 1
Frequency
0
100
300
500
700
70000
80000
90000
100000
110000
120000
130000
sim1[, n_days]

```
## [1] 1402.79
```




```
##        5%
## -8076.904
```

According to the graph, after a 20 day bootstrapped period, there is still a chance of losing. However, there is still 1402.79 mean earnings. 5% Var over a 20 day bootstrapped period is 8076.904.

Portfolio 2

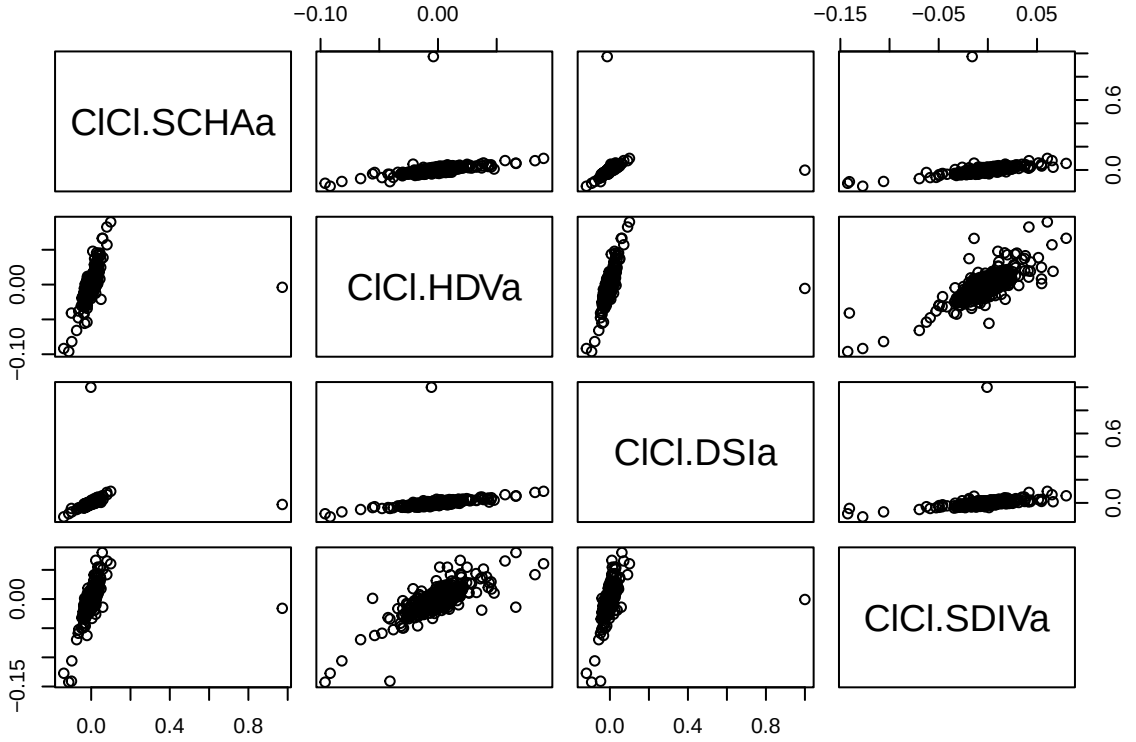
25% SCHA 25% HDV 25% DSI 25% SDIV

```
## 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/SCHA?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/SCHA?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

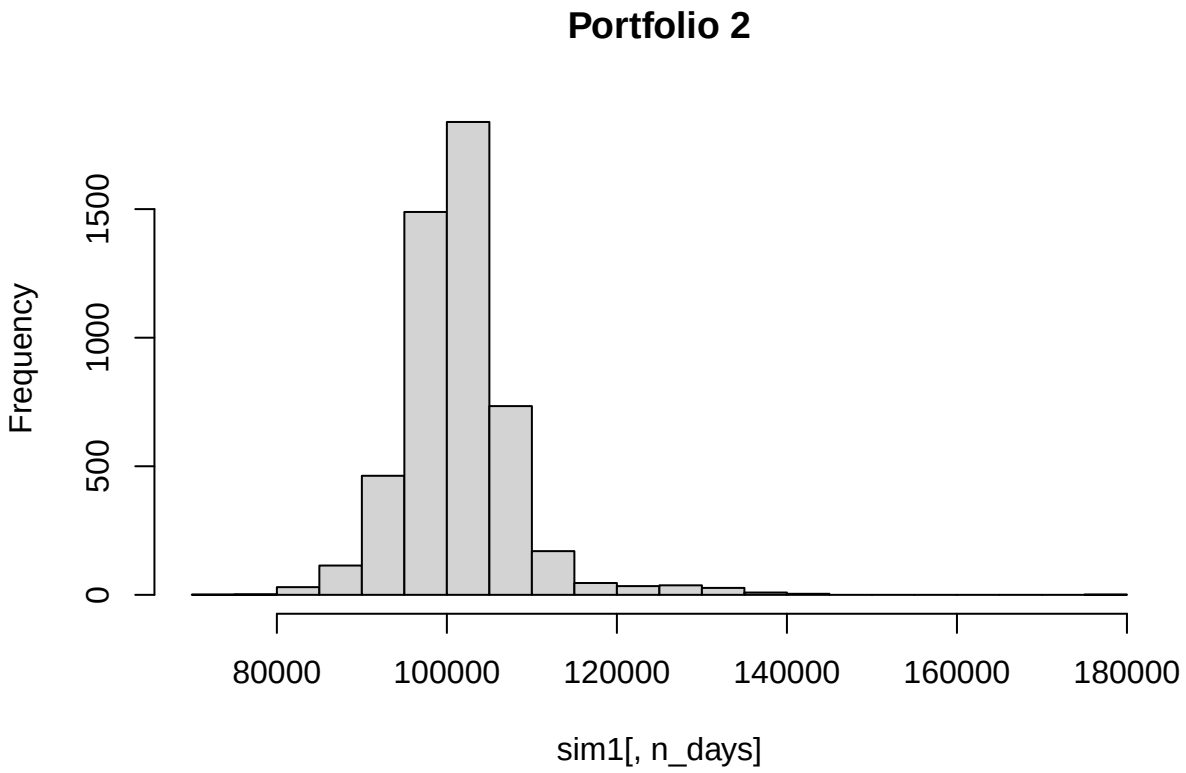
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/HDV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/HDV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/DSI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/DSI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/SDIV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/SDIV?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

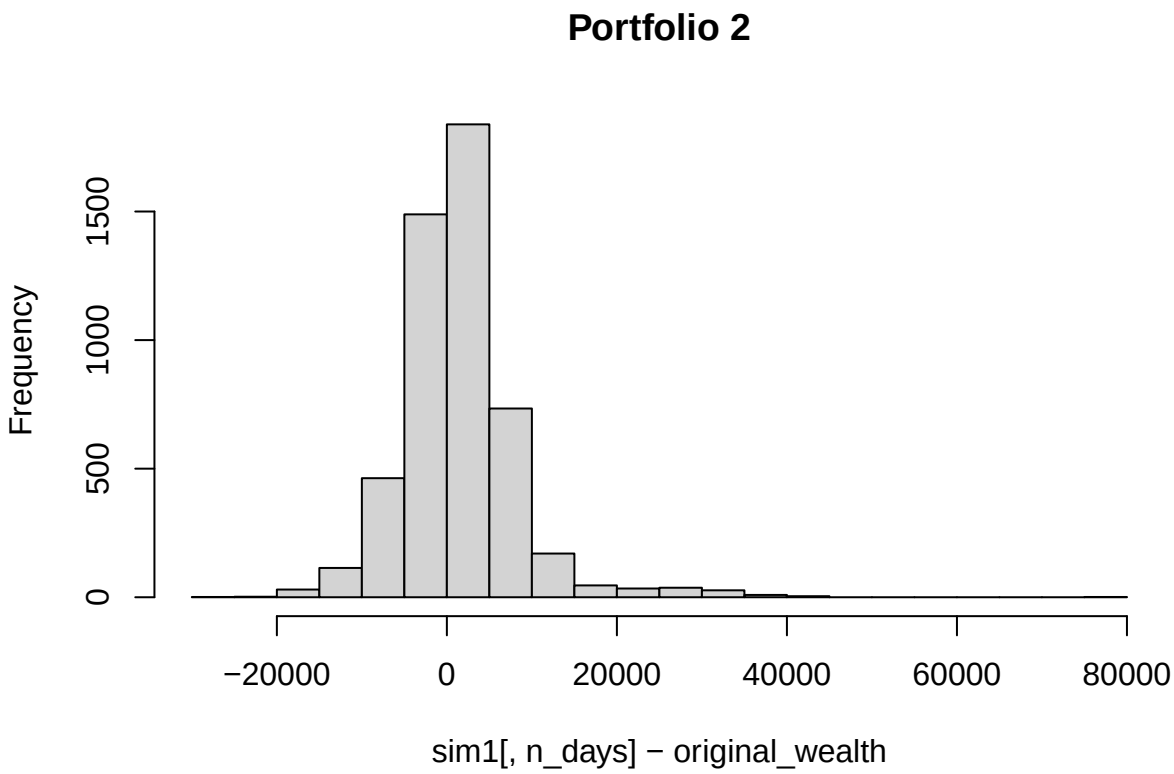


According to the graph, the correlation coefficient of these four ETFs is lower than the first group, which means this group has a higher value risky.

Estimate the 20-day value at risk.




```
## [1] 1425.48
```




```
##        5%
## -7929.719
```

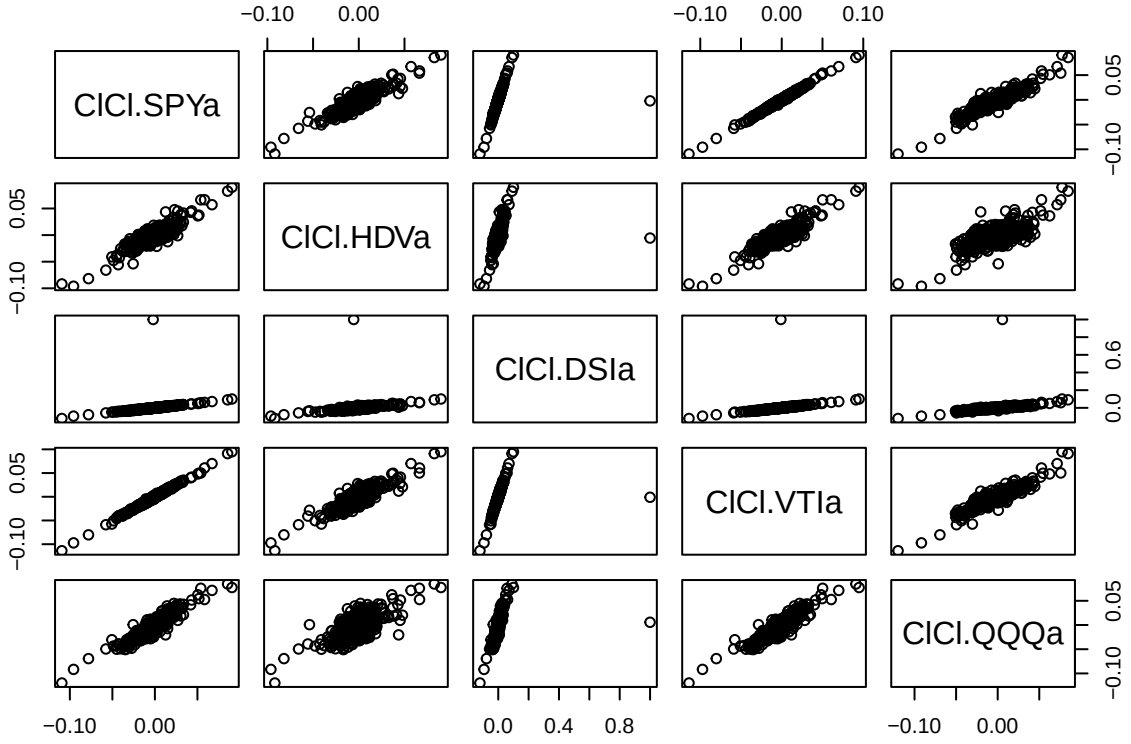
According to the graph, after a 20 day period, there is still a chance of losing. However, there is still 1425.48 mean earnings. 5% Var over a 20 day period is 7929.719.

Portfolio 3

20% SPY 20% HDV 20% DSI 20% VTI 20% QQQ

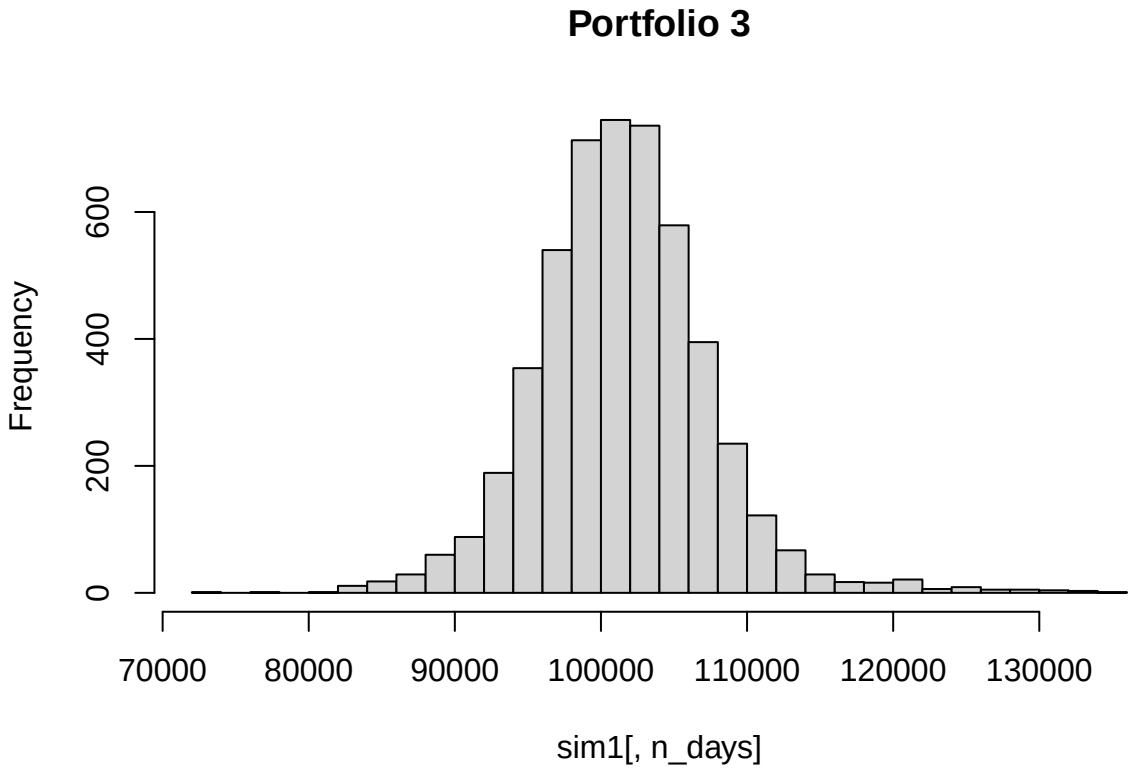
```
## Warning in read.table(file = file, header = header, sep = sep,
## quote = quote, : incomplete final line found by readTableHeader
## on 'https://query1.finance.yahoo.com/v7/finance/download/SPY?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/SPY?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/QQQ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

```
## 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/QQQ?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/VUG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/VUG?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/VTI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'

## 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/VTI?
## period1=-2208988800&period2=1660521600&interval=1d&events=split'
```

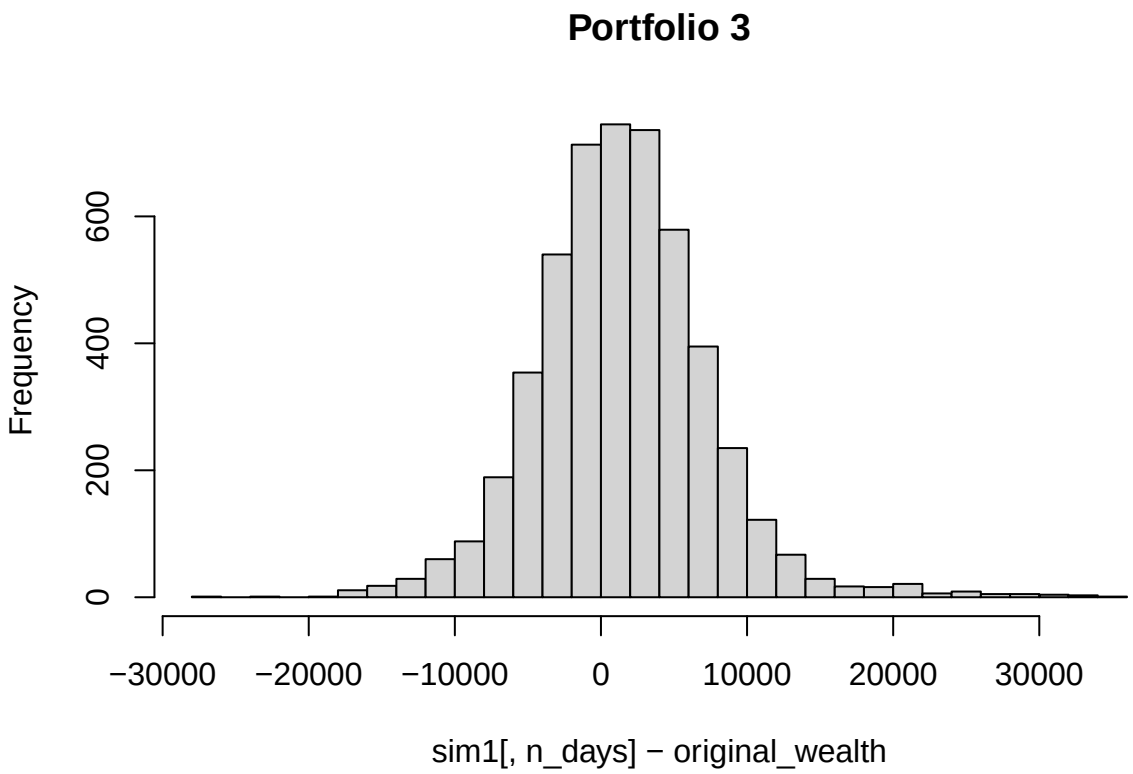


According to the graph, the correlation coefficient of these five ETFs is lower than the first group, which means this group has a higher value risky.

Estimate the 20-day value at risk.




```
## [1] 1494.685
```




```
##        5%
## -7457.022
```

According to the graph, after a 20 day period, there is still a chance of losing. However, there is still 1494.685 mean earnings. 5% Var over a 20 day period is 7457.022.

In Summary: Portfolio 1 is a good long-term investing. The 5% var and the return are the most balanced. Also, it's stable, less risky, and 5% Var over a 20 day bootstrapped period is 8076.904. However, it's mean earning is the lowest. Portfolio 2 and Portfolio 3 are better for the investor who are willing to take higher risks. They have higher mean earnings, 1425.48 and 1494.685. Their 5% Var over a 20 day period are 7929.719 and 7457.022.

Clustering and PCA

```
## Rows: 6497 Columns: 13
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## chr  (1): color
## dbl (12): fixed.acidity, volatile.acidity, citric.acid, residual.sugar, chlo...
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```
## # A tibble: 6 x 13
```

```
##   fixed.~1 volat~2 citri~3 resid~4 chlor~5 free.~6 total~7 density    pH sulph~8
##      <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl> <dbl>   <dbl>
## 1      7.4    0.7     0        1.9   0.076      11      34   0.998  3.51    0.56
## 2      7.8    0.88    0        2.6   0.098      25      67   0.997  3.2     0.68
## 3      7.8    0.76    0.04     2.3   0.092      15      54   0.997  3.26    0.65
## 4     11.2    0.28    0.56     1.9   0.075      17      60   0.998  3.16    0.58
## 5      7.4    0.7     0        1.9   0.076      11      34   0.998  3.51    0.56
## 6      7.4    0.66    0        1.8   0.075      13      40   0.998  3.51    0.56
## # ... with 3 more variables: alcohol <dbl>, quality <dbl>, color <chr>, and
## #   abbreviated variable names 1: fixed.acidity, 2: volatile.acidity,
## #   3: citric.acid, 4: residual.sugar, 5: chlorides, 6: free.sulfur.dioxide,
## #   7: total.sulfur.dioxide, 8: sulphates
## # i Use `colnames()` to see all variable names

## -- Attaching packages --------------------------------------- tidyverse 1.3.2 --
## v tibble  3.1.7     v stringr 1.4.0
## v tidyr   1.2.0     v forcats 0.5.1
## v purrr   0.3.4
## -- Conflicts ------------------------------------------ tidyverse_conflicts() --
## x purrr::accumulate()        masks foreach::accumulate()
## x plyr::arrange()            masks dplyr::arrange()
## x purrr::compact()           masks plyr::compact()
## x mosaic::count()            masks plyr::count(), dplyr::count()
## x purrr::cross()             masks mosaic::cross()
## x mosaic::do()               masks dplyr::do()
## x tidyr::expand()            masks Matrix::expand()
## x plyr::failwith()           masks dplyr::failwith()
## x dplyr::filter()            masks stats::filter()
## x xts::first()               masks dplyr::first()
## x ggstance::geom_errorbarh() masks ggplot2::geom_errorbarh()
## x plyr::id()                 masks dplyr::id()
## x dplyr::lag()               masks stats::lag()
## x xts::last()                masks dplyr::last()
## x plyr::mutate()             masks dplyr::mutate()
## x tidyr::pack()              masks Matrix::pack()
## x plyr::rename()             masks dplyr::rename()
## x mosaic::stat()             masks ggplot2::stat()
## x plyr::summarise()          masks dplyr::summarise()
## x plyr::summarize()          masks dplyr::summarize()
## x mosaic::tally()            masks dplyr::tally()
## x tidyr::unpack()            masks Matrix::unpack()
## x purrr::when()              masks foreach::when()
## Welcome! Want to learn more? See two factoextra-related books at https://goo.gl/ve3WBa
##
## Registered S3 method overwritten by 'GGally':
##   method from
##   +.gg   ggplot2
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
```

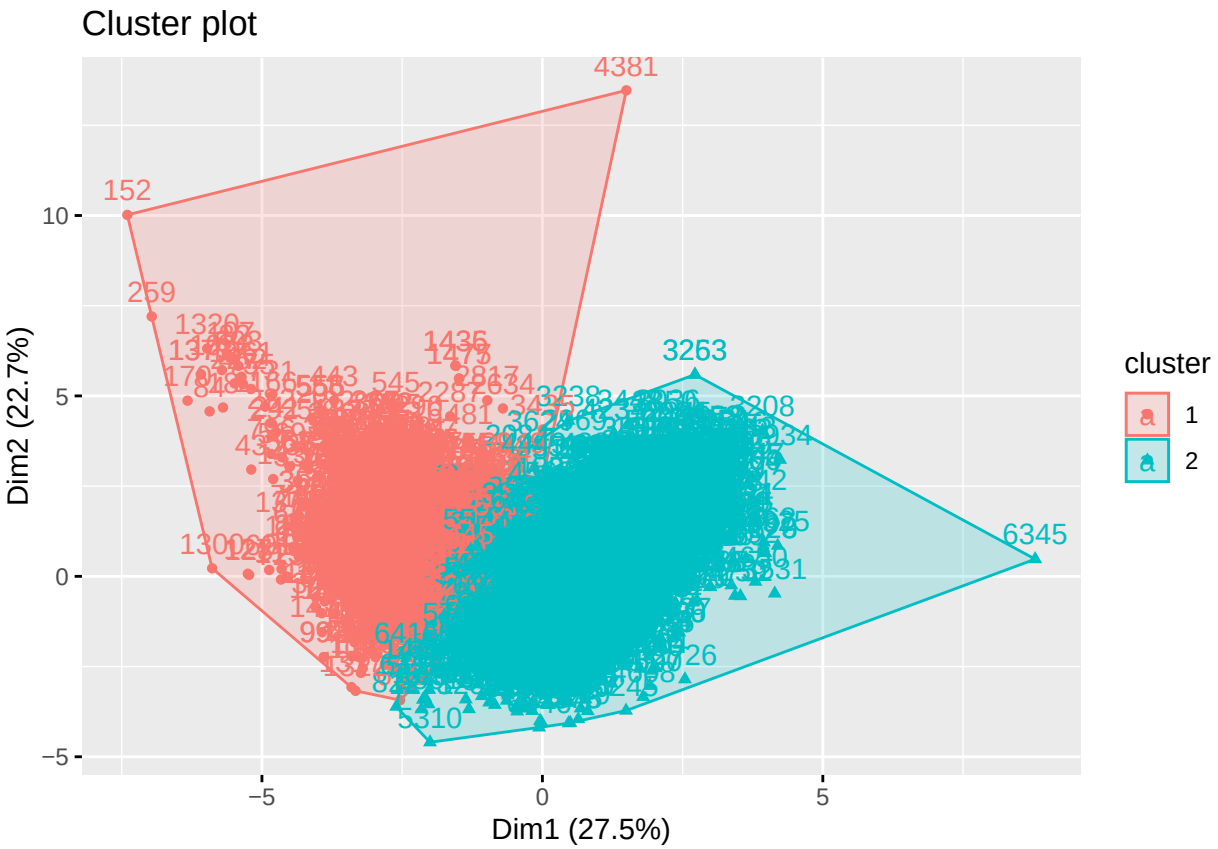
```
##     lift
##
##
## The following object is masked from 'package:mosaic':
##
##     dotPlot
```

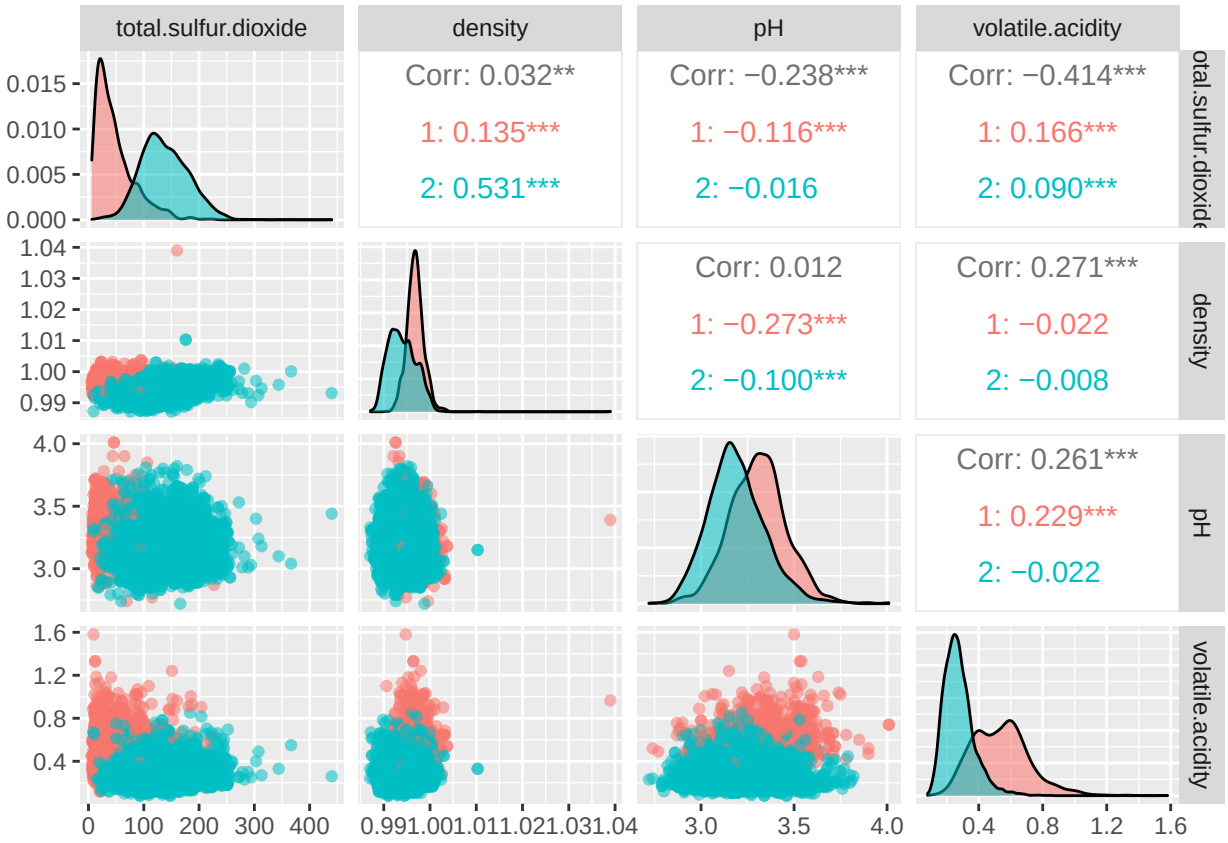
Let's look at K-mean first.

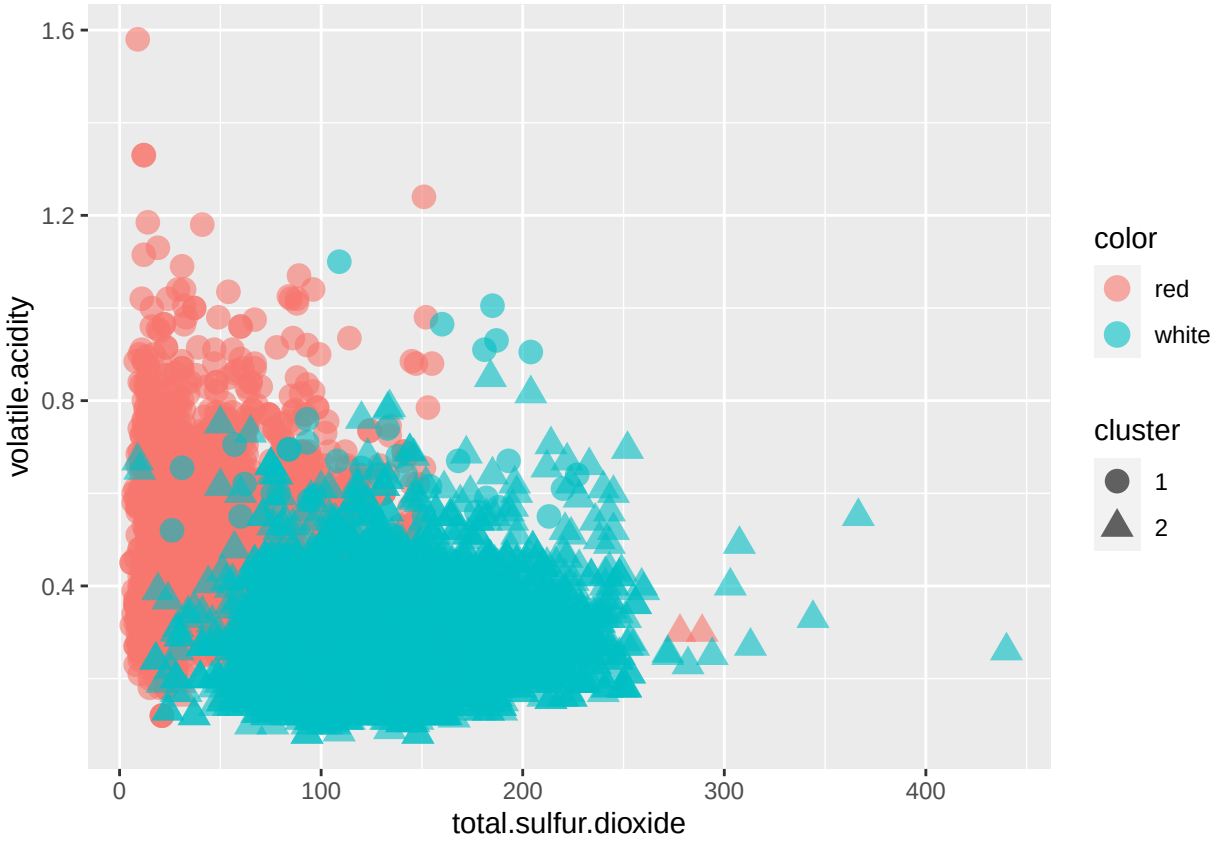
In the code below, we use pam (robust kmean) to get the wine clustering for color. We then compare the kmean clustering result with the real record. True positive: 1525, False Positive:55, False Negative: 74, and True Negative: 4843. The accuracy for kmean in color clustering is 0.9801.

```
## [1] 1580 4917
```

```
## Confusion Matrix and Statistics
##
##                wine1$clr
## pam1$clustering    1    2
##               1 1525   55
##               2   74 4843
##
##                Accuracy : 0.9801
##                  95% CI : (0.9765, 0.9834)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : <2e-16
##
##                   Kappa : 0.9463
##
##  Mcnemar's Test P-Value : 0.113
##
##             Sensitivity : 0.9537
##             Specificity : 0.9888
##          Pos Pred Value : 0.9652
##          Neg Pred Value : 0.9850
##              Prevalence : 0.2461
##          Detection Rate : 0.2347
##    Detection Prevalence : 0.2432
##       Balanced Accuracy : 0.9712
##
##        'Positive' Class : 1
##
```


Cluster plot
4381
152
259
3263
6345
5310
cluster
1
2
Dim2 (22.7%)
Dim1 (27.5%)
10
5
0
−5
−5
0
5


total.sulfur.dioxide
density
pH
volatile.acidity
otal.sulfur.dioxide
density
pH
volatile.acidity
Corr: 0.032**
1: 0.135***
2: 0.531***
Corr: −0.238***
1: −0.116***
2: −0.016
Corr: −0.414***
1: 0.166***
2: 0.090***
Corr: 0.012
1: −0.273***
2: −0.100***
Corr: 0.271***
1: −0.022
2: −0.008
Corr: 0.261***
1: 0.229***
2: −0.022
0.015
0.010
0.005
0.000
1.04
1.03
1.02
1.01
1.00
0.99
4.0
3.5
3.0
1.6
1.2
0.8
0.4
0
100
200
300
400
0.99
1.00
1.01
1.02
1.03
1.04
3.0
3.5
4.0
0.4
0.8
1.2
1.6

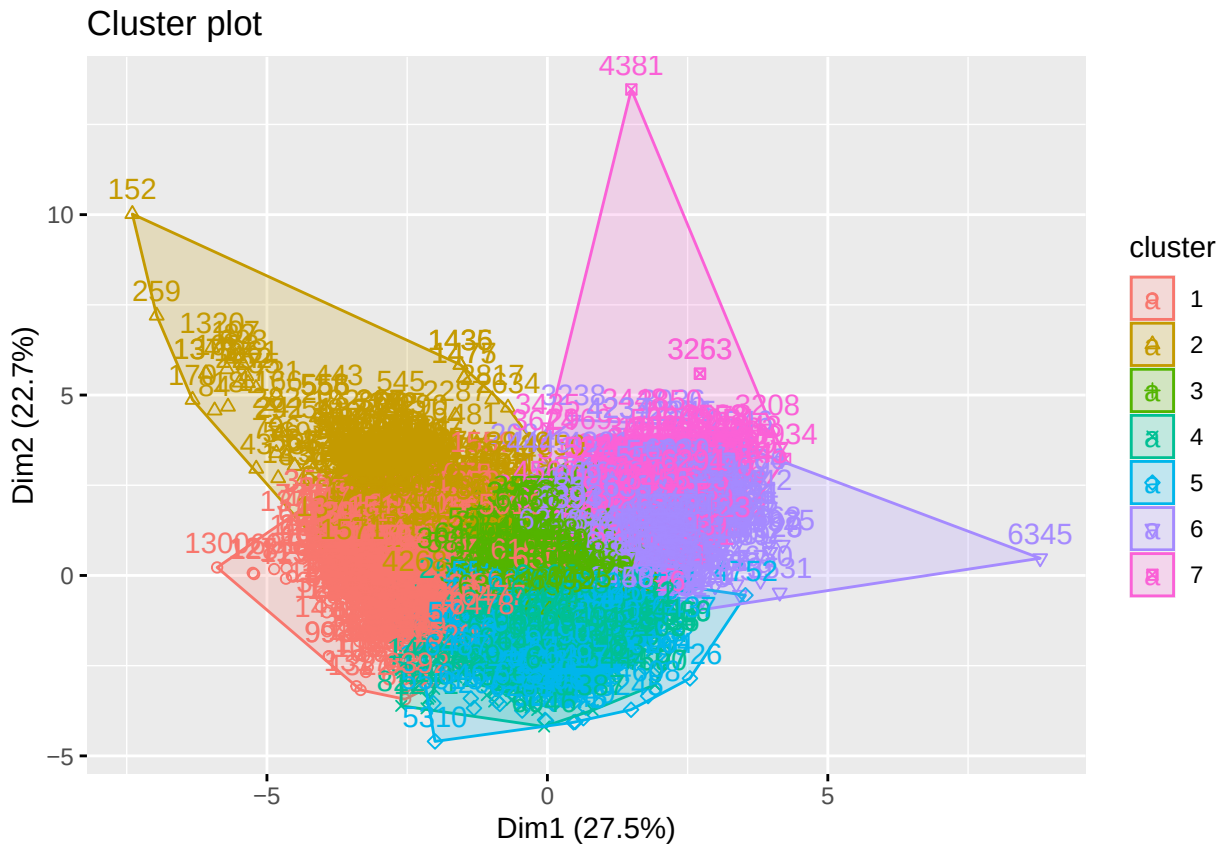


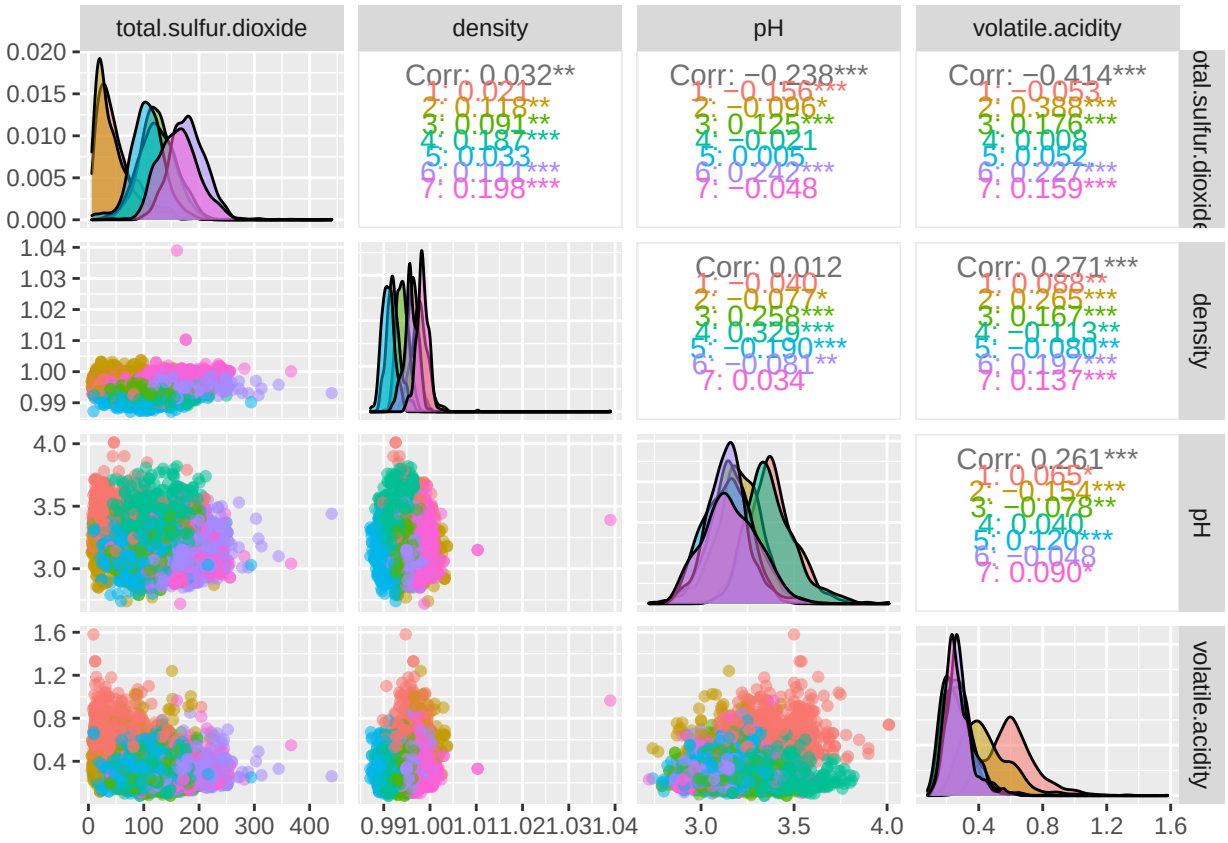
In the code below, we use pam (robust kmean) to get the wine clustering for quality. We then compare the kmean clustering result with the real record. The accuracy is only 0.1973.

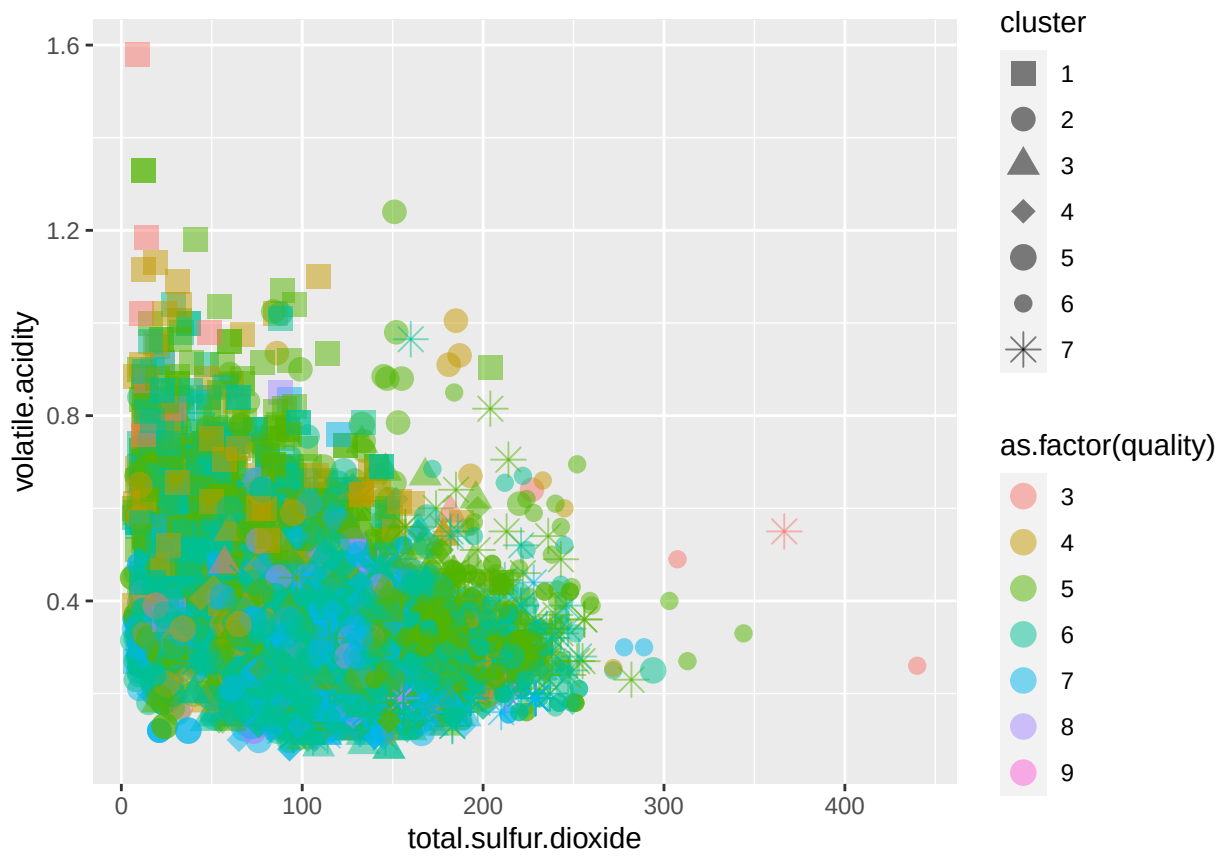
```
## [1]  923  677 1134  827 1174 1017  745
```

```
## Confusion Matrix and Statistics
##
##                wine1$quali
## pam2$clustering   1   2   3   4   5   6   7
##               1   7  62 443 351  55   5   0
##               2   6  20 248 270 122  11   0
##               3   3  67 467 479  99  18   1
##               4   2   9 107 392 267  49   1
##               5   3  27 120 555 387  79   3
##               6   6  18 462 468  54   9   0
##               7   3  13 291 321  95  22   0
##
## Overall Statistics
##
##                Accuracy : 0.1973
##                  95% CI : (0.1877, 0.2072)
##     No Information Rate : 0.4365
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : 0.0536
##
```

```
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                      Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity          0.233333 0.092593  0.21843  0.13822  0.35867 0.046632
## Specificity          0.858358 0.895399  0.84698  0.88118  0.85474 0.840102
## Pos Pred Value       0.007584 0.029542  0.41182  0.47400  0.32964 0.008850
## Neg Pred Value       0.995874 0.966323  0.68842  0.56896  0.87000 0.966423
## Prevalence           0.004618 0.033246  0.32907  0.43651  0.16608 0.029706
## Detection Rate       0.001077 0.003078  0.07188  0.06034  0.05957 0.001385
## Detection Prevalence 0.142066 0.104202  0.17454  0.12729  0.18070 0.156534
## Balanced Accuracy    0.545846 0.493996  0.53271  0.50970  0.60670 0.443367
##                      Class: 7
## Sensitivity          0.0000000
## Specificity          0.8852434
## Pos Pred Value       0.0000000
## Neg Pred Value       0.9991307
## Prevalence           0.0007696
## Detection Rate       0.0000000
## Detection Prevalence 0.1146683
## Balanced Accuracy    0.4426217
```




total.sulfur.dioxide
density
pH
volatile.acidity
otal.sulfur.dioxide
density
pH
volatile.acidity
Corr: 0.032**
1: 0.021
2: 0.118**
3: 0.091**
4: 0.187***
5: 0.033
6: 0.111***
7: 0.198***
Corr: −0.238***
1: −0.156***
2: −0.096*
3: 0.125***
4: −0.021
5: 0.005
6: 0.242***
7: −0.048
Corr: −0.414***
1: −0.053
2: 0.388***
3: 0.176***
4: 0.008
5: 0.052
6: 0.227***
7: 0.159***
Corr: 0.012
1: −0.040
2: −0.077*
3: 0.258***
4: 0.328***
5: −0.190***
6: −0.081**
7: 0.034
Corr: 0.271***
1: 0.088**
2: 0.265***
3: 0.167***
4: −0.113**
5: −0.080**
6: 0.197***
7: 0.137***
Corr: 0.261***
1: 0.065*
2: −0.154***
3: −0.078**
4: 0.040
5: 0.120***
6: −0.048
7: 0.090*
0.020
0.015
0.010
0.005
0.000
1.04
1.03
1.02
1.01
1.00
0.99
4.0
3.5
3.0
1.6
1.2
0.8
0.4
0
100
200
300
400
0.99
1.00
1.01
1.02
1.04
3.0
3.5
4.0
0.4
0.8
1.2
1.6

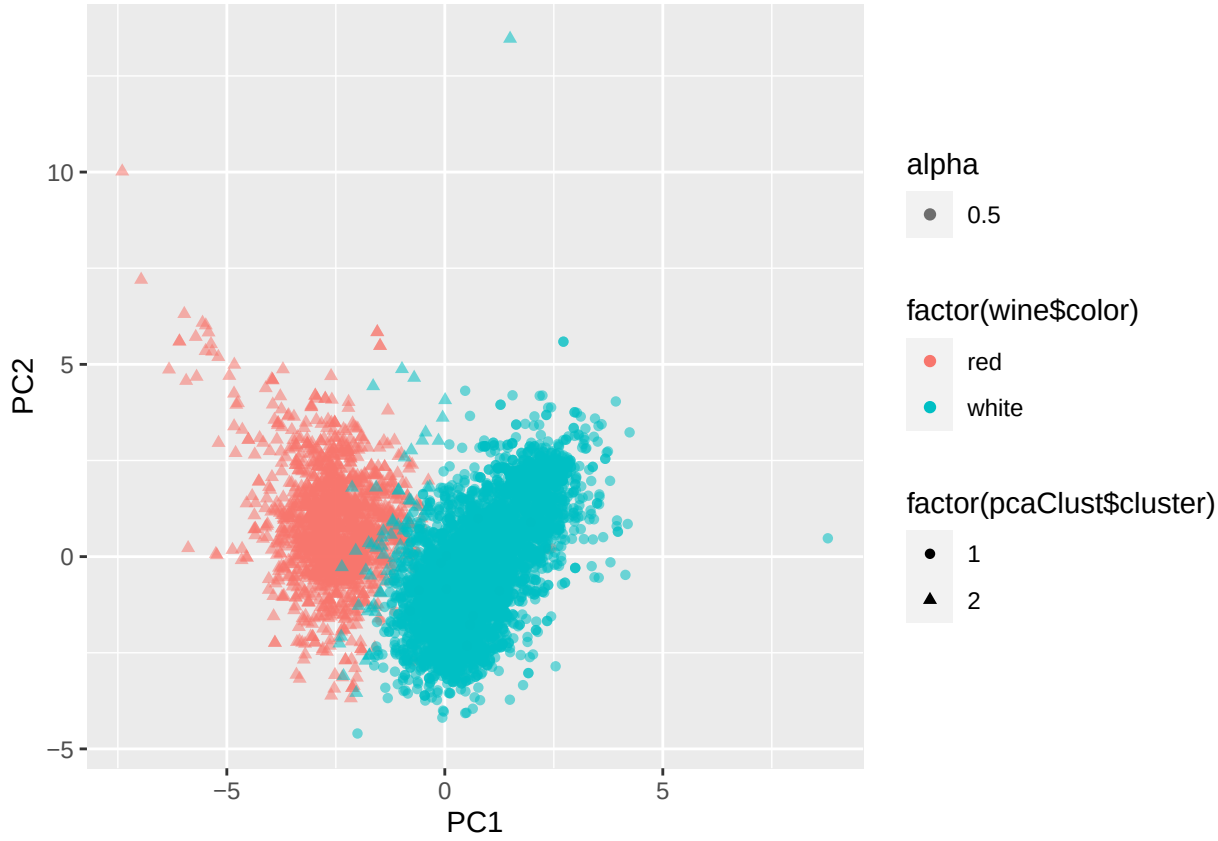


PCA

In the code below, we look into PCA using prcomp. According to the confussion matrix, the accuracy for PCA in color is 0.9854, which is higher than K-mean's accuracy.

```
## Confusion Matrix and Statistics
##
##                  wine1$clr
## pcaClust$cluster    1    2
##                1   24 4827
##                2 1575   71
##
##                Accuracy : 0.0146
##                  95% CI : (0.0118, 0.0178)
##     No Information Rate : 0.7539
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.576
##
##  Mcnemar's Test P-Value : <2e-16
##
##             Sensitivity : 0.015009
##             Specificity : 0.014496
##          Pos Pred Value : 0.004947
##          Neg Pred Value : 0.043135
```

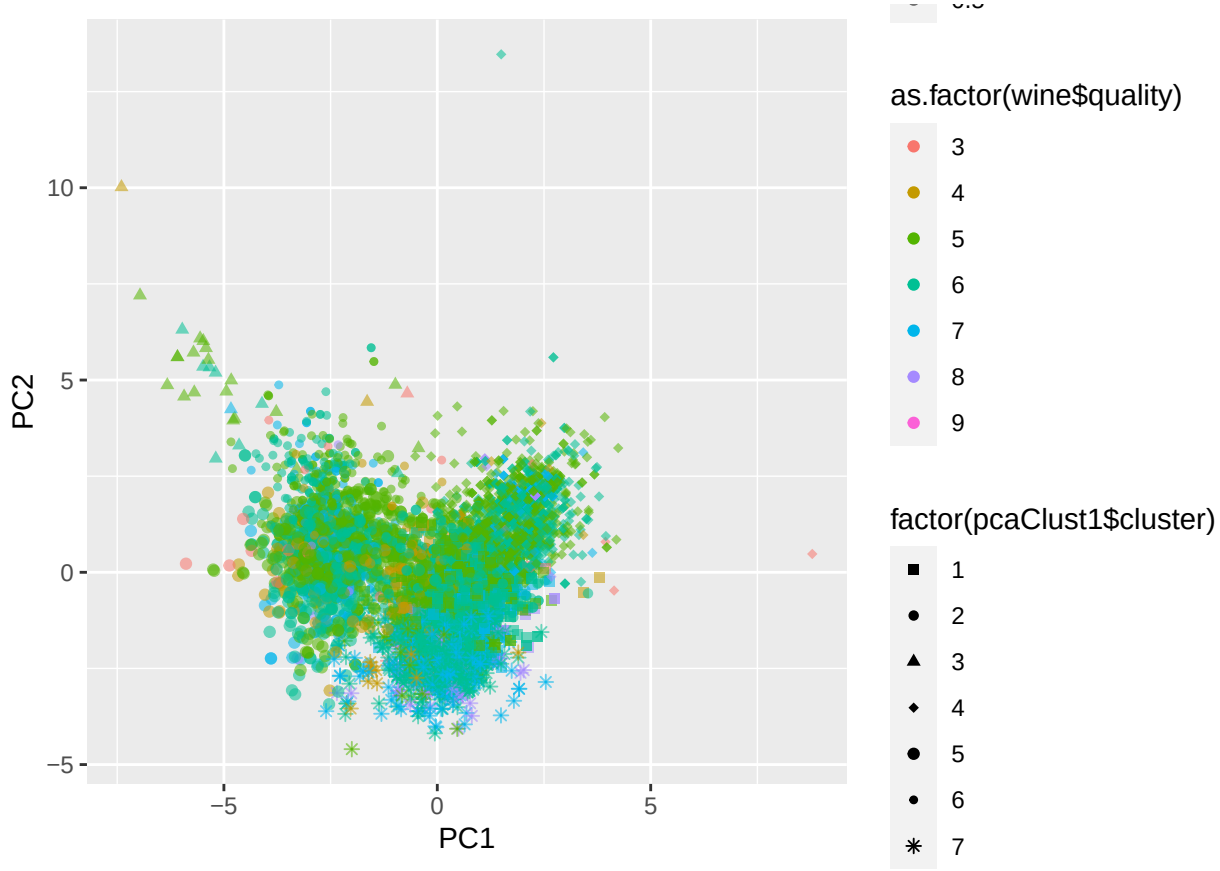
```
##              Prevalence : 0.246114
##          Detection Rate : 0.003694
##    Detection Prevalence : 0.746652
##       Balanced Accuracy : 0.014753
##
##        'Positive' Class : 1
##
```



In the code below, we look into PCA using prcomp. According to the confussion matrix, the accuracy for PCA in quality is 0.1213, which is lower than K-mean's accuracy.

```
## Confusion Matrix and Statistics
##
##                   wine1$quali
## pcaClust1$cluster   1   2   3   4   5   6   7
##                 1   2  28 295 490 192  30   0
##                 2   5  63 410 536 141  26   0
##                 3   1   2  20   9   1   0   0
##                 4   7  24 664 657 122  23   1
##                 5   7  63 470 349  42   2   0
##                 6   4  15 201 266 141  14   0
##                 7   4  21  78 529 440  98   4
##
## Overall Statistics
##
##                Accuracy : 0.1234
##                  95% CI : (0.1155, 0.1317)
```

```
##     No Information Rate : 0.4365
##     P-Value [Acc > NIR] : 1
##
##                   Kappa : -0.0146
##
##  Mcnemar's Test P-Value : <2e-16
##
## Statistics by Class:
##
##                       Class: 1 Class: 2 Class: 3 Class: 4 Class: 5 Class: 6
## Sensitivity          0.0666667 0.291667 0.009355   0.2317 0.038925 0.072539
## Specificity          0.8399567 0.822003 0.997018   0.7703 0.835548 0.900539
## Pos Pred Value       0.0019286 0.053345 0.606061   0.4386 0.045016 0.021841
## Neg Pred Value       0.9948718 0.971219 0.672339   0.5641 0.813623 0.969433
## Prevalence           0.0046175 0.033246 0.329075   0.4365 0.166077 0.029706
## Detection Rate       0.0003078 0.009697 0.003078   0.1011 0.006465 0.002155
## Detection Prevalence 0.1596121 0.181776 0.005079   0.2306 0.143605 0.098661
## Balanced Accuracy    0.4533117 0.556835 0.503186   0.5010 0.437237 0.486539
##                       Class: 7
## Sensitivity          0.8000000
## Specificity          0.8197782
## Pos Pred Value       0.0034072
## Neg Pred Value       0.9998121
## Prevalence           0.0007696
## Detection Rate       0.0006157
## Detection Prevalence 0.1806988
## Balanced Accuracy    0.8098891
```



When distinguishing the color of the wine, PCA did a better job than K-mean but both techniques have relatively high accuracy. When it comes to distinguishing the quality of the wine, K-mean did a slightly better job even though the accuracy is very low. In general, the unsupervised techniques used above is not capable of distinguishing the higher from the lower quality wines.

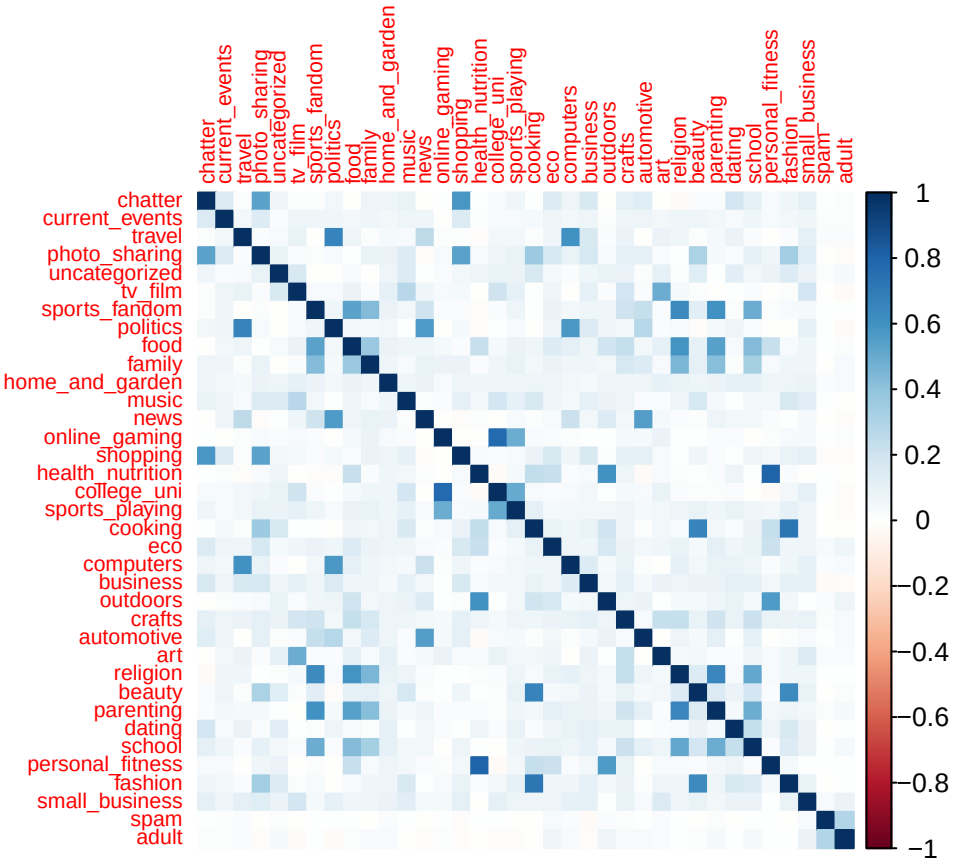
Market segmentation

```
## New names:
## Rows: 7882 Columns: 37
## -- Column specification
## -------------------------------------------------------- Delimiter: "," chr
## (1): ...1 dbl (36): chatter, current_events, travel, photo_sharing,
## uncategorized, tv_...
## i Use `spec()` to retrieve the full column specification for this data. i
## Specify the column types or set `show_col_types = FALSE` to quiet this message.
## corrplot 0.92 loaded
## * `` -> `...1`
```

```
## spec_tbl_df [7,882 x 37] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ ...1            : chr [1:7882] "hmjoe4g3k" "clk1m5w8s" "jcsovtak3" "3oeb4hiln" ...
##  $ chatter         : num [1:7882] 2 3 6 1 5 6 1 5 6 5 ...
##  $ current_events  : num [1:7882] 0 3 3 5 2 4 2 3 2 2 ...
##  $ travel          : num [1:7882] 2 2 4 2 0 2 7 3 0 4 ...
##  $ photo_sharing   : num [1:7882] 2 1 3 2 6 7 1 6 1 4 ...
##  $ uncategorized   : num [1:7882] 2 1 1 0 1 0 0 1 0 0 ...
```

```
##  $ tv_film         : num [1:7882] 1 1 5 1 0 1 1 1 0 5 ...
##  $ sports_fandom   : num [1:7882] 1 4 0 0 0 1 1 1 0 9 ...
##  $ politics        : num [1:7882] 0 1 2 1 2 0 11 0 0 1 ...
##  $ food            : num [1:7882] 4 2 1 0 0 2 1 0 2 5 ...
##  $ family          : num [1:7882] 1 2 1 1 1 1 0 0 2 4 ...
##  $ home_and_garden : num [1:7882] 2 1 1 0 0 1 0 0 1 0 ...
##  $ music           : num [1:7882] 0 0 1 0 0 1 0 2 1 1 ...
##  $ news            : num [1:7882] 0 0 1 0 0 0 1 0 0 0 ...
##  $ online_gaming   : num [1:7882] 0 0 0 0 3 0 0 1 2 1 ...
##  $ shopping        : num [1:7882] 1 0 2 0 2 5 1 3 0 0 ...
##  $ health_nutrition: num [1:7882] 17 0 0 0 0 0 1 1 22 7 ...
##  $ college_uni     : num [1:7882] 0 0 0 1 4 0 1 0 1 4 ...
##  $ sports_playing  : num [1:7882] 2 1 0 0 0 0 1 0 0 1 ...
##  $ cooking         : num [1:7882] 5 0 2 0 1 0 1 10 5 4 ...
##  $ eco             : num [1:7882] 1 0 1 0 0 0 0 0 2 1 ...
##  $ computers       : num [1:7882] 1 0 0 0 1 1 1 1 1 2 ...
##  $ business        : num [1:7882] 0 1 0 1 0 1 3 0 1 0 ...
##  $ outdoors        : num [1:7882] 2 0 0 0 1 0 1 0 3 0 ...
##  $ crafts          : num [1:7882] 1 2 2 3 0 0 0 1 0 0 ...
##  $ automotive      : num [1:7882] 0 0 0 0 0 1 0 1 0 4 ...
##  $ art             : num [1:7882] 0 0 8 2 0 0 1 0 1 0 ...
##  $ religion        : num [1:7882] 1 0 0 0 0 0 1 0 0 13 ...
##  $ beauty          : num [1:7882] 0 0 1 1 0 0 0 5 5 1 ...
##  $ parenting       : num [1:7882] 1 0 0 0 0 0 0 1 0 3 ...
##  $ dating          : num [1:7882] 1 1 1 0 0 0 0 0 0 0 ...
##  $ school          : num [1:7882] 0 4 0 0 0 0 0 0 1 3 ...
##  $ personal_fitness: num [1:7882] 11 0 0 0 0 0 0 0 12 2 ...
##  $ fashion         : num [1:7882] 0 0 1 0 0 0 0 4 3 1 ...
##  $ small_business  : num [1:7882] 0 0 0 0 1 0 0 0 1 0 ...
##  $ spam            : num [1:7882] 0 0 0 0 0 0 0 0 0 0 ...
##  $ adult           : num [1:7882] 0 0 0 0 0 0 0 0 0 0 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   ...1 = col_character(),
##   ..   chatter = col_double(),
##   ..   current_events = col_double(),
##   ..   travel = col_double(),
##   ..   photo_sharing = col_double(),
##   ..   uncategorized = col_double(),
##   ..   tv_film = col_double(),
##   ..   sports_fandom = col_double(),
##   ..   politics = col_double(),
##   ..   food = col_double(),
##   ..   family = col_double(),
##   ..   home_and_garden = col_double(),
##   ..   music = col_double(),
##   ..   news = col_double(),
##   ..   online_gaming = col_double(),
##   ..   shopping = col_double(),
##   ..   health_nutrition = col_double(),
##   ..   college_uni = col_double(),
##   ..   sports_playing = col_double(),
##   ..   cooking = col_double(),
##   ..   eco = col_double(),
```

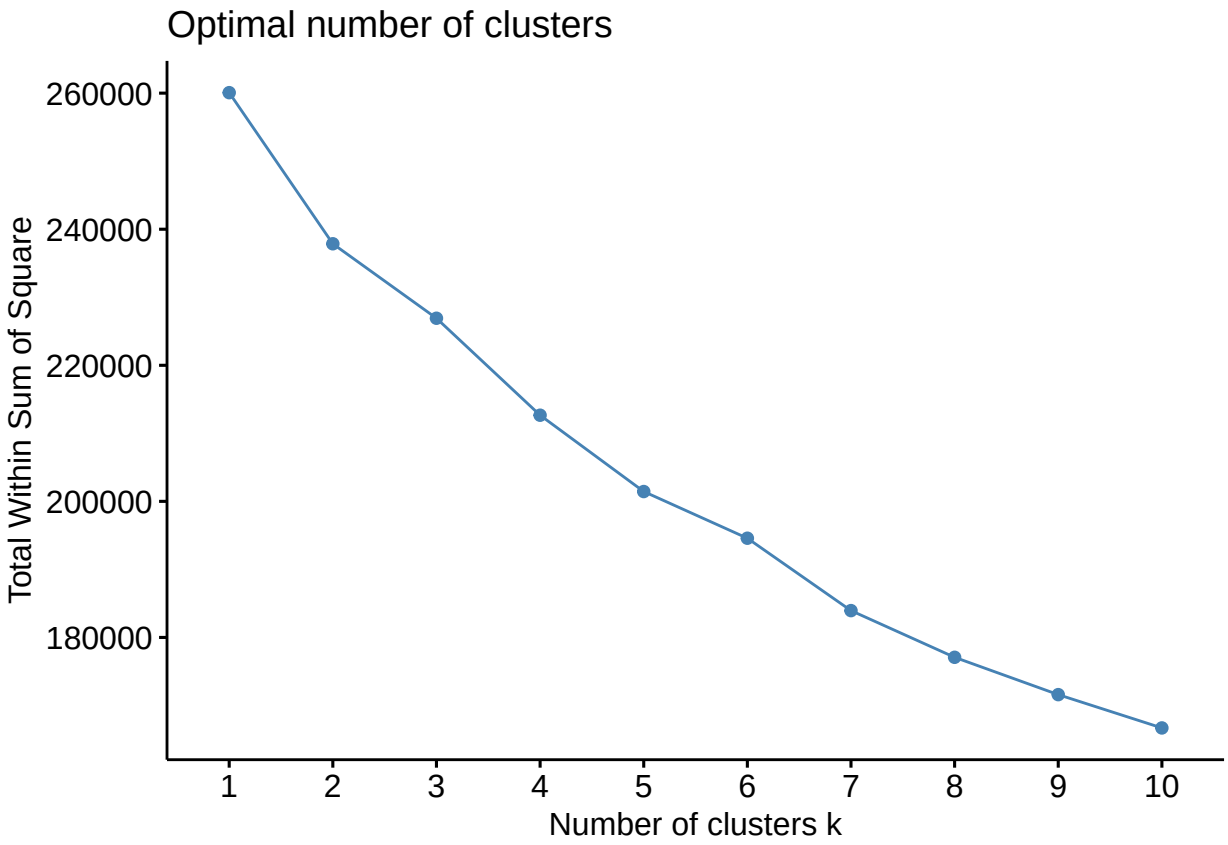
```
##   ..   computers = col_double(),
##   ..   business = col_double(),
##   ..   outdoors = col_double(),
##   ..   crafts = col_double(),
##   ..   automotive = col_double(),
##   ..   art = col_double(),
##   ..   religion = col_double(),
##   ..   beauty = col_double(),
##   ..   parenting = col_double(),
##   ..   dating = col_double(),
##   ..   school = col_double(),
##   ..   personal_fitness = col_double(),
##   ..   fashion = col_double(),
##   ..   small_business = col_double(),
##   ..   spam = col_double(),
##   ..   adult = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```



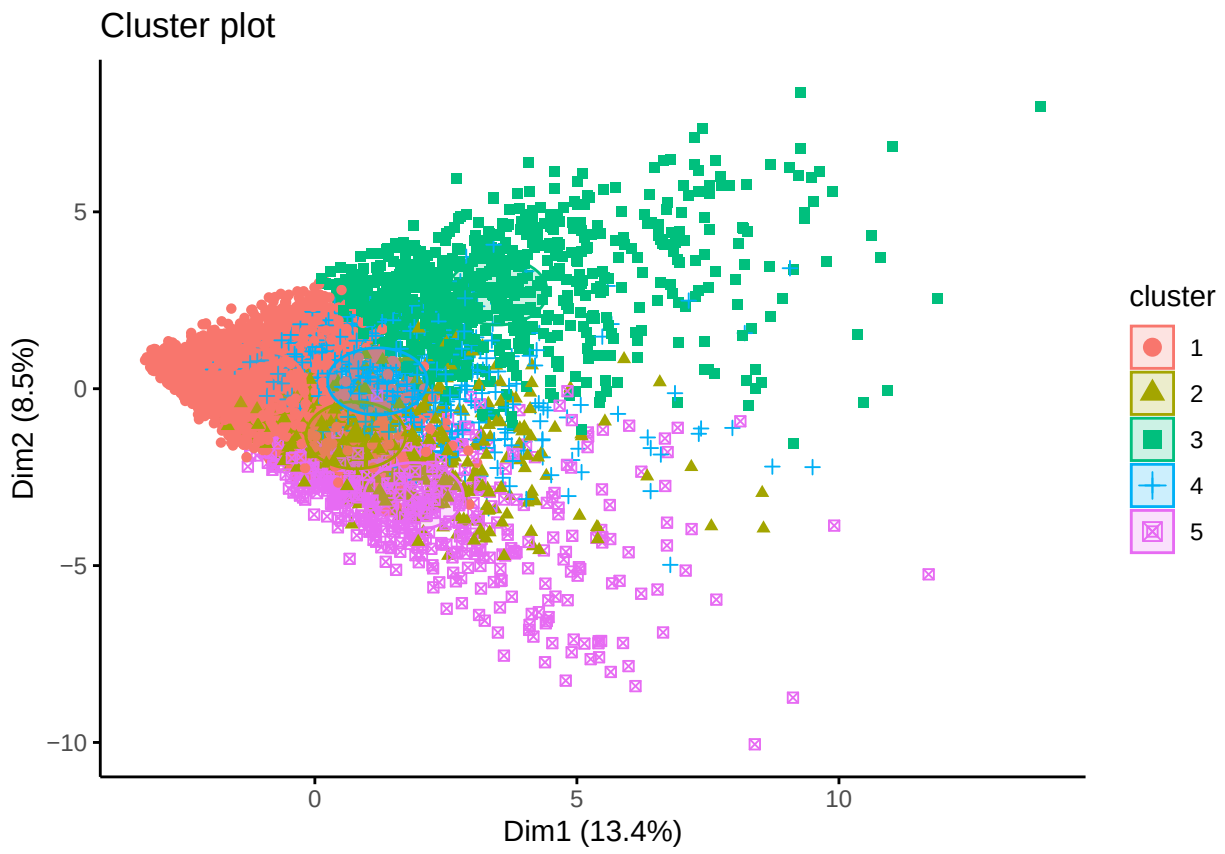
As we can see on the plot, photo sharing and chatter, politics and travel, religion and sports fandom and some other pair seems to have some strong correlation as pairs. Among all pairs, college_uni and online gaming has a extremely strong correlation.

Here we want to exclude chatter, spam and adult columns due to the fact that these columns are used to catch a lot of uncategorized data.

Now we need to decide the number of cluster we need for k-means



As we can see, there is much of an 'elbow' here. Here we choose 5 clusters as the number of clusters since the drop between 5 and 6 is not significant.



As we can see in the plot, the plot explains about 22% of the entire dataset, which is not super bad given the amount of data points and the noise within the data. We can find out what people are interested in the most in each cluster.

```
##                  V1 k
## 1  health_nutrition 2
## 2               eco 2
## 3          outdoors 2
## 4  personal_fitness 2
## 5     sports_fandom 3
## 6              food 3
## 7            family 3
## 8   home_and_garden 3
## 9            crafts 3
## 10         religion 3
## 11        parenting 3
## 12           school 3
## 13           travel 4
## 14          tv_film 4
## 15         politics 4
## 16             news 4
## 17        computers 4
## 18         business 4
## 19       automotive 4
## 20           dating 4
```

```
## 21   current_events 5
## 22    photo_sharing 5
## 23    uncategorized 5
## 24            music 5
## 25    online_gaming 5
## 26         shopping 5
## 27      college_uni 5
## 28   sports_playing 5
## 29          cooking 5
## 30              art 5
## 31           beauty 5
## 32          fashion 5
## 33   small_business 5
```

Now we can summarize the characteristics in each cluster.

Cluster 1: Middle Aged Adults who has Families These people shows an interest in family, parenting, school, food and sports fandom. They seemed to be couples who are raising children and having interests in area like gardeing and sports. They appears to be more settled down and not affected much by trends. They can be a good target for products involving housing decoration, family entertainment and etc.

Cluster 3: Younger Generation who lives through college life There people have passion in areas such as music, online gaming, sports playing, shopping and cooking. They seemed to be more energetic and following trends in the society. They have a stronger willing for sharing perspectives in life through social media and many of them are on the start-up stage of their own business. They are a great fit for products like electronic involving new technologies, gaming equipment, shopping deals and etc.

Cluster 4: Healthism Believer These people really cares about their personal health and are willing to spend time and money to keep themselves fit. They spend a lot of time in outdoor activities and has a relatively comprehensive understanding on nutrition. They can be a good target for nutrition supplement product, personal fitness classes, and eco-friendly products.

Cluster 5: Highly Educated Adults These people seems to focus on politics, news, traveling and dating. They don't necessarily have a family yet but they are mature enough to finance themselves for automotives. They counld be a potential target for cars and political news. They should also be a good fit for dating application.

Among all these, there seems to be no feature standing out in cluster 2, whcih is acceptable. As we can see previously on the plot, there are a lot of overlapping in the region of cluster 2 and thus maybe a smaller number of clusters may yield a better result compare to the current one.

The Reuters corpus

** Question: What are the least 10 frequent used words in the entire data set, can they be defined as less popular vocabularies? ** Approach: DocumentTermMatrix is used to calculate the frequencies of least 10 used words.

```
## 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

##
## 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

##   [1] "i"          "me"         "my"         "myself"     "we"
##   [6] "our"        "ours"       "ourselves"  "you"        "your"
##  [11] "yours"      "yourself"   "yourselves" "he"         "him"
##  [16] "his"        "himself"    "she"        "her"        "hers"
##  [21] "herself"    "it"         "its"        "itself"     "they"
##  [26] "them"       "their"      "theirs"     "themselves" "what"
##  [31] "which"      "who"        "whom"       "this"       "that"
##  [36] "these"      "those"      "am"         "is"         "are"
##  [41] "was"        "were"       "be"         "been"       "being"
##  [46] "have"       "has"        "had"        "having"     "do"
##  [51] "does"       "did"        "doing"      "would"      "should"
##  [56] "could"      "ought"      "i'm"        "you're"     "he's"
##  [61] "she's"      "it's"       "we're"      "they're"    "i've"
##  [66] "you've"     "we've"      "they've"    "i'd"        "you'd"
##  [71] "he'd"       "she'd"      "we'd"       "they'd"     "i'll"
##  [76] "you'll"     "he'll"      "she'll"     "we'll"      "they'll"
##  [81] "isn't"      "aren't"     "wasn't"     "weren't"    "hasn't"
##  [86] "haven't"    "hadn't"     "doesn't"    "don't"      "didn't"
##  [91] "won't"      "wouldn't"   "shan't"     "shouldn't"  "can't"
##  [96] "cannot"     "couldn't"   "mustn't"    "let's"      "that's"
## [101] "who's"      "what's"     "here's"     "there's"    "when's"
## [106] "where's"    "why's"      "how's"      "a"          "an"
## [111] "the"        "and"        "but"        "if"         "or"
## [116] "because"    "as"         "until"      "while"      "of"
## [121] "at"         "by"         "for"        "with"       "about"
## [126] "against"    "between"    "into"       "through"    "during"
## [131] "before"     "after"      "above"      "below"      "to"
## [136] "from"       "up"         "down"       "in"         "out"
## [141] "on"         "off"        "over"       "under"      "again"
## [146] "further"    "then"       "once"       "here"       "there"
```

```
## [151] "when"       "where"      "why"        "how"        "all"
## [156] "any"        "both"       "each"       "few"        "more"
## [161] "most"       "other"      "some"       "such"       "no"
## [166] "nor"        "not"        "only"       "own"        "same"
## [171] "so"         "than"       "too"        "very"
```

```
##   [1] "a"             "a's"           "able"          "about"
##   [5] "above"         "according"     "accordingly"   "across"
##   [9] "actually"      "after"         "afterwards"    "again"
##  [13] "against"       "ain't"         "all"           "allow"
##  [17] "allows"        "almost"        "alone"         "along"
##  [21] "already"       "also"          "although"      "always"
##  [25] "am"            "among"         "amongst"       "an"
##  [29] "and"           "another"       "any"           "anybody"
##  [33] "anyhow"        "anyone"        "anything"      "anyway"
##  [37] "anyways"       "anywhere"      "apart"         "appear"
##  [41] "appreciate"    "appropriate"   "are"           "aren't"
##  [45] "around"        "as"            "aside"         "ask"
##  [49] "asking"        "associated"    "at"            "available"
##  [53] "away"          "awfully"       "b"             "be"
##  [57] "became"        "because"       "become"        "becomes"
##  [61] "becoming"      "been"          "before"        "beforehand"
##  [65] "behind"        "being"         "believe"       "below"
##  [69] "beside"        "besides"       "best"          "better"
##  [73] "between"       "beyond"        "both"          "brief"
##  [77] "but"           "by"            "c"             "c'mon"
##  [81] "c's"           "came"          "can"           "can't"
##  [85] "cannot"        "cant"          "cause"         "causes"
##  [89] "certain"       "certainly"     "changes"       "clearly"
##  [93] "co"            "com"           "come"          "comes"
##  [97] "concerning"    "consequently"  "consider"      "considering"
## [101] "contain"       "containing"    "contains"      "corresponding"
## [105] "could"         "couldn't"      "course"        "currently"
## [109] "d"             "definitely"    "described"     "despite"
## [113] "did"           "didn't"        "different"     "do"
## [117] "does"          "doesn't"       "doing"         "don't"
## [121] "done"          "down"          "downwards"     "during"
## [125] "e"             "each"          "edu"           "eg"
## [129] "eight"         "either"        "else"          "elsewhere"
## [133] "enough"        "entirely"      "especially"    "et"
## [137] "etc"           "even"          "ever"          "every"
## [141] "everybody"     "everyone"      "everything"    "everywhere"
## [145] "ex"            "exactly"       "example"       "except"
## [149] "f"             "far"           "few"           "fifth"
## [153] "first"         "five"          "followed"      "following"
## [157] "follows"       "for"           "former"        "formerly"
## [161] "forth"         "four"          "from"          "further"
## [165] "furthermore"   "g"             "get"           "gets"
## [169] "getting"       "given"         "gives"         "go"
## [173] "goes"          "going"         "gone"          "got"
## [177] "gotten"        "greetings"     "h"             "had"
## [181] "hadn't"        "happens"       "hardly"        "has"
## [185] "hasn't"        "have"          "haven't"       "having"
## [189] "he"            "he's"          "hello"         "help"
```

```
## [193] "hence"         "her"           "here"          "here's"
## [197] "hereafter"     "hereby"        "herein"        "hereupon"
## [201] "hers"          "herself"       "hi"            "him"
## [205] "himself"       "his"           "hither"        "hopefully"
## [209] "how"           "howbeit"       "however"       "i"
## [213] "i'd"           "i'll"          "i'm"           "i've"
## [217] "ie"            "if"            "ignored"       "immediate"
## [221] "in"            "inasmuch"      "inc"           "indeed"
## [225] "indicate"      "indicated"     "indicates"     "inner"
## [229] "insofar"       "instead"       "into"          "inward"
## [233] "is"            "isn't"         "it"            "it'd"
## [237] "it'll"         "it's"          "its"           "itself"
## [241] "j"             "just"          "k"             "keep"
## [245] "keeps"         "kept"          "know"          "knows"
## [249] "known"         "l"             "last"          "lately"
## [253] "later"         "latter"        "latterly"      "least"
## [257] "less"          "lest"          "let"           "let's"
## [261] "like"          "liked"         "likely"        "little"
## [265] "look"          "looking"       "looks"         "ltd"
## [269] "m"             "mainly"        "many"          "may"
## [273] "maybe"         "me"            "mean"          "meanwhile"
## [277] "merely"        "might"         "more"          "moreover"
## [281] "most"          "mostly"        "much"          "must"
## [285] "my"            "myself"        "n"             "name"
## [289] "namely"        "nd"            "near"          "nearly"
## [293] "necessary"     "need"          "needs"         "neither"
## [297] "never"         "nevertheless"  "new"           "next"
## [301] "nine"          "no"            "nobody"        "non"
## [305] "none"          "noone"         "nor"           "normally"
## [309] "not"           "nothing"       "novel"         "now"
## [313] "nowhere"       "o"             "obviously"     "of"
## [317] "off"           "often"         "oh"            "ok"
## [321] "okay"          "old"           "on"            "once"
## [325] "one"           "ones"          "only"          "onto"
## [329] "or"            "other"         "others"        "otherwise"
## [333] "ought"         "our"           "ours"          "ourselves"
## [337] "out"           "outside"       "over"          "overall"
## [341] "own"           "p"             "particular"    "particularly"
## [345] "per"           "perhaps"       "placed"        "please"
## [349] "plus"          "possible"      "presumably"    "probably"
## [353] "provides"      "q"             "que"           "quite"
## [357] "qv"            "r"             "rather"        "rd"
## [361] "re"            "really"        "reasonably"    "regarding"
## [365] "regardless"    "regards"       "relatively"    "respectively"
## [369] "right"         "s"             "said"          "same"
## [373] "saw"           "say"           "saying"        "says"
## [377] "second"        "secondly"      "see"           "seeing"
## [381] "seem"          "seemed"        "seeming"       "seems"
## [385] "seen"          "self"          "selves"        "sensible"
## [389] "sent"          "serious"       "seriously"     "seven"
## [393] "several"       "shall"         "she"           "should"
## [397] "shouldn't"     "since"         "six"           "so"
## [401] "some"          "somebody"      "somehow"       "someone"
## [405] "something"     "sometime"      "sometimes"     "somewhat"
```

```
## [409] "somewhere"     "soon"          "sorry"         "specified"
## [413] "specify"       "specifying"    "still"         "sub"
## [417] "such"          "sup"           "sure"          "t"
## [421] "t's"           "take"          "taken"         "tell"
## [425] "tends"         "th"            "than"          "thank"
## [429] "thanks"        "thanx"         "that"          "that's"
## [433] "thats"         "the"           "their"         "theirs"
## [437] "them"          "themselves"    "then"          "thence"
## [441] "there"         "there's"       "thereafter"    "thereby"
## [445] "therefore"     "therein"       "theres"        "thereupon"
## [449] "these"         "they"          "they'd"        "they'll"
## [453] "they're"       "they've"       "think"         "third"
## [457] "this"          "thorough"      "thoroughly"    "those"
## [461] "though"        "three"         "through"       "throughout"
## [465] "thru"          "thus"          "to"            "together"
## [469] "too"           "took"          "toward"        "towards"
## [473] "tried"         "tries"         "truly"         "try"
## [477] "trying"        "twice"         "two"           "u"
## [481] "un"            "under"         "unfortunately" "unless"
## [485] "unlikely"      "until"         "unto"          "up"
## [489] "upon"          "us"            "use"           "used"
## [493] "useful"        "uses"          "using"         "usually"
## [497] "uucp"          "v"             "value"         "various"
## [501] "very"          "via"           "viz"           "vs"
## [505] "w"             "want"          "wants"         "was"
## [509] "wasn't"        "way"           "we"            "we'd"
## [513] "we'll"         "we're"         "we've"         "welcome"
## [517] "well"          "went"          "were"          "weren't"
## [521] "what"          "what's"        "whatever"      "when"
## [525] "whence"        "whenever"      "where"         "where's"
## [529] "whereafter"    "whereas"       "whereby"       "wherein"
## [533] "whereupon"     "wherever"      "whether"       "which"
## [537] "while"         "whither"       "who"           "who's"
## [541] "whoever"       "whole"         "whom"          "whose"
## [545] "why"           "will"          "willing"       "wish"
## [549] "with"          "within"        "without"       "won't"
## [553] "wonder"        "would"         "would"         "wouldn't"
## [557] "x"             "y"             "yes"           "yet"
## [561] "you"           "you'd"         "you'll"        "you're"
## [565] "you've"        "your"          "yours"         "yourself"
## [569] "yourselves"    "z"             "zero"

## Help on topic 'stopwords' was found in the following packages:
##
##   Package               Library
##   tm                    /Library/Frameworks/R.framework/Versions/4.2/Resources/library
##   corpus                /Library/Frameworks/R.framework/Versions/4.2/Resources/library
##
##
## Using the first match ...

## [1] "DocumentTermMatrix"    "simple_triplet_matrix"

##
```

```
## Attaching package: 'data.table'

## The following object is masked from 'package:slam':
##
##     rollup

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

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

## The following objects are masked from 'package:reshape2':
##
##     dcast, melt

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

## [1] 0

##     count
##  1:    NA
##  2:    NA
##  3:    NA
##  4:    NA
##  5:    NA
##  6:    NA
##  7:    NA
##  8:    NA
##  9:    NA
## 10:    NA
```

** Conclusion: The least 10 frequency used words in the entire data set are thing - 130, cchina - 132, moving - 133, considered - 133, either - 134, success - 134, ever - 137, initial - 138, figure - 138, quickly - 139. To some extent, we can assume that these terms are less popular because of the large amount of data we have which covering many authors and works.

Association Rule Mining

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

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

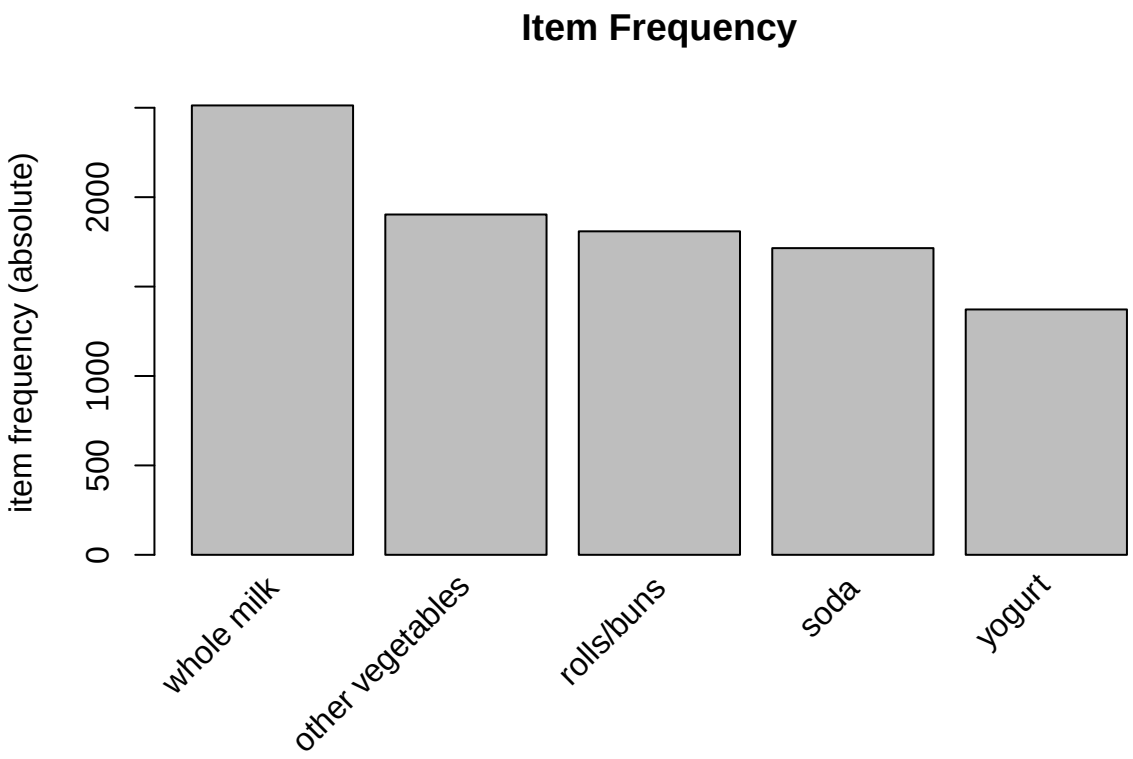
```
## The following object is masked from 'package:tidyr':
##
##     crossing

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

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

## 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

##
## 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
```

We can take a look at the data before we clean it up to get a general idea

```
## Eclat
##
## parameter specification:
##  tidLists support minlen maxlen            target  ext
##     FALSE    0.07      1     15 frequent itemsets TRUE
##
```

```
## algorithmic control:
##  sparse sort verbose
##       7   -2    TRUE
##
## Absolute minimum support count: 688
##
## create itemset ...
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [18 item(s)] done [0.00s].
## creating sparse bit matrix ... [18 row(s), 9835 column(s)] done [0.00s].
## writing  ... [19 set(s)] done [0.00s].
## Creating S4 object  ... done [0.00s].
```

```
##      items                           support    count
## [1]  {other vegetables, whole milk} 0.07483477  736
## [2]  {whole milk}                   0.25551601 2513
## [3]  {other vegetables}             0.19349263 1903
## [4]  {rolls/buns}                   0.18393493 1809
## [5]  {yogurt}                       0.13950178 1372
## [6]  {soda}                         0.17437722 1715
## [7]  {root vegetables}              0.10899847 1072
## [8]  {tropical fruit}               0.10493137 1032
## [9]  {bottled water}                0.11052364 1087
## [10] {sausage}                      0.09395018  924
## [11] {shopping bags}                0.09852567  969
## [12] {citrus fruit}                 0.08276563  814
## [13] {pastry}                       0.08896797  875
## [14] {pip fruit}                    0.07564820  744
## [15] {whipped/sour cream}           0.07168277  705
## [16] {fruit/vegetable juice}        0.07229283  711
## [17] {newspapers}                   0.07981698  785
## [18] {bottled beer}                 0.08052872  792
## [19] {canned beer}                  0.07768175  764
```



As we can see, whole milk has the highest frequency among all other, which is at a level of 2500. Then it followed by other vegetables, rolls, soda, and yogurt.

Now as we have some initial idea about the dataset, we can move on and try to figure out different association rules among all the values to see their correlation. First we can try support > 0.005 and confidence > 0.1 to see how many rules we get.

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##         0.1    0.1    1 none FALSE            TRUE       5   0.005      1
##  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: 49
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [120 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 done [0.00s].
## writing ... [1582 rule(s)] done [0.00s].
```

```
## creating S4 object  ... done [0.00s].

## set of 1582 rules
##
## rule length distribution (lhs + rhs):sizes
##   1   2   3   4
##   8 755 771  48
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   1.000   2.000   3.000   2.543   3.000   4.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.005084   Min.   :0.1000   Min.   :0.008134   Min.   :0.4457
##  1st Qu.:0.005897   1st Qu.:0.1454   1st Qu.:0.022979   1st Qu.:1.4732
##  Median :0.007321   Median :0.2189   Median :0.037112   Median :1.8174
##  Mean   :0.010537   Mean   :0.2557   Mean   :0.053306   Mean   :1.9028
##  3rd Qu.:0.010371   3rd Qu.:0.3315   3rd Qu.:0.058058   3rd Qu.:2.2444
##  Max.   :0.255516   Max.   :0.7000   Max.   :1.000000   Max.   :4.6399
##      count
##  Min.   :  50.0
##  1st Qu.:  58.0
##  Median :  72.0
##  Mean   : 103.6
##  3rd Qu.: 102.0
##  Max.   :2513.0
##
## mining info:
##  data ntransactions support confidence
##     g          9835   0.005        0.1
##                                                                 call
##  apriori(data = g, parameter = list(support = 0.005, conf = 0.1))
```

In this case, we generated 1582 rules, which is quite a lot to analyze. We can change the parameters to try to reduce the total number of rules and get stronger rules. Here we try with support > 0.001 and confidence > 0.85.

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.85    0.1    1 none FALSE            TRUE       5   0.001      1
##  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: 9
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
```

```
## sorting and recoding items ... [157 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 6 done [0.01s].
## writing ... [199 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].

## set of 199 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5  6
## 14 98 80  7
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.402   5.000   6.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.001017   Min.   :0.8500   Min.   :0.001017   Min.   : 3.327
##  1st Qu.:0.001017   1st Qu.:0.8708   1st Qu.:0.001118   1st Qu.: 3.558
##  Median :0.001220   Median :0.9091   Median :0.001322   Median : 3.634
##  Mean   :0.001229   Mean   :0.9095   Mean   :0.001359   Mean   : 4.134
##  3rd Qu.:0.001322   3rd Qu.:0.9231   3rd Qu.:0.001423   3rd Qu.: 4.586
##  Max.   :0.003152   Max.   :1.0000   Max.   :0.003559   Max.   :11.235
##      count
##  Min.   :10.00
##  1st Qu.:10.00
##  Median :12.00
##  Mean   :12.09
##  3rd Qu.:13.00
##  Max.   :31.00
##
## mining info:
##  data ntransactions support confidence
##     g          9835   0.001       0.85
##                                                                  call
##  apriori(data = g, parameter = list(support = 0.001, conf = 0.85))
```

This time we got a total number of 199 rules, which is much less that the previous case. We can see that among all the groups, group 3 and group 4 are the most common. For the sake of time, we will keep on with these threshold.

To get a better understanding of the rules, we can sort them by confidence. Here are the top 5 rules below.

```
##     lhs                     rhs              support confidence    coverage     lift count
## [1] {rice,
##      sugar}              => {whole milk} 0.001220132          1 0.001220132 3.913649    12
## [2] {canned fish,
##      hygiene articles}   => {whole milk} 0.001118454          1 0.001118454 3.913649    11
## [3] {butter,
##      rice,
##      root vegetables}    => {whole milk} 0.001016777          1 0.001016777 3.913649    10
## [4] {flour,
##      root vegetables,
##      whipped/sour cream} => {whole milk} 0.001728521          1 0.001728521 3.913649    17
```

```
## [5] {butter,
##      domestic eggs,
##      soft cheese}        => {whole milk} 0.001016777          1 0.001016777 3.913649    10
```

As we can see in the chart, whole milk appears on all the rhs with a confidence of 1. This implies that as all these lhs items was purchased, such as rice, sugar canned fish and etc, whole milk was always purchased with them. We can also look at these rules based on their lift.

```
##     lhs                          rhs                   support confidence    coverage      lift count
## [1] {liquor,
##      red/blush wine}         => {bottled beer}    0.001931876  0.9047619 0.002135231 11.235269    19
## [2] {citrus fruit,
##      fruit/vegetable juice,
##      other vegetables,
##      soda}                   => {root vegetables} 0.001016777  0.9090909 0.001118454  8.340400    10
## [3] {oil,
##      other vegetables,
##      tropical fruit,
##      whole milk,
##      yogurt}                 => {root vegetables} 0.001016777  0.9090909 0.001118454  8.340400    10
## [4] {other vegetables,
##      rice,
##      whole milk,
##      yogurt}                 => {root vegetables} 0.001321810  0.8666667 0.001525165  7.951182    13
## [5] {oil,
##      other vegetables,
##      tropical fruit,
##      whole milk}             => {root vegetables} 0.001321810  0.8666667 0.001525165  7.951182    13
```

By sorting with lift, we can see that liquor and bottled bear has the heighest life, which indicates that they are about 11 times more likely to be purchased together compare to selling individually.

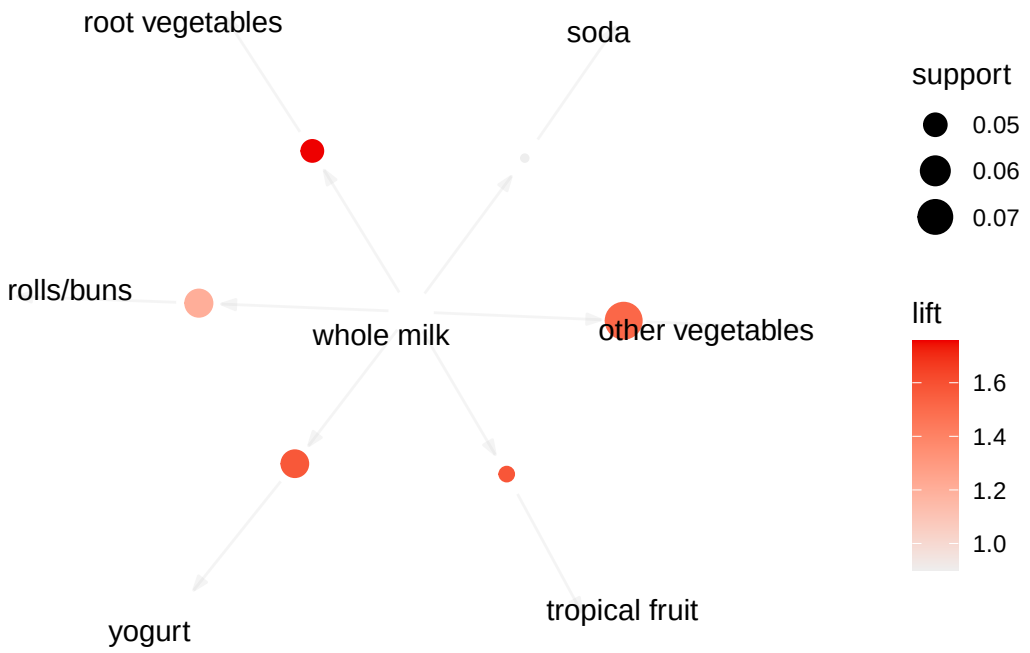
Since we have know that whole milk has the highest frequency among all items, it could be helpful for us to understand the purchase pattern of customers who purchase whole milk.

```
##     lhs             rhs                support    confidence coverage lift
## [1] {whole milk} => {other vegetables} 0.07483477 0.2928770  0.255516 1.513634
## [2] {whole milk} => {rolls/buns}       0.05663447 0.2216474  0.255516 1.205032
## [3] {whole milk} => {yogurt}           0.05602440 0.2192598  0.255516 1.571735
## [4] {whole milk} => {root vegetables}  0.04890696 0.1914047  0.255516 1.756031
## [5] {whole milk} => {tropical fruit}   0.04229792 0.1655392  0.255516 1.577595
##     count
## [1] 736
## [2] 557
## [3] 551
## [4] 481
## [5] 416
```

In this case, we can see that other vegetables, rolls and yogurt seems to be purchased with a confidence of 20-30% when whole milk is purchased. We can have a better understanding by turing this information in visual graph.

```
## Warning in plot.rules(new_rules_s, method = "graph", control = list(nodeCol
## = grey.colors(1), : The parameter interactive is deprecated. Use
## engine='interactive' instead.
```

```
## Warning: Unknown control parameters: nodeCol, edgeCol, alpha

## Available control parameters (with default values):
## layout    =  stress
## circular  =  FALSE
## ggraphdots    =  NULL
## edges     =  <environment>
## nodes     =  <environment>
## nodetext  =  <environment>
## colors    =  c("#EE0000FF", "#EEEEEEFF")
## engine    =  ggplot2
## max   =  100
## verbose   =  FALSE
```



The graph showed us a better visual for the rules. The bigger the dot size, the stronger the support that type of grocery has with whole mild. The more the color is towards red, the higher the lift it has with whole milk. As we can see, root vegetables has the strongest lift with whole milk among all other grocery. On the other hand, the other vegetables have a stronger lift. Yogurt kind of sits in the middle, where its lift and support are both quite strong. This make sense in the following way. Root vegetables such as potatoes' recipe involves a lot of use of whole milk, which explains why it has a strong lift. For yogurt, it is usually placed next to whole milk on the shelf. Thus it's support and lift to whole milk are both high. The only vague correlation is the high support between whole milk and other vegetables. One possible reason is that just like whole milk, other vegetables are also a frequently purchased item. Even though they do not necessarily have a relationship, their high requency caused the model falsely draw them toghther.