STA380, Part 2: Exercises

Shreyansh Agarawal (sa55742), Praneet Kumar Alamuri (pa22222), Maanvi Goyal (mg65952), Prathmesh Savale (ps33296)

2022-08-15

Link to github repository - https://github.com/praths007/STA-380-Part2 (https://github.com/praths007/STA-380-Part2)

Probability practice|Part A

Y = a visitor clicked Yes.

N = a visitor clicked No.

R = a visitor is a random clicker.

T = a visitor is a truthful clicker

P(Y)= 0.65, P(N)= 0.35 , P(T)=0.7 , P(R)=0.3

P(Y|R)=0.5

P(N|R)=0.5

As per the law of total probability:

$$P(A) = P(A|B).P(B) + P(A|C).P(C)$$

The probability of people who answered *'Yes'* can be calculated as:

$$P(Y) = P(Y|T).P(T) + P(Y|R).P(R)$$

$$0.65 = P(Y \cap T) + 0.5 * 0.3$$

$$0.5 = P(Y \cap T)$$

Hence, there are **50%** of the people who are truthful clickers and answered 'Yes'

$$5/7 = P(T|Y)$$

Hence, there are **71.43%** of the people who are truthful clickers and given they answered 'Yes'

Probability practice|Part B

P = a person has tested positive

N = a person has tested negative

D = a person has a disease

ND = a person does not has a disease

P(D)= 0.000025, P(ND)= 0.999975 , P(P|D)=0.993 , P(N|ND)=0.9999

$$P(P|ND) + P(N|ND) = 1$$

P(P|ND)=1-P(N|ND)=1-0.9999

P(P|ND)=0.0001

As per the law of total probability:

$$P(A) = P(A|B).\,P(B) + P(A|C).\,P(C)$$

The probability of people who are positive can be calculated as:

$$P(P) = P(P|D).\,P(D) + P(P|ND).\,P(ND)$$

$$P(P) = (0.993)(0.000025) + (0.0001)(0.999975)$$

$$P(P) = 0.00012482$$

As per Naive Bayes Theorem:

$$P(A|B) = [P(B|A).\,P(A)]/P(B)$$

The probability of people who have disease when they tested positive is:

$$P(D|P) = [P(P|D).\,P(D)]/P(P)$$

$$P(D|P) = [(0.993)(0.000025)]/(0.00012482)$$

$$P(D|P) = 0.19888$$

Hence, the probability of people who have disease when they tested positive is **19.88%**

Wrangling the Billboard Top 100|Part A

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

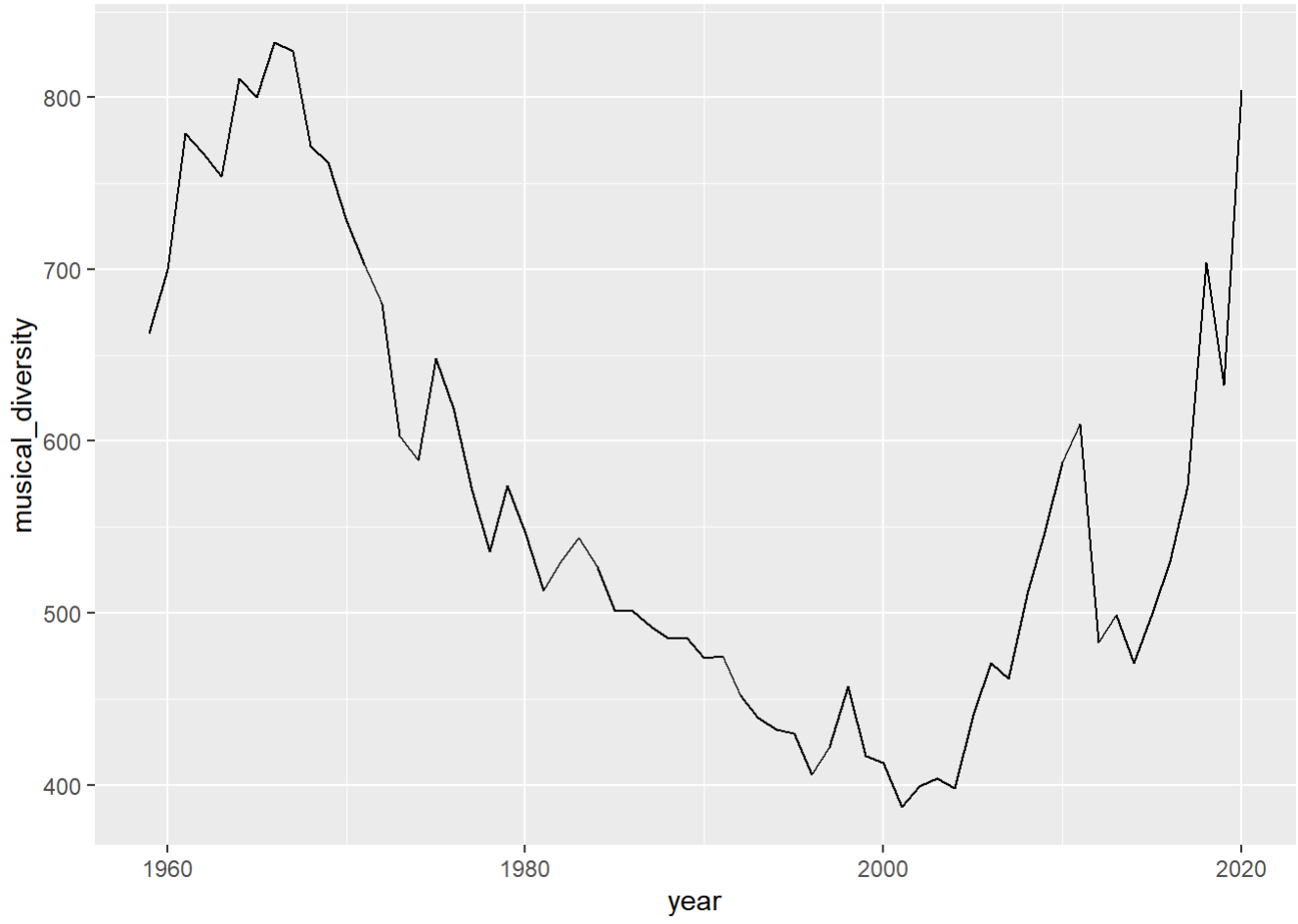
```
## [1] "Below table highlights the 10 most popular songs sung by various performers since 1958 o
n the Billboard Top 100"
```

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

Wrangling the Billboard Top 100|Part B

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

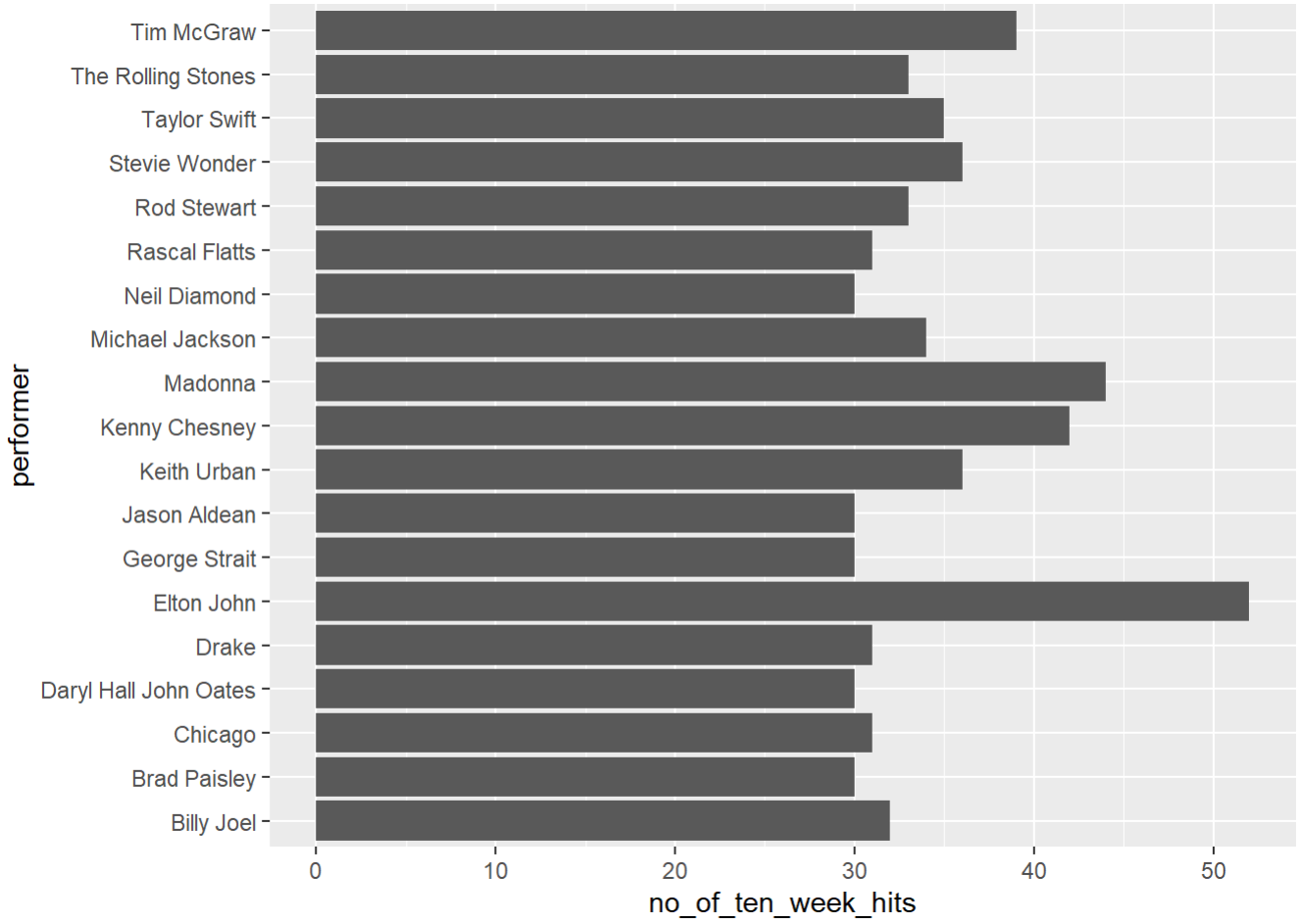
```
## [1] "Below line chart represents the musical diversity of given year as the number of unique
songs that appeared in the Billboard Top 100 that year"
```



Wrangling the Billboard Top 100|Part C

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

```
## [1] "Below bar plot represents 19 artists in U.S. musical history since 1958 who have had at
least 30 songs that were ten-week hits"
```



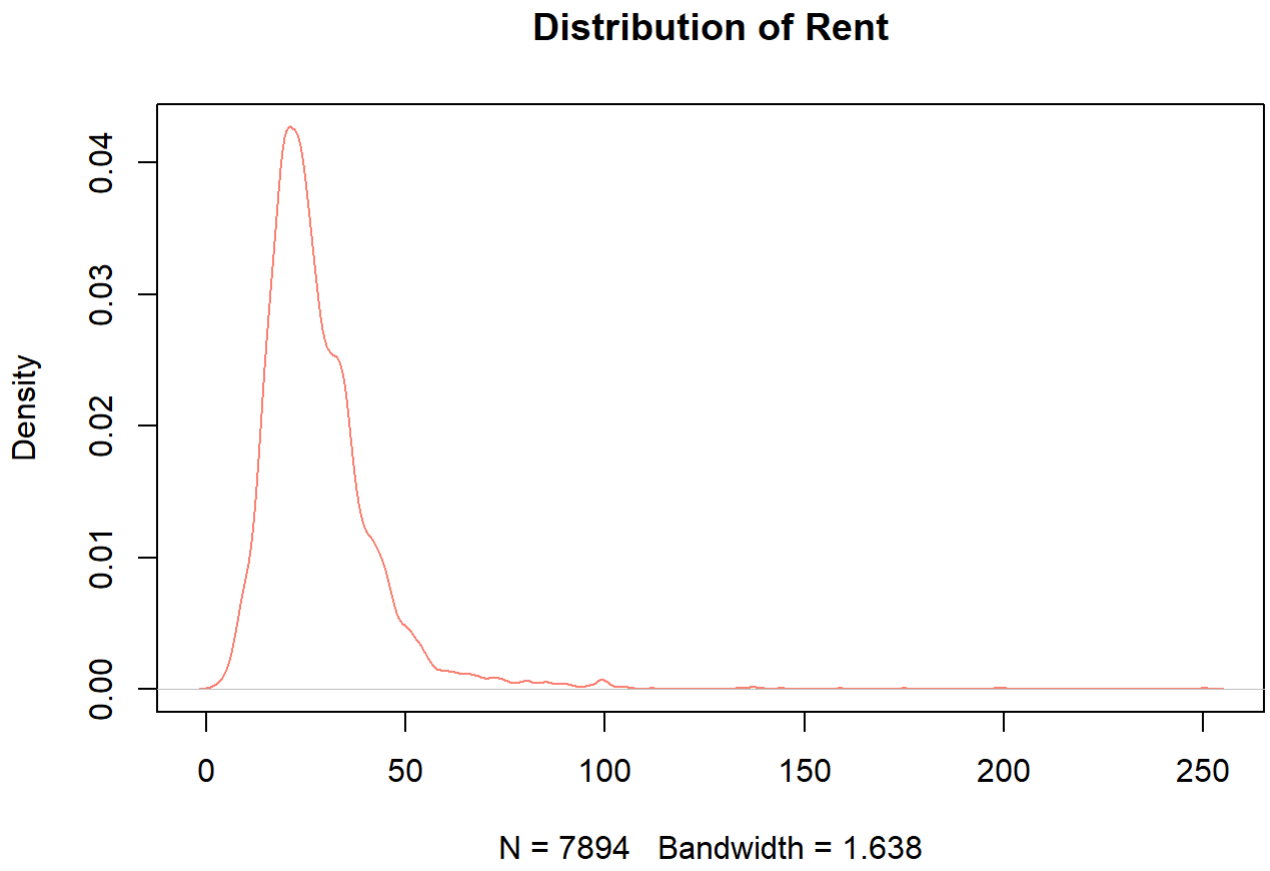
Visual story telling part 1: green buildings

The excel guru had the following methodology -

- Removing the outliers i.e. buildings with less than 10% occupancy which resulted in median rent for non-green houses = $25 , green houses = $27.60 .
- The difference in rent per square foot is: $2.60 which is resulting in an added revenue of $650,000
- Now, the extra cost in $5,000,000 for a green building certification and hence the cost can be recovered in approximately ~8 years, and then on, there will be a net profit of $650,000 per year.

Although the calculations look reasonable, there are a few things that the excel guru has missed which we can now take a look into:

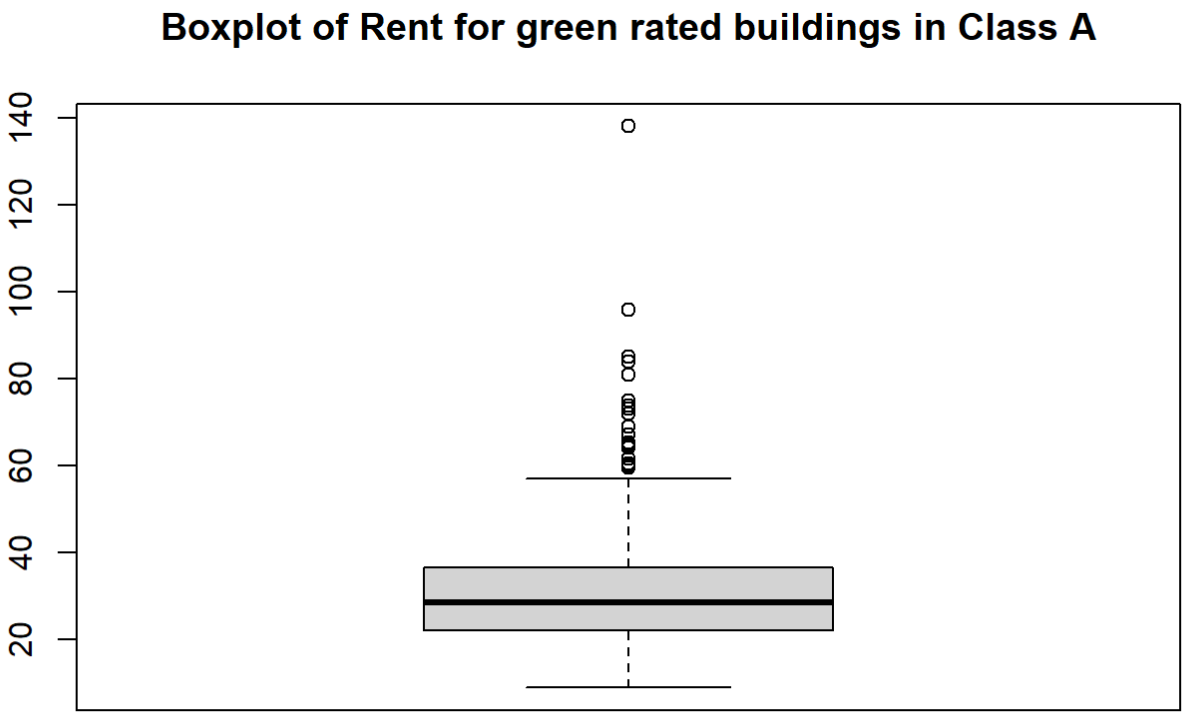
We are using median rent as we can see from the distribution that we have a right tail which will skew our mean.



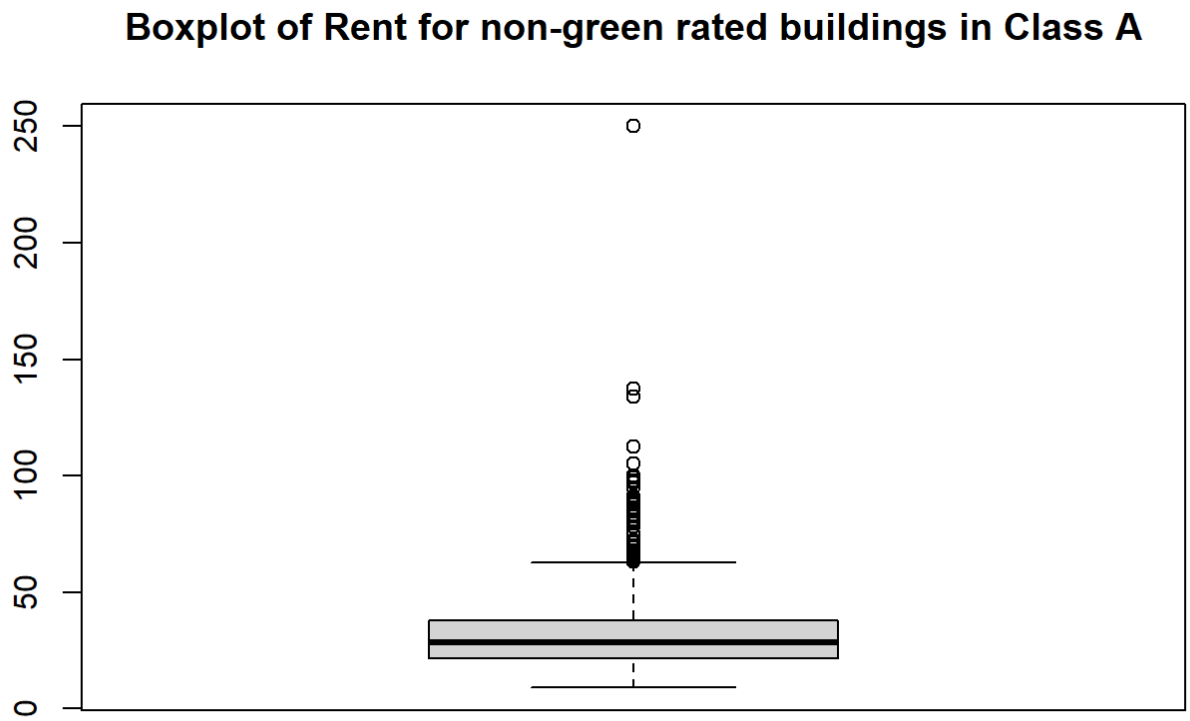
Looking into house classes

Class A Houses

- When we look into the proportion of Class A houses in non-green and green houses, we see that about 80% of the green houses are Class A as opposed to only ~35% in non-green houses
- There may be the case that the median rent for green-house are higher just because they have a higher proportion of Class A houses
- Let's look into the median rent for Class A houses


Boxplot of Rent for green rated buildings in Class A
20
40
60
80
100
120
140

```
## [1] 28.44
```

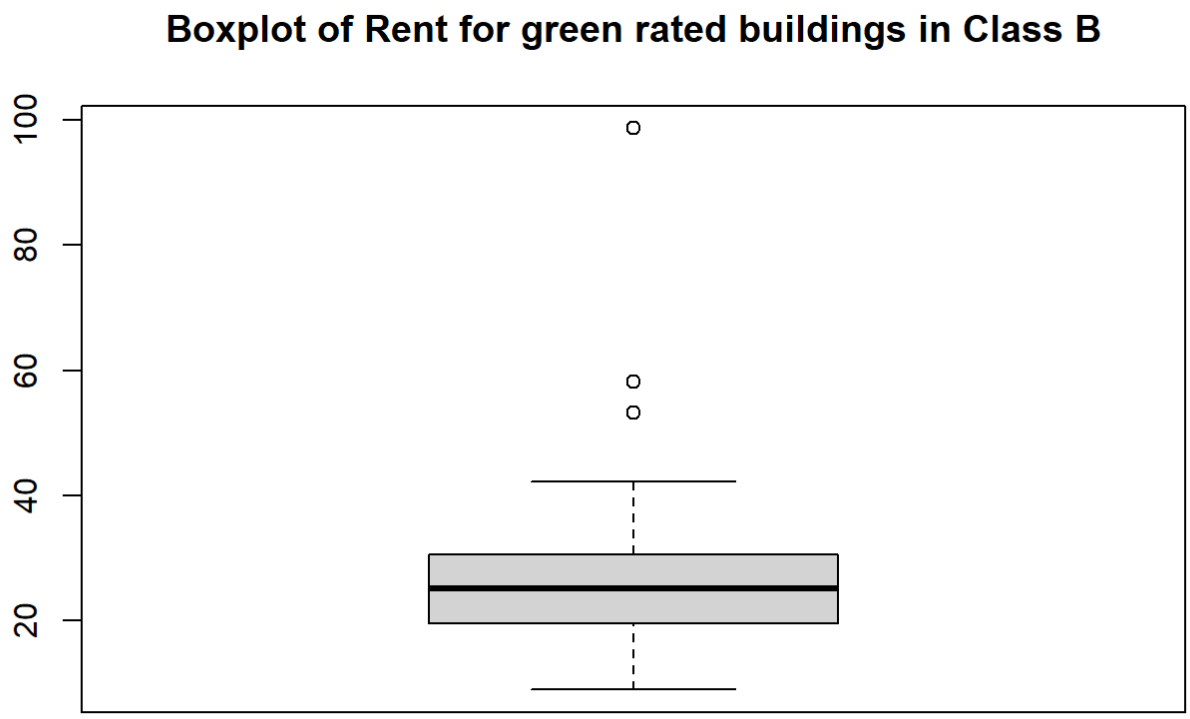



```
## [1] 28.2
```

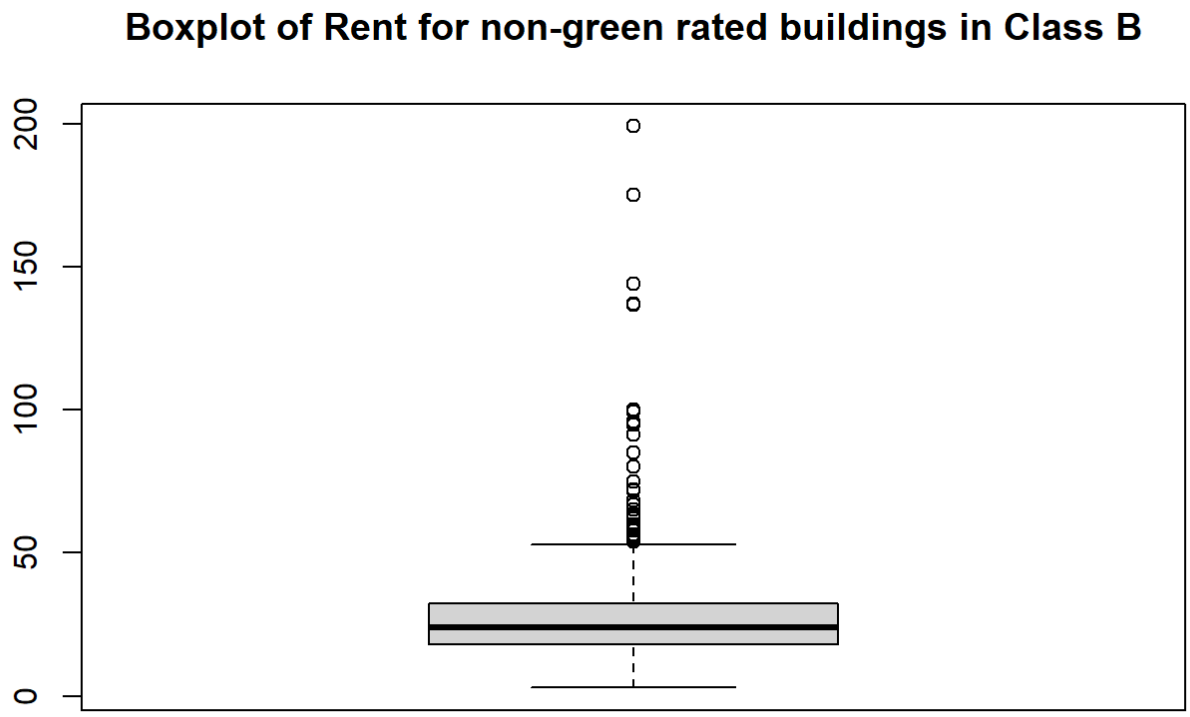
- Class A Overall : 28.2
- Class A | Green: 28.44
- Class A | Non-Green: 28.2
- As we can see that, the rent of Class A houses does not depend on whether the house is green rating certified or not
- So, if we compare a Class A non-green house and a Class A green house, there isn't much difference in rent; only 0.2 per square feet i.e. just $50,000 extra premium per year

Class B Houses

- Let's look into the median rent for Class B houses


Boxplot of Rent for green rated buildings in Class B
100
80
60
40
20

```
## [1] 25.1
```




```
## [1] 24
```

- Class B Overall : 24.0
- Class B | Green: 25.1
- Class B | Non-Green: 24.0
- As we can see that the rent for Class B Green house is slightly higher than Class B non-green house, i.e. ~1.1 higher rent per square feet, which translates to ~275,000 extra premium per year.
- If there is $5,000,000 extra cost for green certification, at 90% occupancy rate, it might take about 20 years to break even, and then the building can earn ~275,000 extra premium per year.

Class C Houses

Looking into Class C houses rent for green houses, we see that the rent for green building Class C houses are more than Class A and B which does not make sense. This might be due to very few green buildings belonging to Class C and hence we cannot rely on the median/ mean rent.

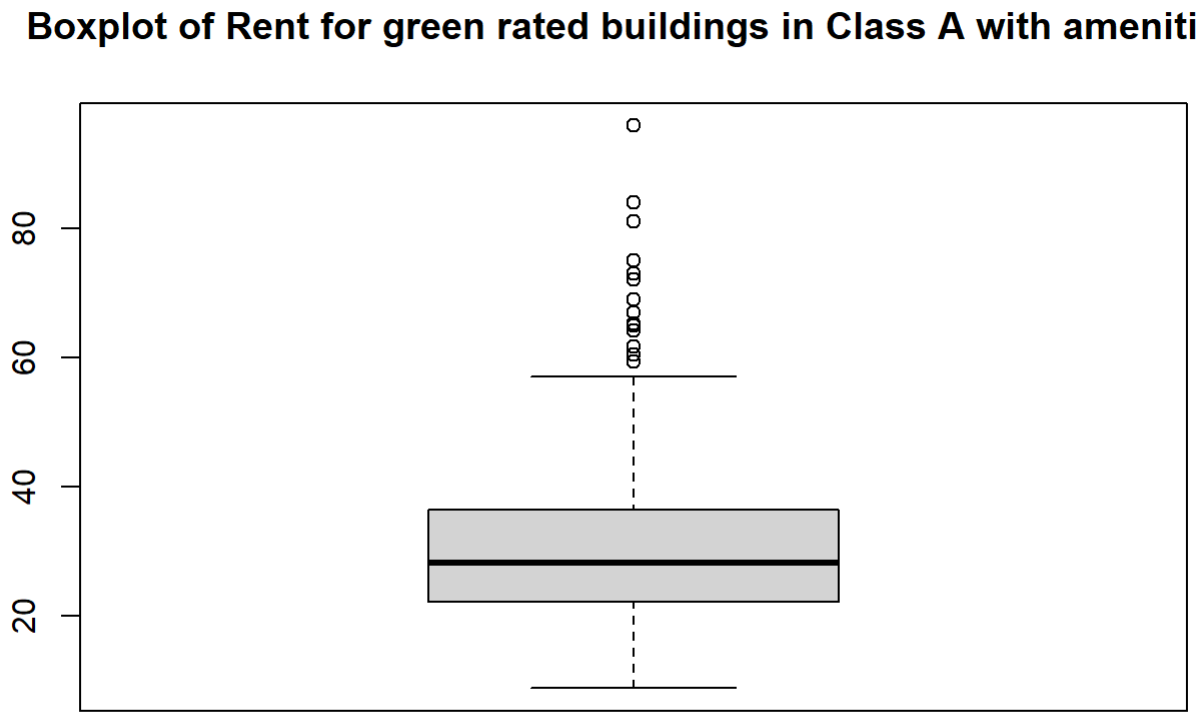
Overall, based on house class, if we compare Class A houses, then there isn't any cost benefit in getting a green building certification.

For Class B houses, there is some cost benefit but again, it'll take ~20 years to break even and does not sound like a very viable option. But again, it depends on the builder if for them a time span of 20 years sound economically viable.

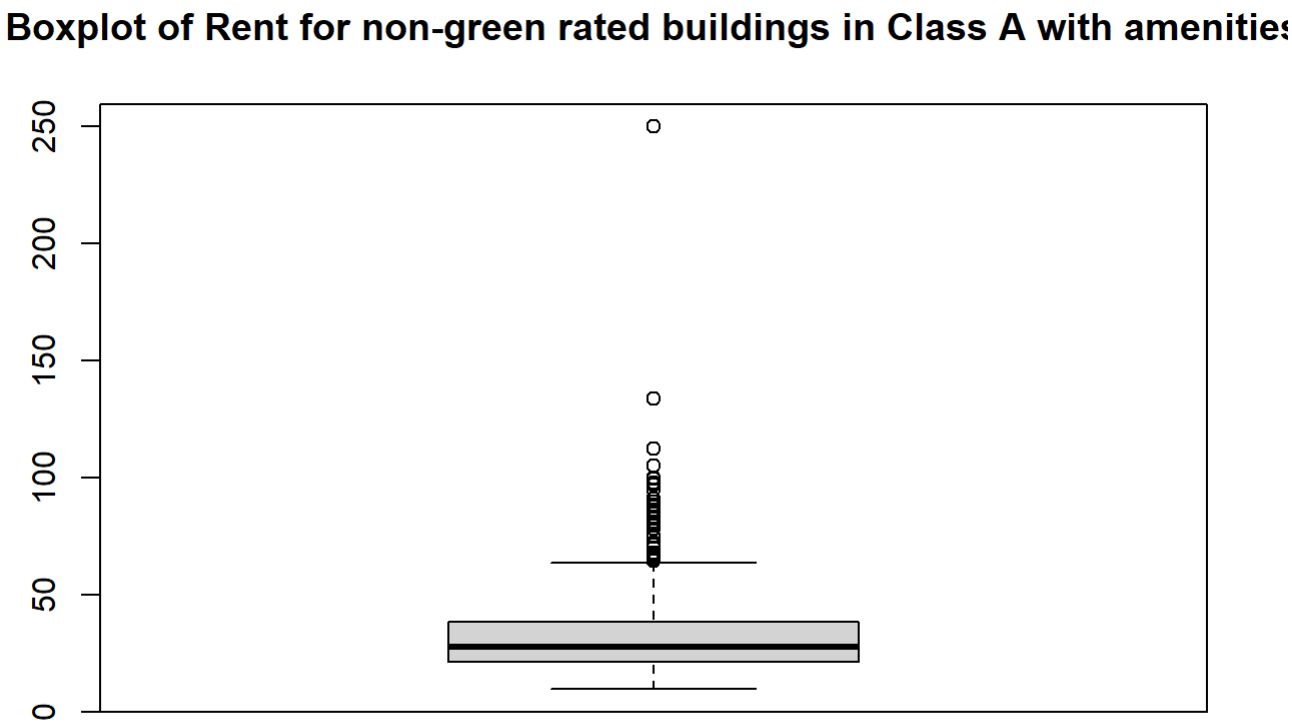
Looking into Amenities

Since, ~75% of green rated buildings have good amenities (amenities=1), let's compare the median rent amongst such buildings by class.

Class A buildings with amenities:



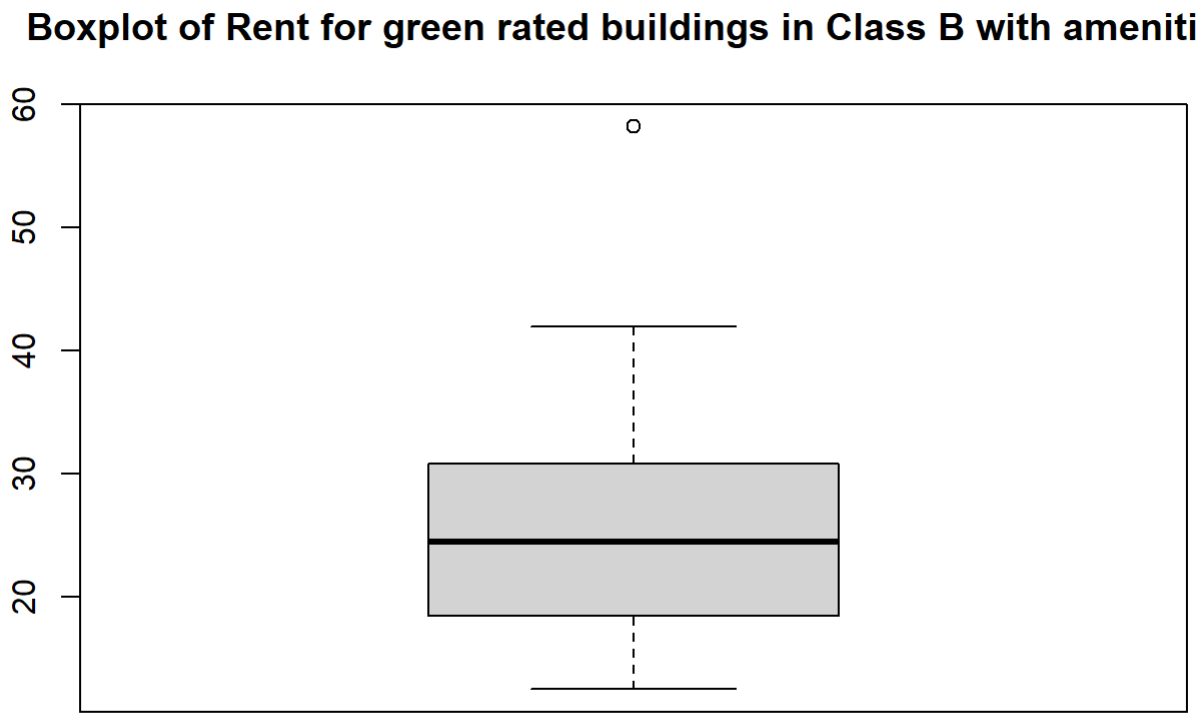

```
## [1] 28.2
```



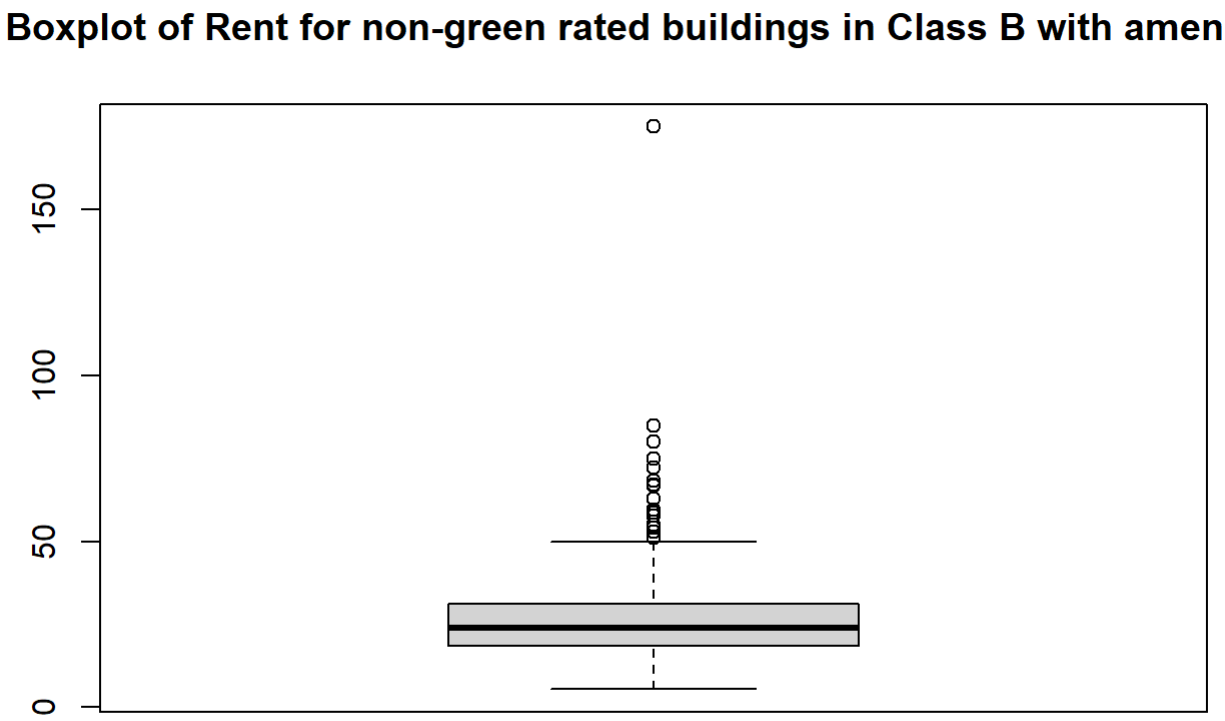

```
## [1] 27.73
```

- Green: 28.2
- Non-Green: 27.73
- Here, we can see that the premium for green building isn't that much, only 0.43 per sq feet which translates to ~$ 105,750 added premium in a year and will take the building ~47 years to break even

Class B buildings with amenities:


Boxplot of Rent for green rated buildings in Class B with amenities
60
50
40
30
20

```
## [1] 24.46
```




```
## [1] 23.89
```

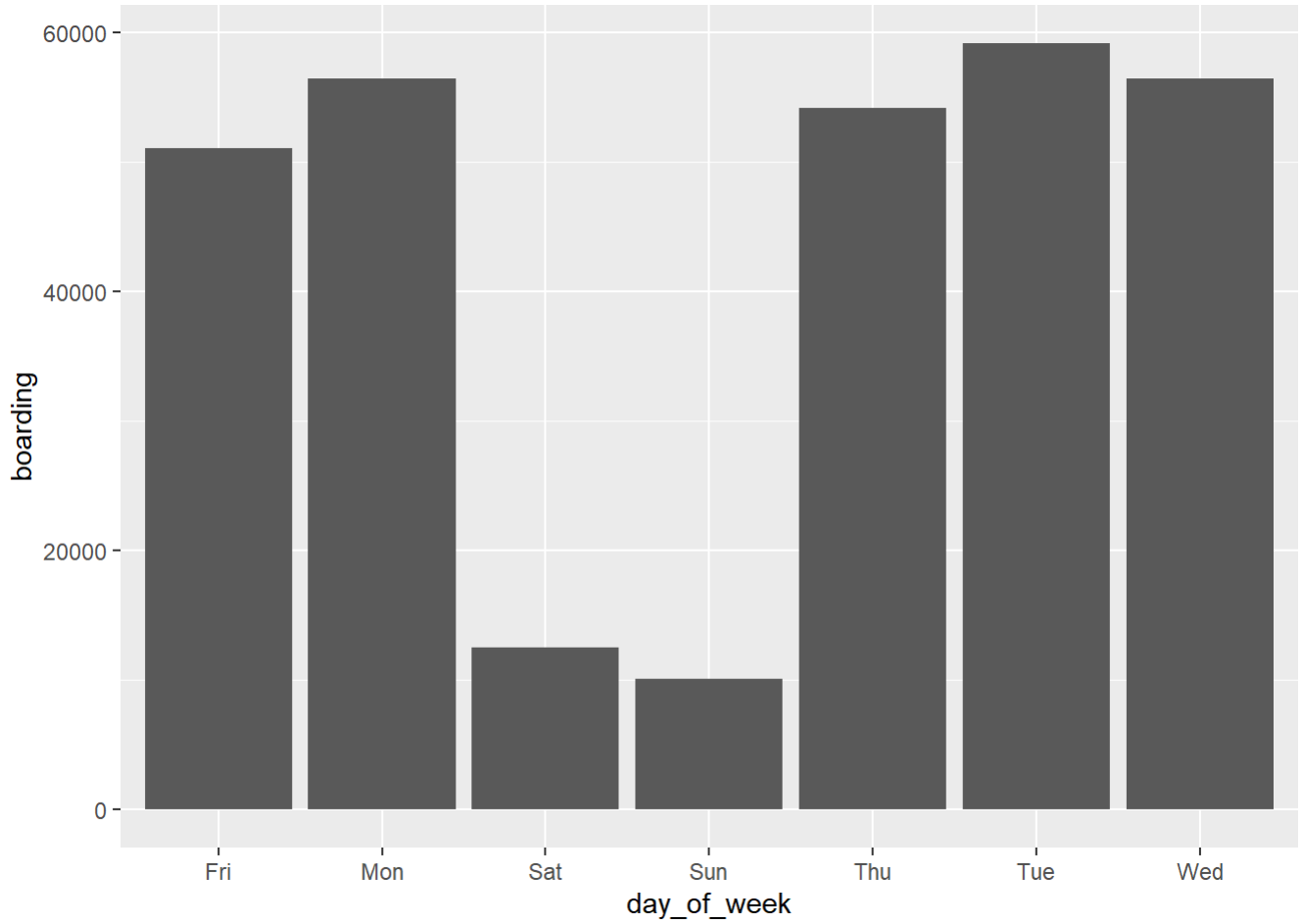
- Green: 24.46
- Non-Green: 23.89
- Here, we can see that the premium for green building isn't that much, only 0.51 per sq feet which translates to ~$ 128,250 added premium in a year and will take the building ~39 years to break even

Hence based on our analysis and looking into House class, amenities etc. we can see that there isn't much premium for buildings with green rating. In nutshell, we can say that green building status isn't the only factor that is contributing to the "extra perceived" rent when we compare green buildings and non-green buildings, and hence we need to consider all the variables to make an informed estimate for the same.

Visual story telling part 2: Capital Metro data

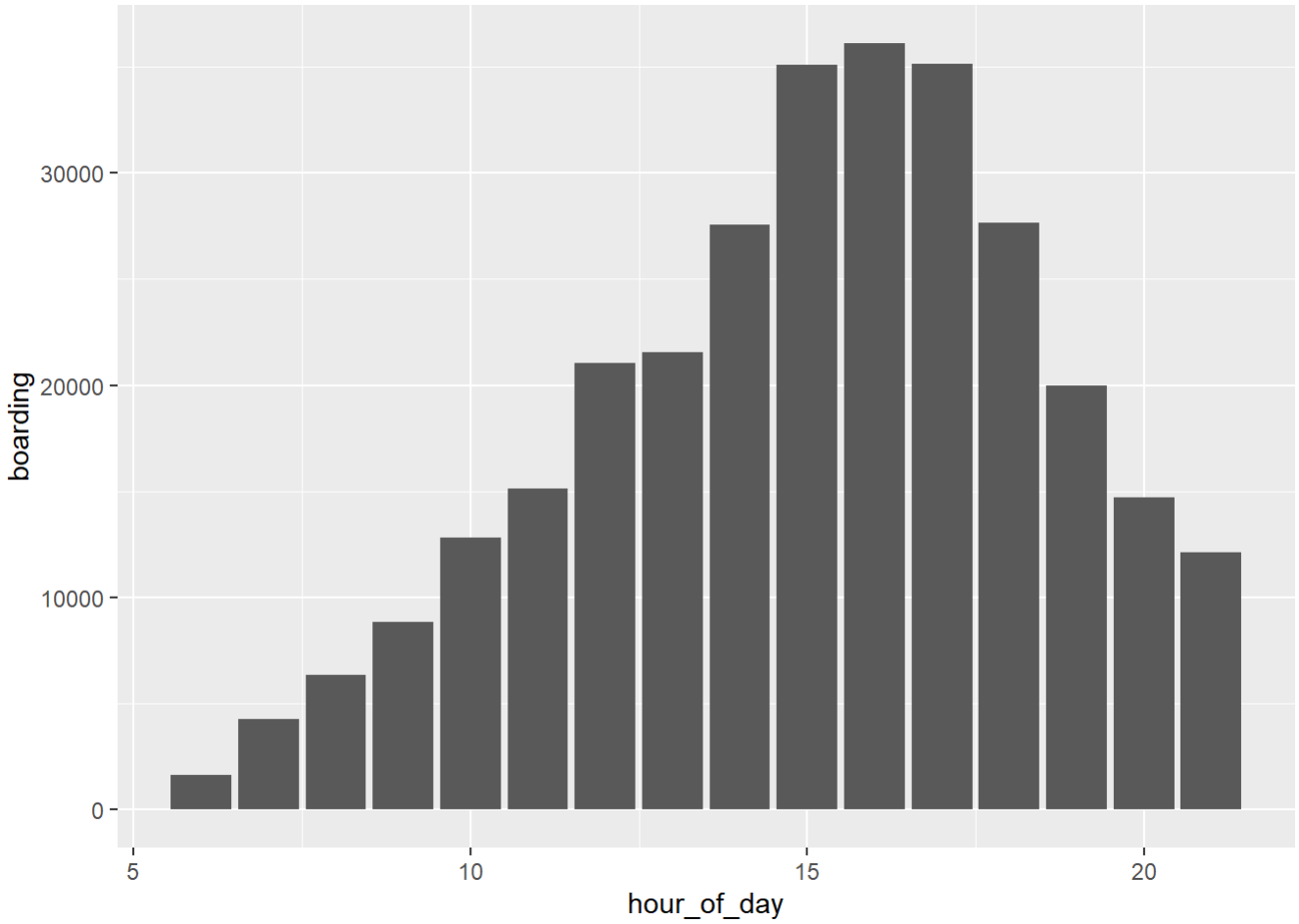
Let's look into how the the boarding varies by each day…

As you can see from the chart below is that traffic during Saturday and Sunday is very low as compared to weekdays which makes sense as since classes are not conducted during weekends, less students are near the UT area and hence the lower traffic.

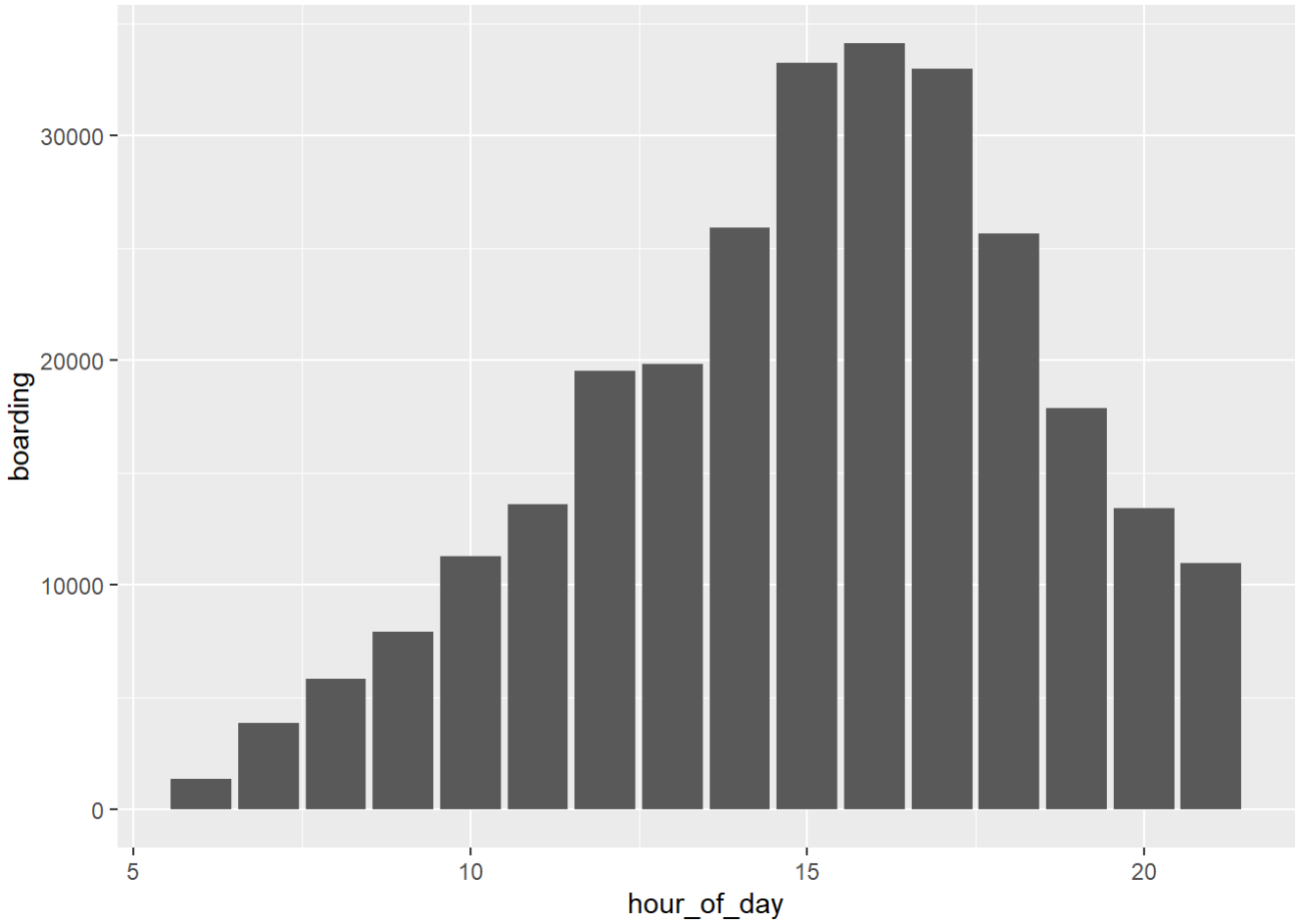


Now lets's observe if the traffic varies by the time of day (all days combined)…

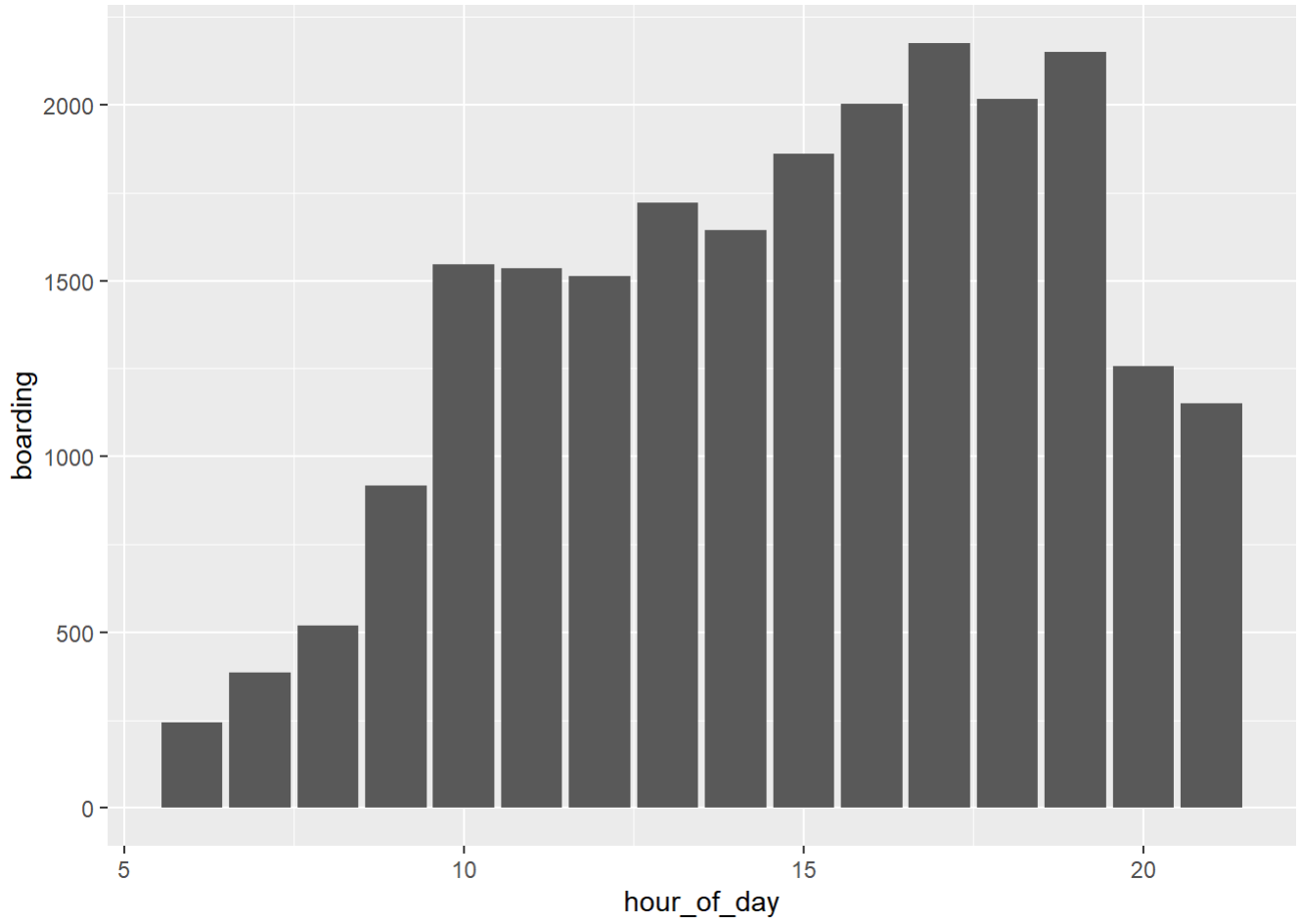
From the graph below, we can see high traffic during 2PM to 6PM, and highest traffic during 3PM to 5PM which makes sense as students return from campus during 3-5 PM the most. Post 6PM the traffic tapers off as less and less students are near the campus area.



Now let's look into the traffic by hours during the weekdays…

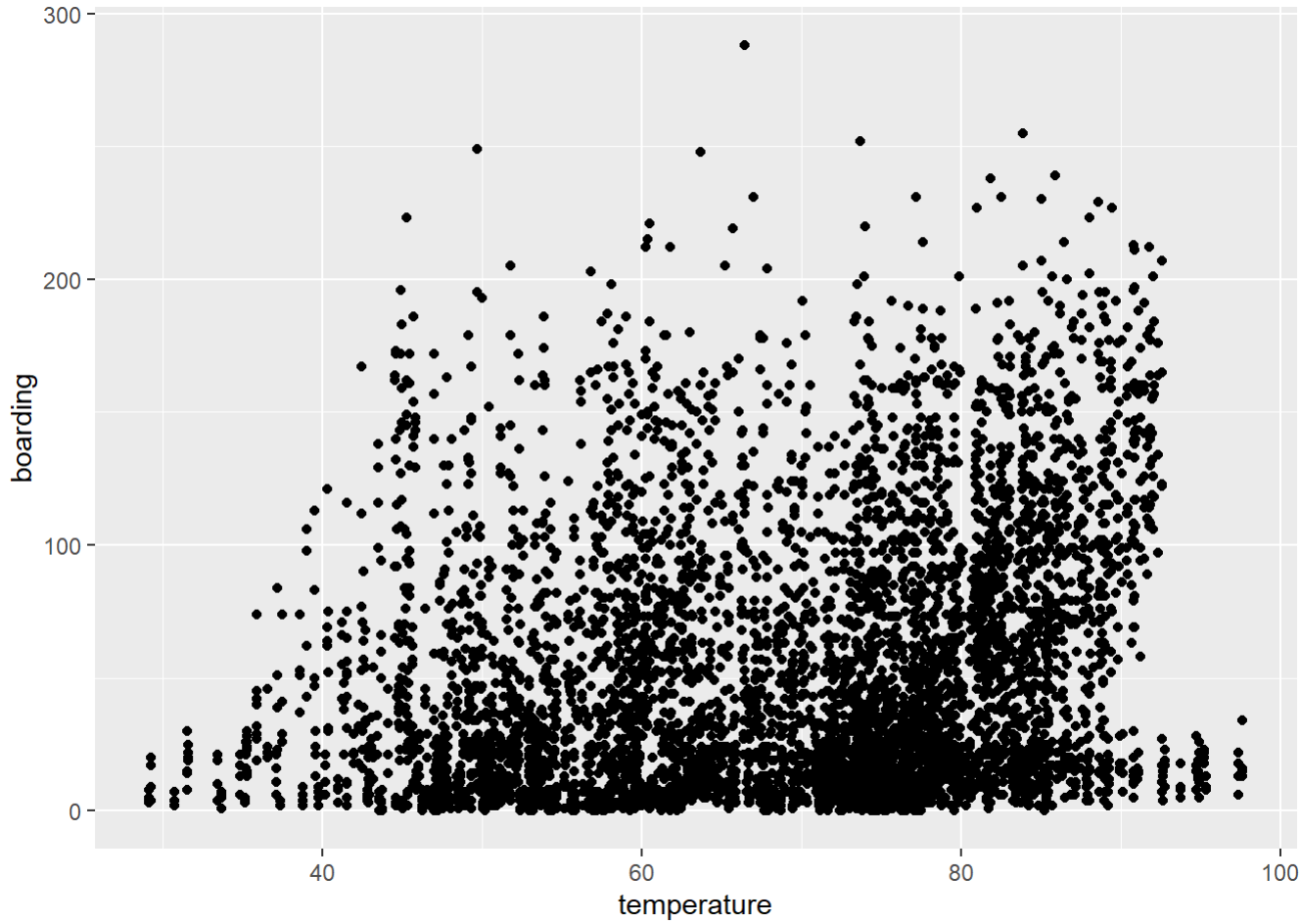


If we look into the traffic during only weekends, we can see that the traffic is more or less the same from 10 AM to 6PM (overall traffic much lower than weekdays), which makes sense as there are no classes, so there aren't any peak hours as such and the traffic is consistent across hours.



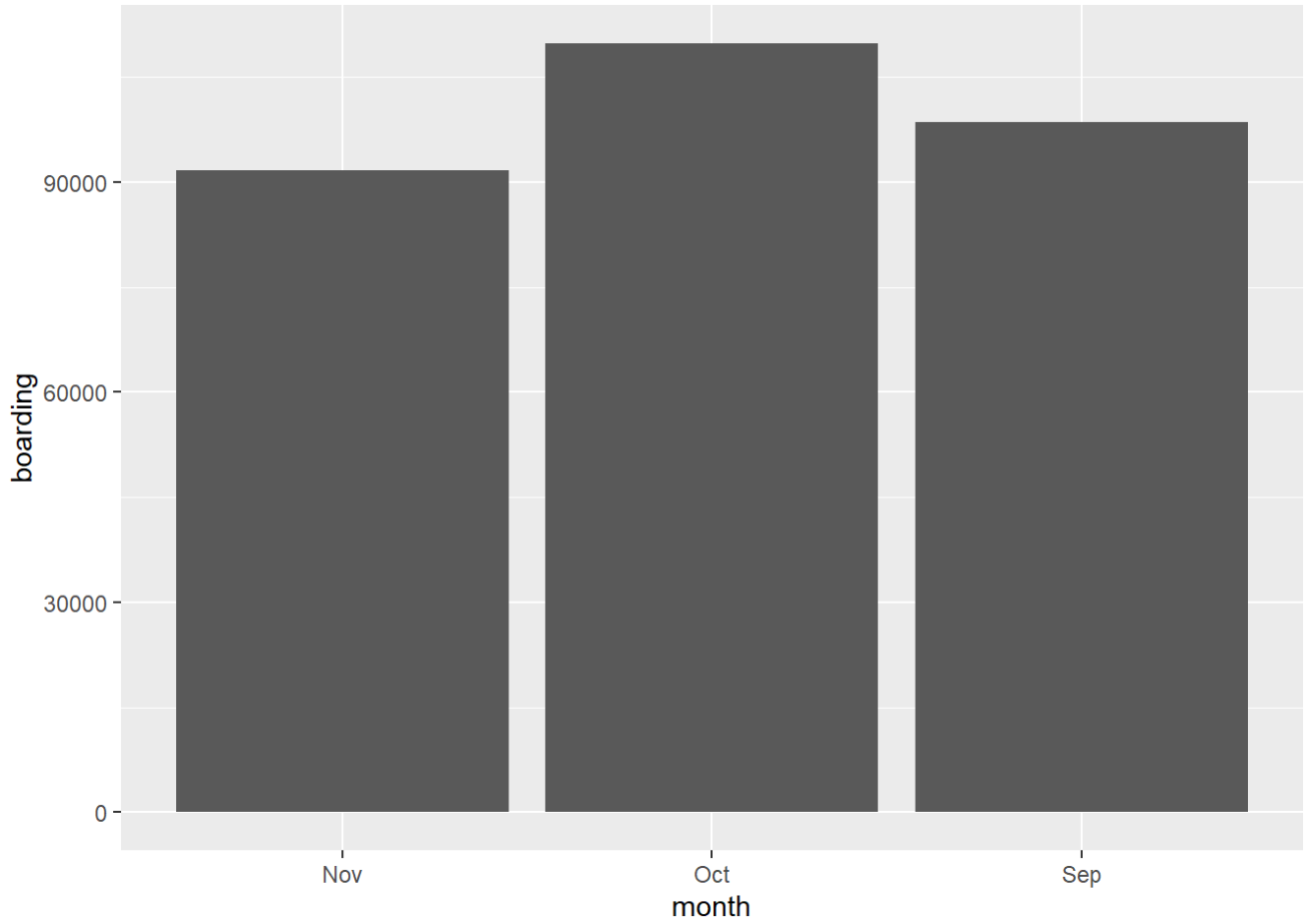
Now, let's check if there is any relation b/w traffic and temperature.

From the scatter plot we can see there is no clear trend b/w temperature and traffic in the UT area. I guess because you have to attend class irrespective of the temperature.



Now lets' check if the traffic changes by months…

Form the plot below, we can see that there is slightly higher traffic during October.



Portfolio modeling

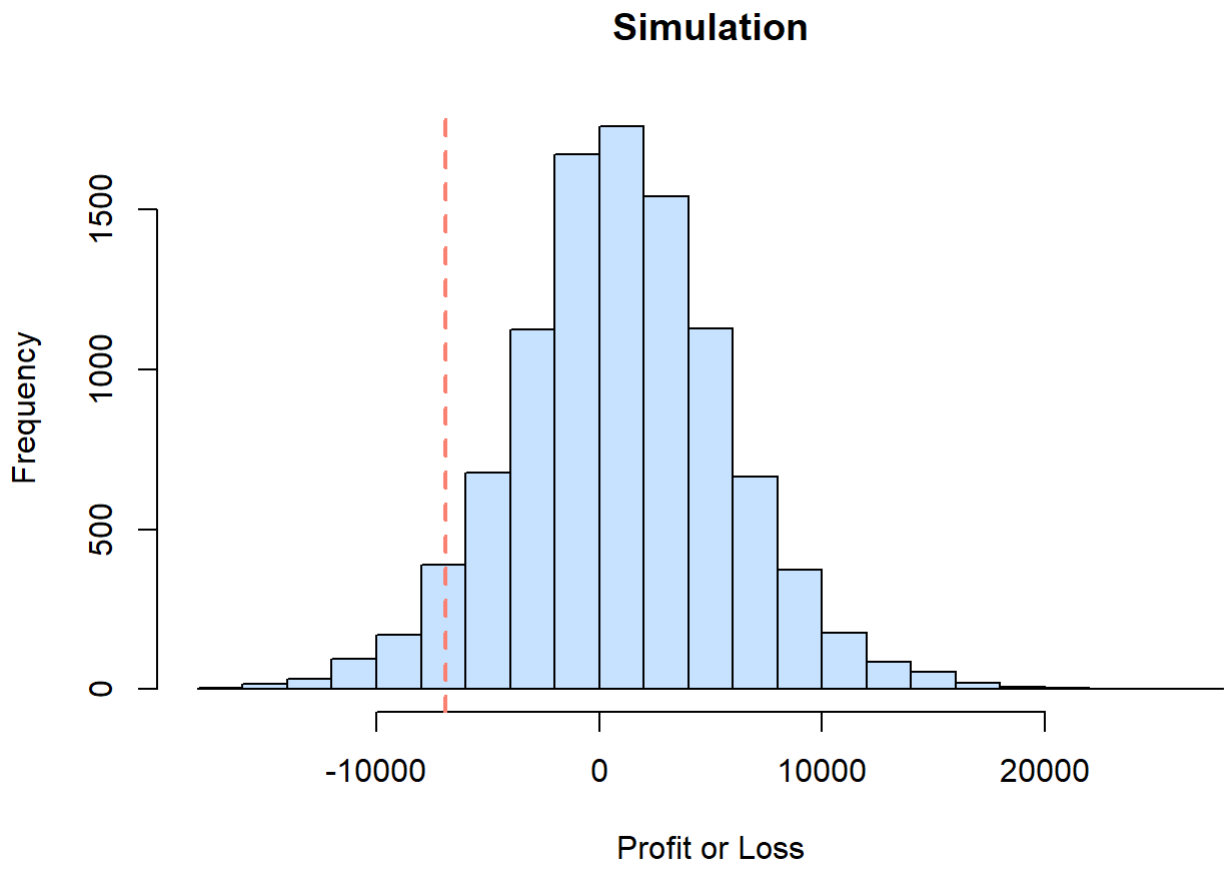
We have created 2 portfolios using the $100,00 and assessed the performance in a span of 20 days using the bootstrap re sampling. Additionally, we have assumed that our portfolio is re balanced each day at zero transaction cost.

Portfolio 1:

The first portfolio that we have is slightly conservative i.e. low risk and low return which comprises of 5 large cap stocks with equal distribution. We have 3 stocks from Pharama giants i.e. Pfizer, Novartis and Johnson & Johnson, and two large cap funds, Nestle and JP Morgan & Chase. The idea is to select strong companies which are less likely to fall to create a conservative portfolio.

Below is the profit simulation using bootstrap re-sampling.

We got a expected (mean) return of $950 and we are 95% confident that our loss in 20 day period won't exceed $6,834 i.e our VaR at 5%.



Portfolio 2:

Our 2nd portfolio is a bit more risky, with relatively high risk and high return compared to the first portfolio.

We have a mix of large and mid cap funds from different industries such as banking, entertainment , oil & gas which is exposing us to a variety of risks and market trends. Below is our portfolio and $100,000 is equally distributed amongst these stocks.

ASML -> ASML Holding N.V.

GS-> Goldman Sachs Group Inc.

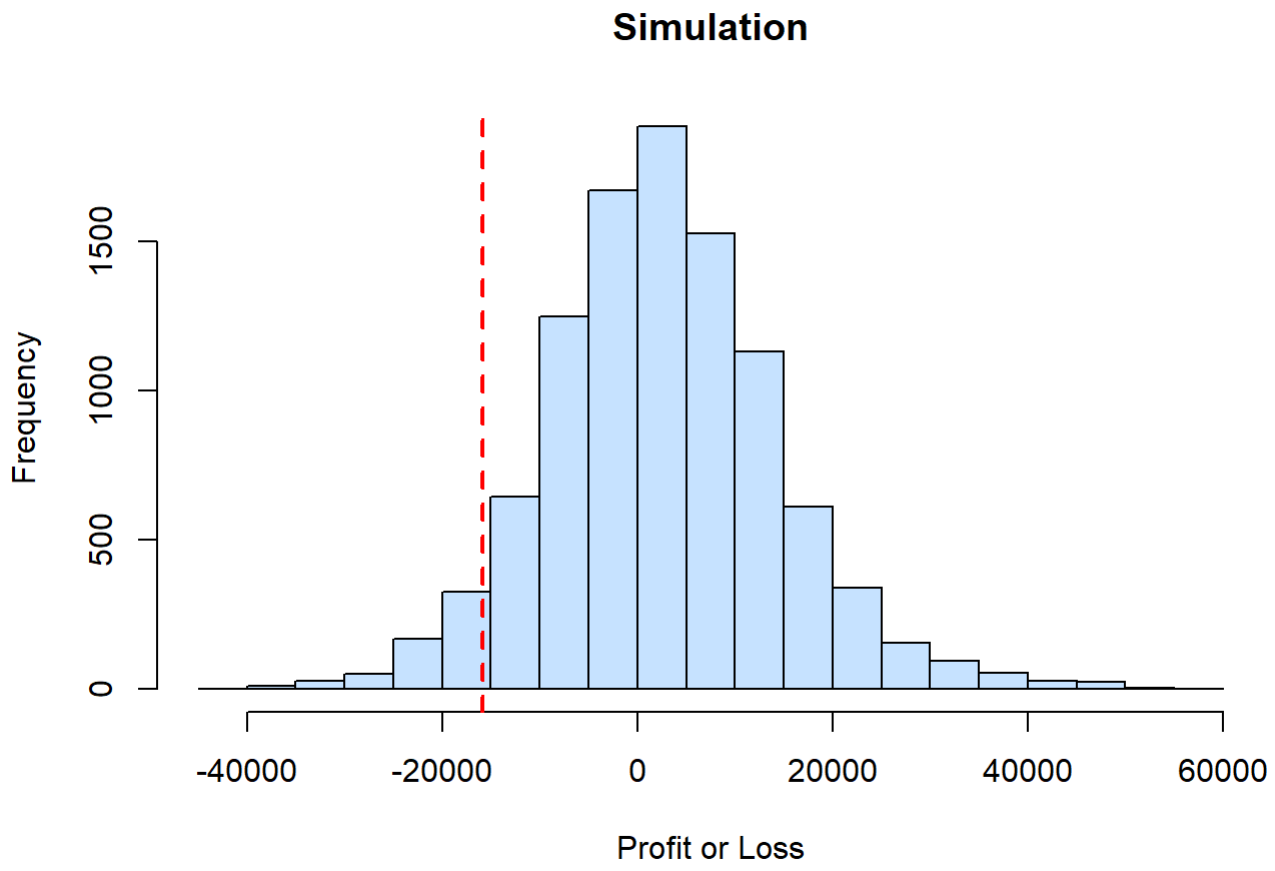
MTDR-> Matador Resources Company

AX-> Axos Financial Inc

CZR-> Caesars Entertainment, Inc.

Below is the profit simulation using bootstrap re-sampling.

We got a expected (mean) return of $2,727 and we are 95% confident that our loss in 20 day period won't exceed $15,697 i.e our VaR at 5%.



Clustering and PCA

```
##
## Attaching package: 'purrr'
```

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

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

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

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

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

```
## The following object is masked from 'package:reshape':
##
##     melt
```

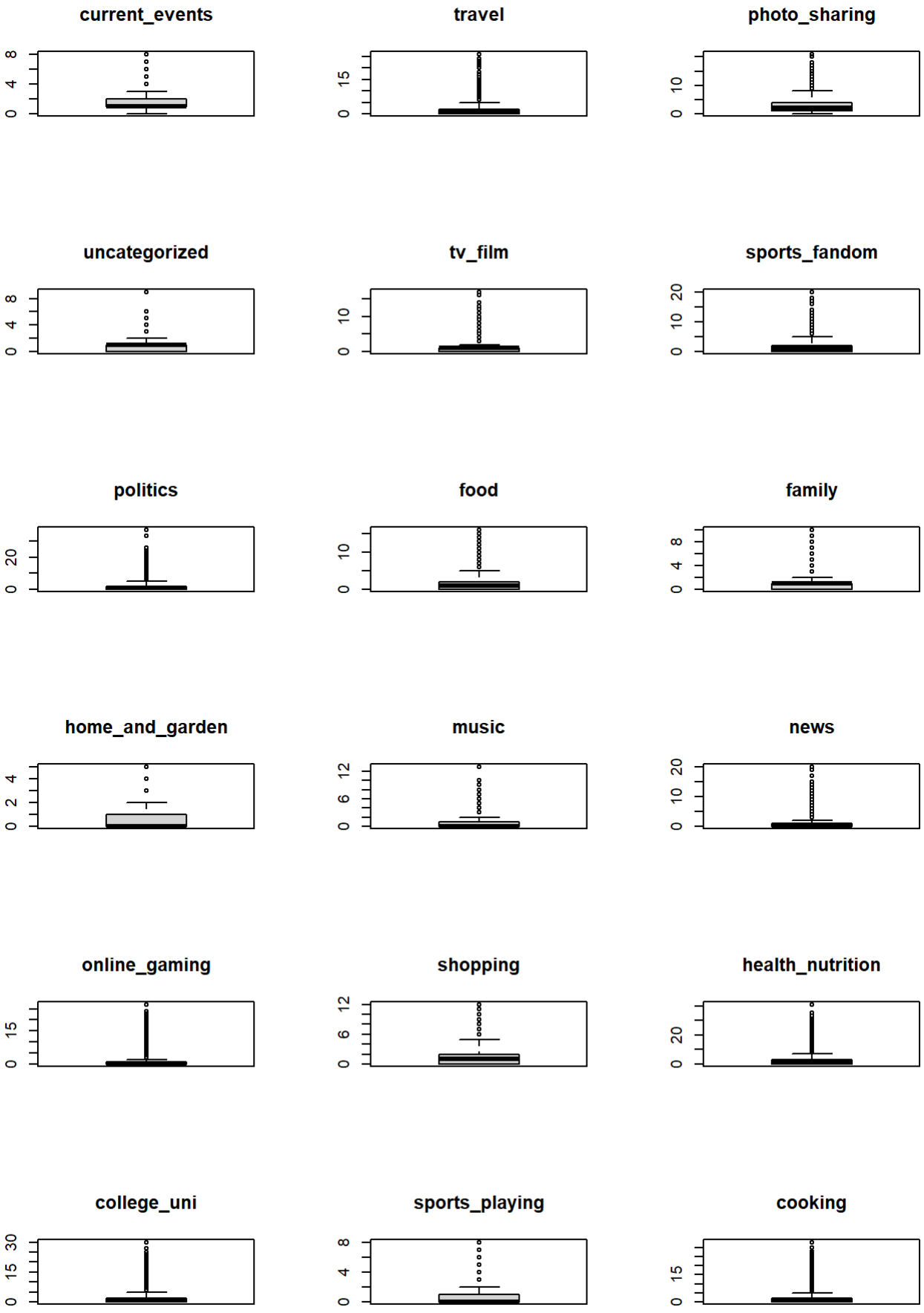
```
## The following objects are masked from 'package:dplyr':
##
##     between, first, last
```

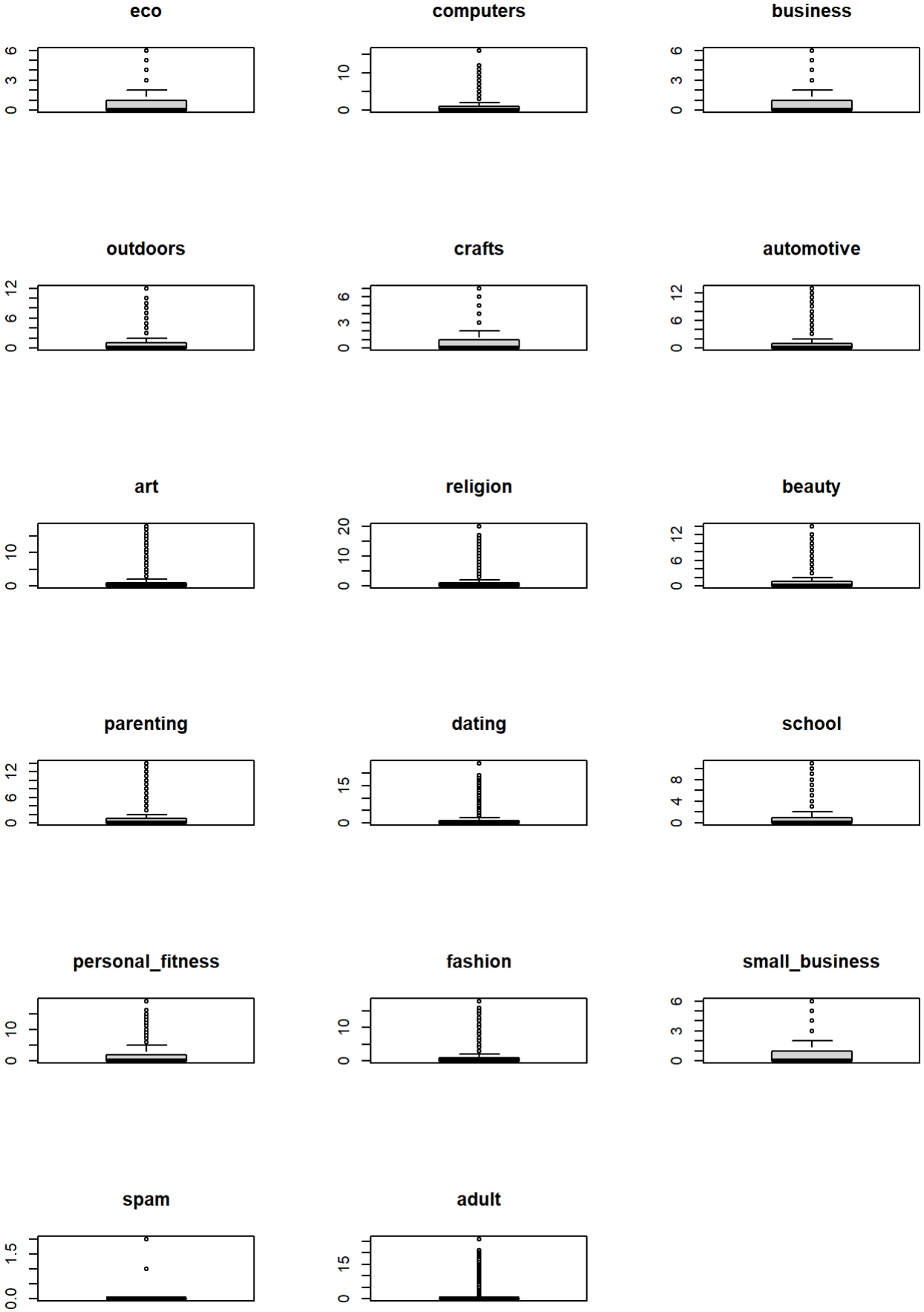
```
##     chatter       current_events      travel       photo_sharing
##  Min.   : 0.000   Min.   :0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 2.000   1st Qu.:1.000   1st Qu.: 0.000   1st Qu.: 1.000
##  Median : 3.000   Median :1.000   Median : 1.000   Median : 2.000
##  Mean   : 4.399   Mean   :1.526   Mean   : 1.585   Mean   : 2.697
##  3rd Qu.: 6.000   3rd Qu.:2.000   3rd Qu.: 2.000   3rd Qu.: 4.000
##  Max.   :26.000   Max.   :8.000   Max.   :26.000   Max.   :21.000
##  uncategorized      tv_film      sports_fandom       politics
##  Min.   :0.000   Min.   : 0.00   Min.   : 0.000   Min.   : 0.000
##  1st Qu.:0.000   1st Qu.: 0.00   1st Qu.: 0.000   1st Qu.: 0.000
##  Median :1.000   Median : 1.00   Median : 1.000   Median : 1.000
##  Mean   :0.813   Mean   : 1.07   Mean   : 1.594   Mean   : 1.789
##  3rd Qu.:1.000   3rd Qu.: 1.00   3rd Qu.: 2.000   3rd Qu.: 2.000
##  Max.   :9.000   Max.   :17.00   Max.   :20.000   Max.   :37.000
##       food            family        home_and_garden      music
##  Min.   : 0.000   Min.   : 0.0000   Min.   :0.0000   Min.   : 0.0000
##  1st Qu.: 0.000   1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.: 0.0000
##  Median : 1.000   Median : 1.0000   Median :0.0000   Median : 0.0000
##  Mean   : 1.397   Mean   : 0.8639   Mean   :0.5207   Mean   : 0.6793
##  3rd Qu.: 2.000   3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000
##  Max.   :16.000   Max.   :10.0000   Max.   :5.0000   Max.   :13.0000
##       news        online_gaming       shopping      health_nutrition
##  Min.   : 0.000   Min.   : 0.000   Min.   : 0.000   Min.   : 0.000
##  1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000   1st Qu.: 0.000
##  Median : 0.000   Median : 0.000   Median : 1.000   Median : 1.000
##  Mean   : 1.206   Mean   : 1.209   Mean   : 1.389   Mean   : 2.567
##  3rd Qu.: 1.000   3rd Qu.: 1.000   3rd Qu.: 2.000   3rd Qu.: 3.000
##  Max.   :20.000   Max.   :27.000   Max.   :12.000   Max.   :41.000
##   college_uni     sports_playing      cooking            eco
##  Min.   : 0.000   Min.   :0.0000   Min.   : 0.000   Min.   :0.0000
##  1st Qu.: 0.000   1st Qu.:0.0000   1st Qu.: 0.000   1st Qu.:0.0000
##  Median : 1.000   Median :0.0000   Median : 1.000   Median :0.0000
##  Mean   : 1.549   Mean   :0.6392   Mean   : 1.998   Mean   :0.5123
##  3rd Qu.: 2.000   3rd Qu.:1.0000   3rd Qu.: 2.000   3rd Qu.:1.0000
##  Max.   :30.000   Max.   :8.0000   Max.   :33.000   Max.   :6.0000
##    computers          business         outdoors           crafts
##  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   : 0.6491   Mean   :0.4232   Mean   : 0.7827   Mean   :0.5159
##  3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.: 1.0000   3rd Qu.:1.0000
##  Max.   :16.0000   Max.   :6.0000   Max.   :12.0000   Max.   :7.0000
##    automotive           art             religion          beauty
##  Min.   : 0.0000   Min.   : 0.0000   Min.   : 0.000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.: 0.0000   1st Qu.: 0.000   1st Qu.: 0.0000
##  Median : 0.0000   Median : 0.0000   Median : 0.000   Median : 0.0000
##  Mean   : 0.8299   Mean   : 0.7248   Mean   : 1.095   Mean   : 0.7052
##  3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.000   3rd Qu.: 1.0000
##  Max.   :13.0000   Max.   :18.0000   Max.   :20.000   Max.   :14.0000
##    parenting           dating            school        personal_fitness
##  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   : 0.9213   Mean   : 0.7109   Mean   : 0.7677   Mean   : 1.462
##  3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 1.0000   3rd Qu.: 2.000
##  Max.   :14.0000   Max.   :24.0000   Max.   :11.0000   Max.   :19.000
##     fashion        small_business        spam             adult
##  Min.   : 0.0000   Min.   :0.0000   Min.   :0.00000   Min.   : 0.0000
##  1st Qu.: 0.0000   1st Qu.:0.0000   1st Qu.:0.00000   1st Qu.: 0.0000
##  Median : 0.0000   Median :0.0000   Median :0.00000   Median : 0.0000
##  Mean   : 0.9966   Mean   :0.3363   Mean   :0.00647   Mean   : 0.4033
##  3rd Qu.: 1.0000   3rd Qu.:1.0000   3rd Qu.:0.00000   3rd Qu.: 0.0000
##  Max.   :18.0000   Max.   :6.0000   Max.   :2.00000   Max.   :26.0000
```

No missing values/ null values were observed in any of the features.

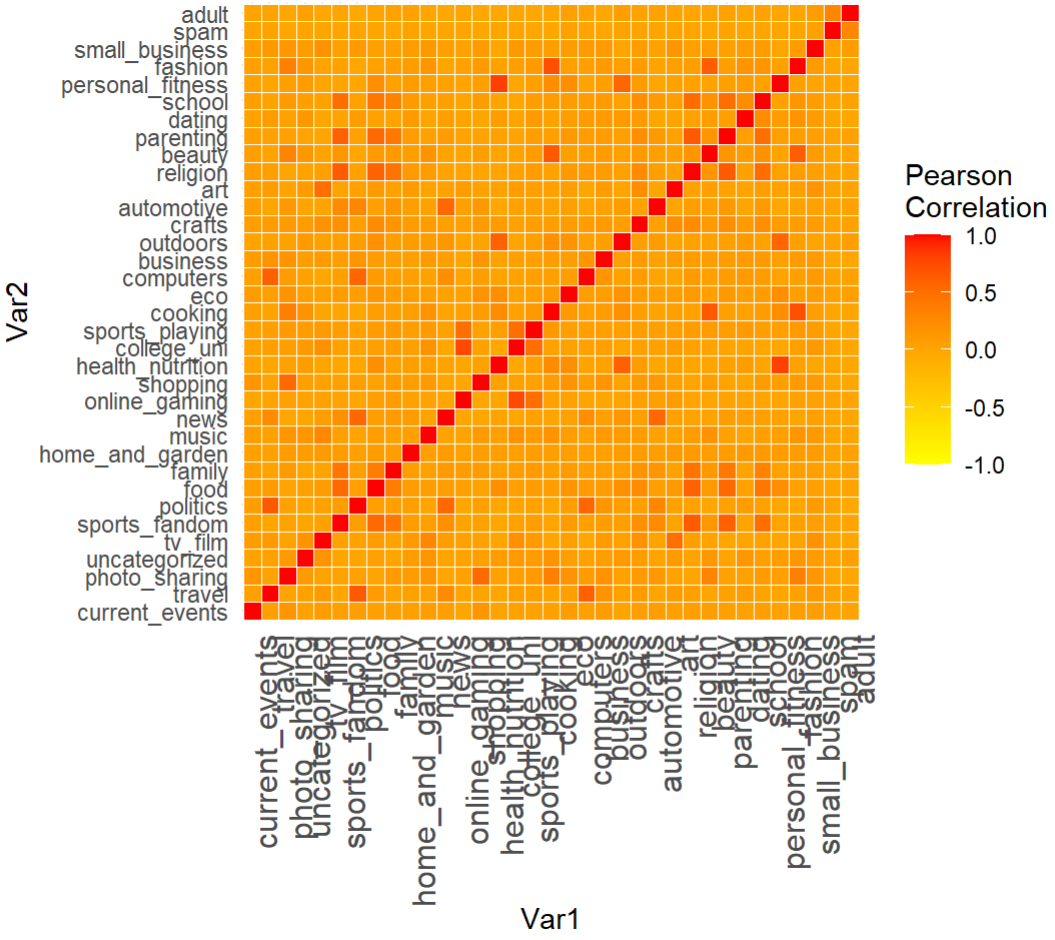
The following box plots help us to get a sense of the outliers and of there is a lot of variation in the feature. In case of high variance, features may not help in explaining the variability of the dependent variable.


current_events
0 4 8
travel
0 15
photo_sharing
0 10
uncategorized
0 4 8
tv_film
0 10
sports_fandom
0 10 20
politics
0 20
food
0 10
family
0 4 8
home_and_garden
0 2 4
music
0 6 12
news
0 10 20
online_gaming
0 15
shopping
0 6 12
health_nutrition
0 20
college_uni
0 15 30
sports_playing
0 4 8
cooking
0 15


eco
0 3 6
computers
0 10
business
0 3 6
outdoors
0 6 12
crafts
0 3 6
automotive
0 6 12
art
0 10
religion
0 10 20
beauty
0 6 12
parenting
0 6 12
dating
0 15
school
0 4 8
personal_fitness
0 10
fashion
0 10
small_business
0 3 6
spam
0.0 1.5
adult
0 15

From the below heatmap to understand bi-variate relations, we can infer that only few features have a correlation higher than 0.5

```
## Warning in melt(cormat): The melt generic in data.table has been passed a matrix
## and will attempt to redirect to the relevant reshape2 method; please note that
## reshape2 is deprecated, and this redirection is now deprecated as well. To
## continue using melt methods from reshape2 while both libraries are attached,
## e.g. melt.list, you can prepend the namespace like reshape2::melt(cormat). In
## the next version, this warning will become an error.
```

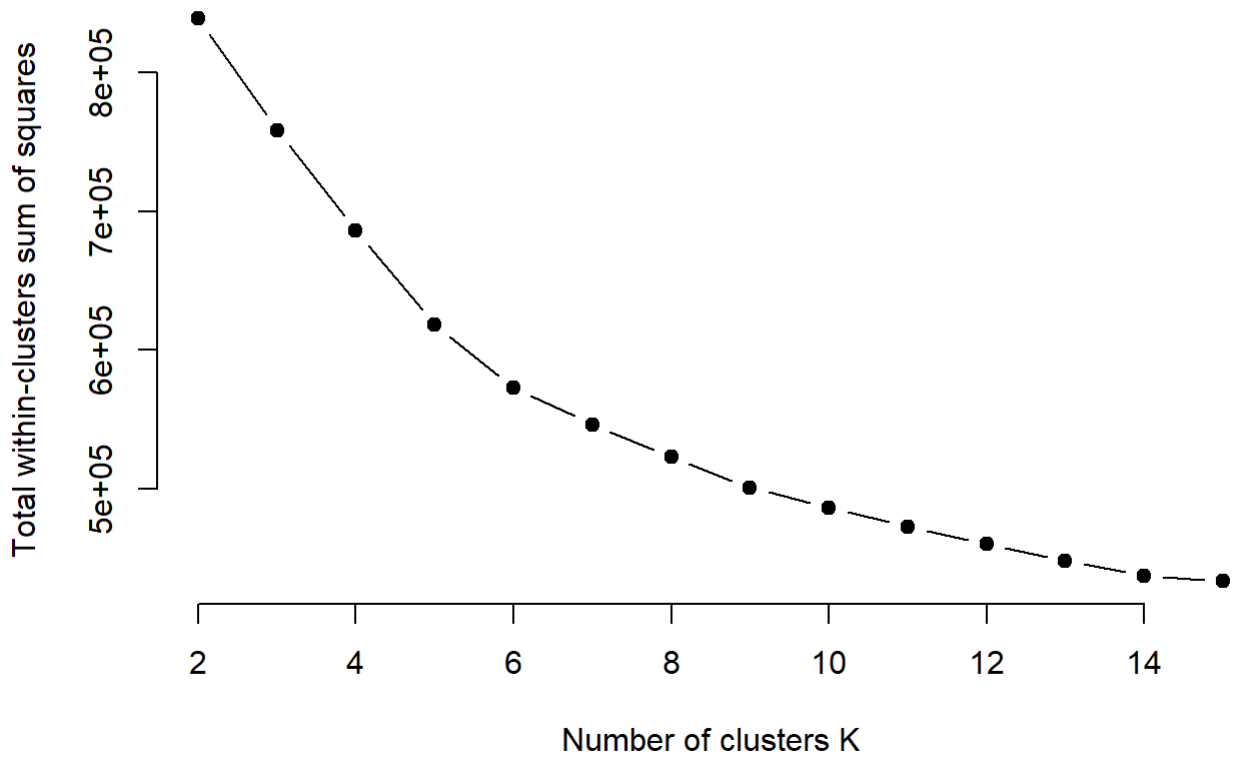


Dropping the “uncategorised” and “chatter” features as they will not reflect the customers’ behavior and are rather a function of the annotators work

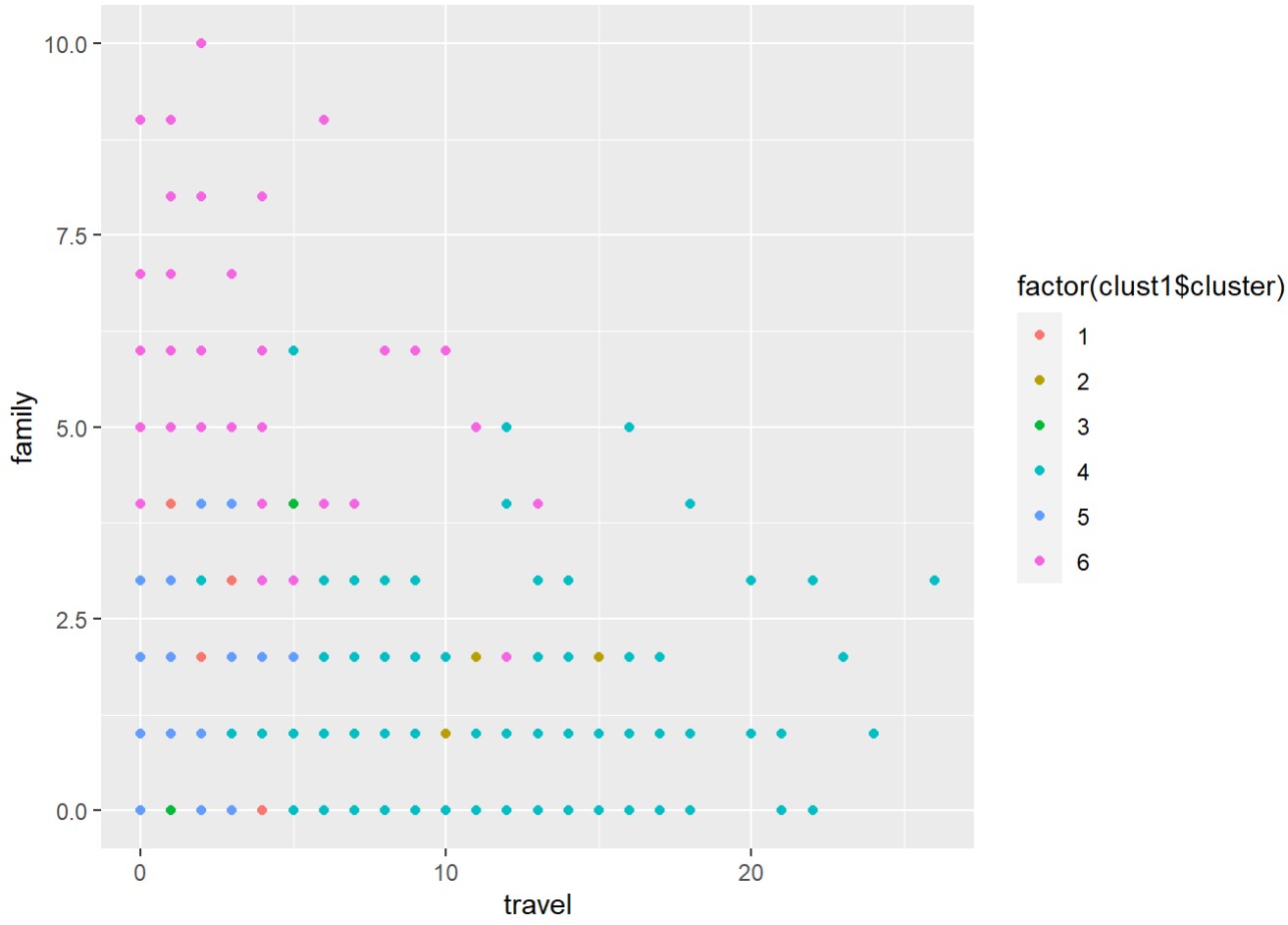
As we have a large number of features and ~8K data points, we choose to go ahead with K means clustering. To decide the number of clusters, we are plotting the WSS at different number of clusters ranging from 2 to 15.

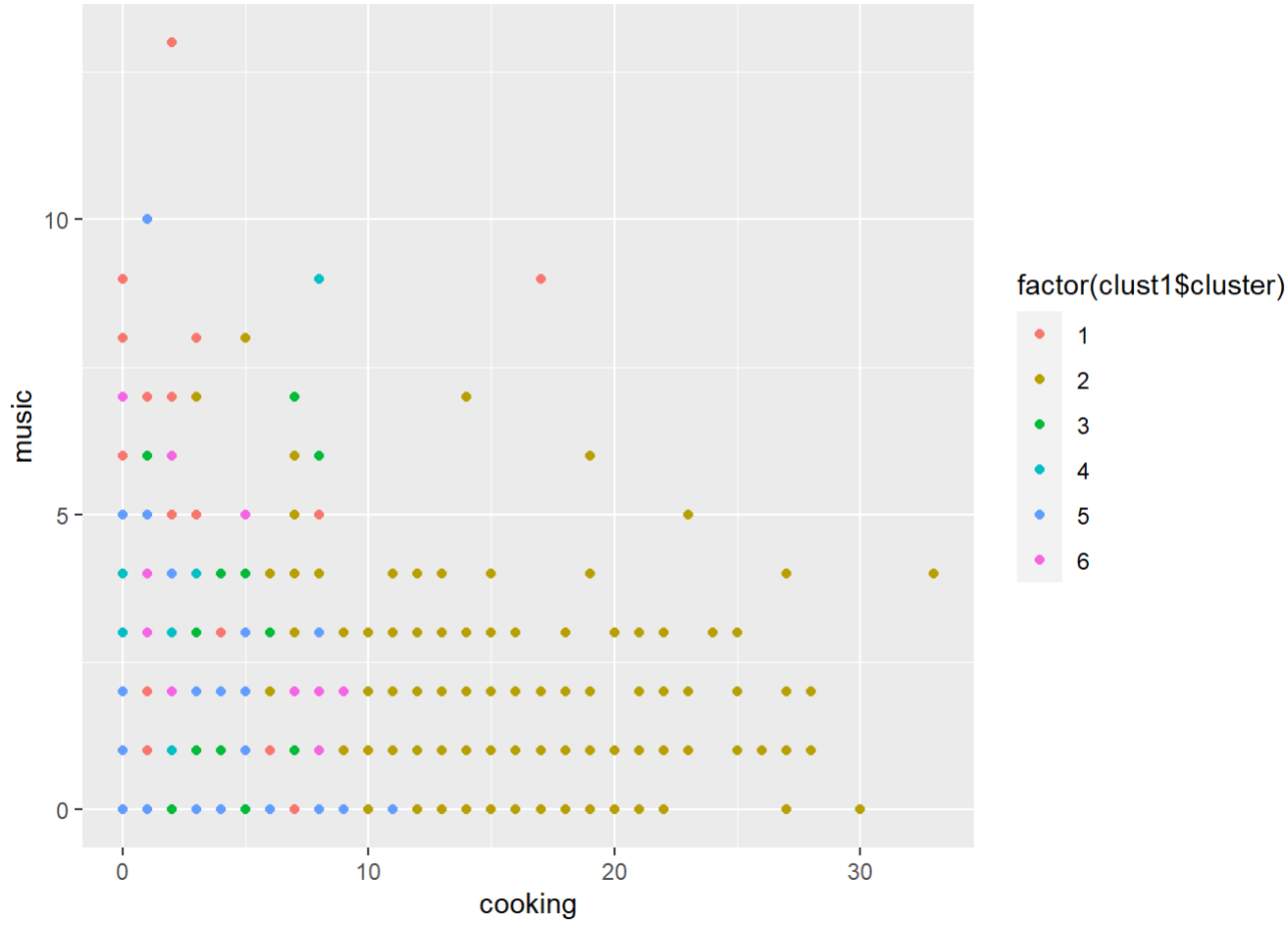
We see a clear elbow at K=6, thus we will use this to run the K-Means model

```
## Warning: Quick-TRANSfer stage steps exceeded maximum (= 394100)
```

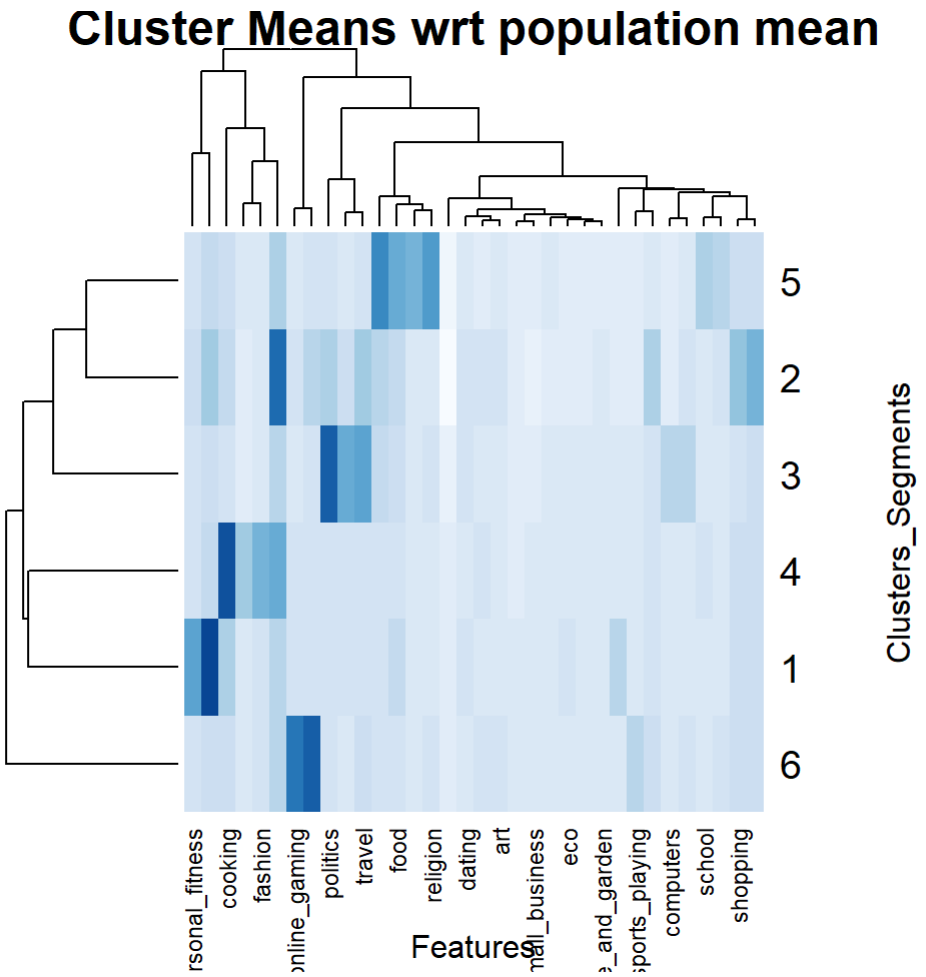


Plotting the clusters on a 2D plot to understand behavior is not possible due to the sheer number of features.


10.0
7.5
5.0
2.5
0.0
family
0
10
20
travel
factor(clust1$cluster)
1
2
3
4
5
6


10
5
0
music
0
10
20
30
cooking
factor(clust1$cluster)
1
2
3
4
5
6

Here, we are plotting the difference between the mean of the population and the mean of each cluster to understand how the behavioral characteristics of the cluster differ from the complete population and set the cluster population apart from the population.



Below are the key observations for the 6 clusters formed -

1. Older People with families - Food, Religion, Parenting, Sports Fandom
2. Tech and News enthusiasts, possibly people in late 20s or 30s - Politics, Travel, News, Automotive, Computers
3. Younger crowd/ Highly Active Social media users - Cooking, Beauty, Fashion, Photo sharing
4. Health and Fitness Enthusiasts - Outdoors, Health Nutrition, Personal Fitness
5. College students - Sports_Playing, College_Uni, Online_Gaming
6. No clear segment description - Photo Sharing is the most dominant feature. Other features that are slightly above mean are : current events, shopping, tv_films, college_uni, travel, politics

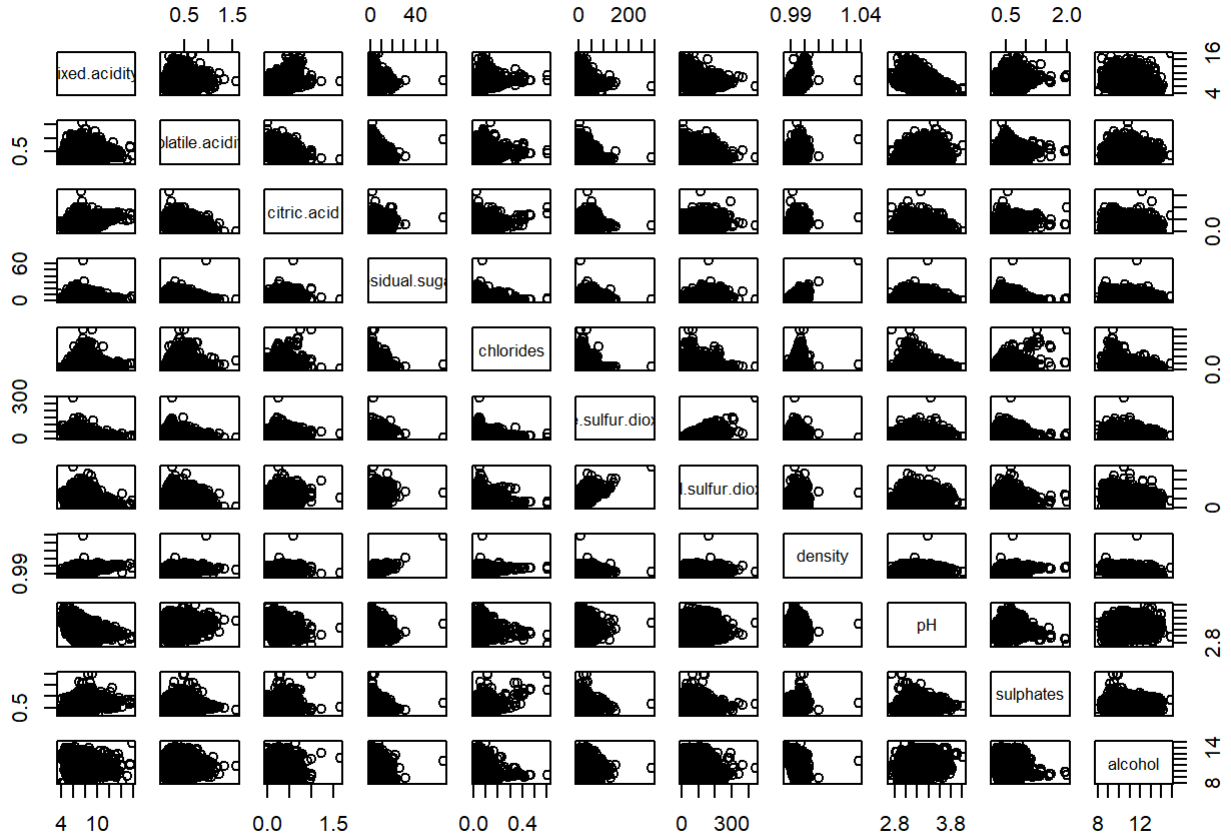
Market segmentation

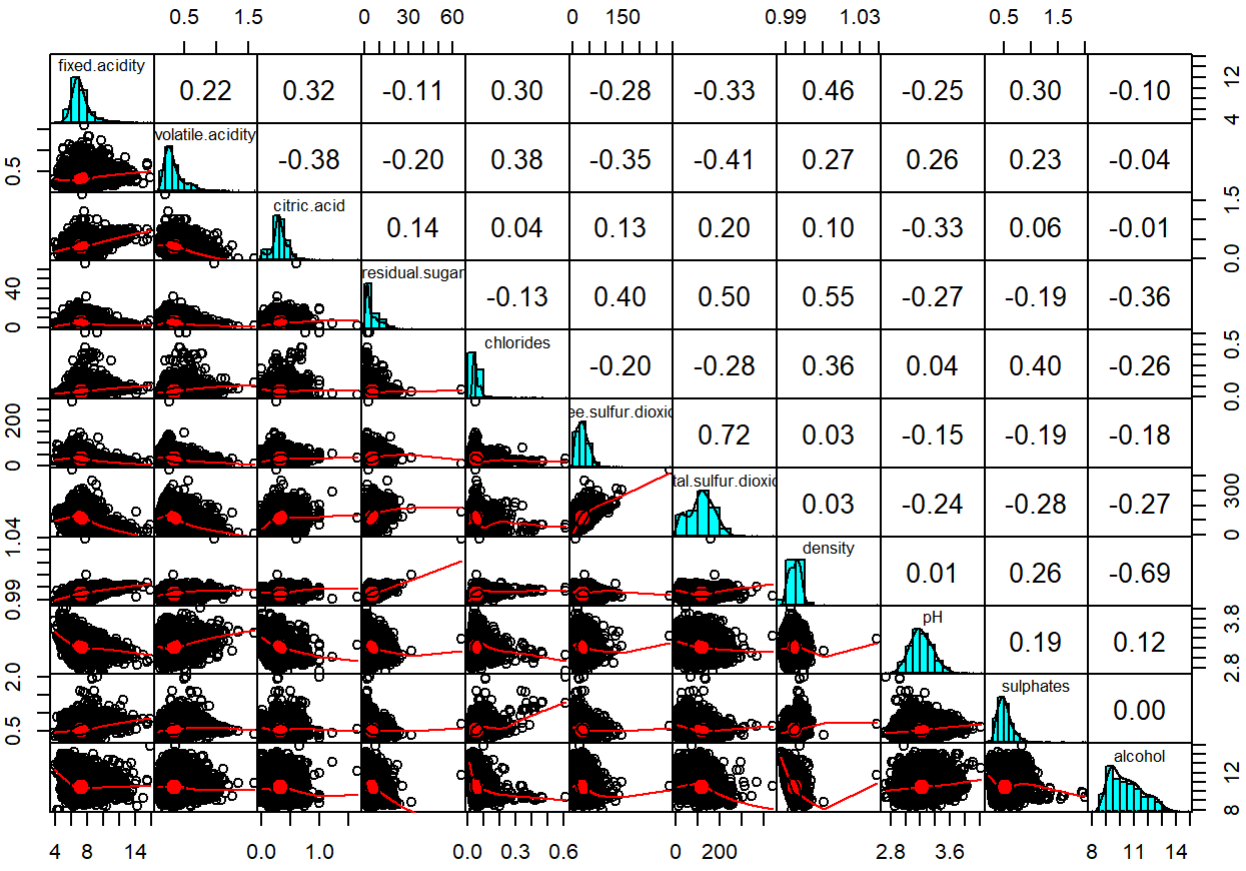
```
##   fixed.acidity volatile.acidity citric.acid residual.sugar chlorides
## 1           7.4             0.70        0.00            1.9     0.076
## 2           7.8             0.88        0.00            2.6     0.098
## 3           7.8             0.76        0.04            2.3     0.092
## 4          11.2             0.28        0.56            1.9     0.075
## 5           7.4             0.70        0.00            1.9     0.076
## 6           7.4             0.66        0.00            1.8     0.075
##   free.sulfur.dioxide total.sulfur.dioxide density   pH sulphates alcohol
## 1                  11                   34  0.9978 3.51      0.56     9.4
## 2                  25                   67  0.9968 3.20      0.68     9.8
## 3                  15                   54  0.9970 3.26      0.65     9.8
## 4                  17                   60  0.9980 3.16      0.58     9.8
## 5                  11                   34  0.9978 3.51      0.56     9.4
## 6                  13                   40  0.9978 3.51      0.56     9.4
##   quality color
## 1       5   red
## 2       5   red
## 3       5   red
## 4       6   red
## 5       5   red
## 6       5   red
```

```
## [1] 6497   13
```

```
##  [1] "fixed.acidity"        "volatile.acidity"     "citric.acid"
##  [4] "residual.sugar"       "chlorides"            "free.sulfur.dioxide"
##  [7] "total.sulfur.dioxide" "density"              "pH"
## [10] "sulphates"            "alcohol"              "quality"
## [13] "color"
```

The below pairwise plots help understand the bi-variate relation between the available X variables.


fixed.acidity
volatile.acidity
citric.acid
residual.sugar
chlorides
free.sulfur.dioxide
total.sulfur.dioxide
density
pH
sulphates
alcohol


fixed.acidity
0.22
0.32
-0.11
0.30
-0.28
-0.33
0.46
-0.25
0.30
-0.10
volatile.acidity
-0.38
-0.20
0.38
-0.35
-0.41
0.27
0.26
0.23
-0.04
citric.acid
0.14
0.04
0.13
0.20
0.10
-0.33
0.06
-0.01
residual.sugar
-0.13
0.40
0.50
0.55
-0.27
-0.19
-0.36
chlorides
-0.20
-0.28
0.36
0.04
0.40
-0.26
free.sulfur.dioxide
0.72
0.03
-0.15
-0.19
-0.18
total.sulfur.dioxide
0.03
-0.24
-0.28
-0.27
density
0.01
0.26
-0.69
pH
0.19
0.12
sulphates
0.00
alcohol

High covariance between X variables helps in achieving better Principal Components. Below table help us understand the same.

```
##                       fixed.acidity volatile.acidity   citric.acid
## fixed.acidity           1.68074049     0.0467451842  6.112206e-02
## volatile.acidity        0.04674518     0.0271051686 -9.043060e-03
## citric.acid             0.06112206    -0.0090430597  2.111728e-02
## residual.sugar         -0.69072041    -0.1535371274  9.848991e-02
## chlorides               0.01354363     0.0021751806  1.985392e-04
## free.sulfur.dioxide    -6.50600262    -1.0302424657  3.433720e-01
## total.sulfur.dioxide  -24.11203016    -3.8569326160  1.603646e+00
## density                 0.00178405     0.0001339362  4.190011e-05
## pH                     -0.05267540     0.0069210740 -7.706052e-03
## sulphates               0.05779170     0.0055363458  1.215219e-03
## alcohol                -0.14759400    -0.0073912113 -1.818756e-03
##                       residual.sugar     chlorides free.sulfur.dioxide
## fixed.acidity           -0.690720406  1.354363e-02         -6.50600262
## volatile.acidity        -0.153537127  2.175181e-03         -1.03024247
## citric.acid              0.098489906  1.985392e-04          0.34337205
## residual.sugar          22.636696458 -2.149219e-02         34.02168454
## chlorides               -0.021492189  1.227353e-03         -0.12128380
## free.sulfur.dioxide     34.021684542 -1.212838e-01        315.04119227
## total.sulfur.dioxide   133.244854379 -5.537142e-01        723.26197160
## density                  0.007882813  3.809423e-05          0.00136877
## pH                      -0.204498058  2.518380e-04         -0.41624898
## sulphates               -0.131634584  2.062309e-03         -0.49775610
## alcohol                 -2.039566839 -1.073521e-02         -3.80716486
##                       total.sulfur.dioxide       density            pH
## fixed.acidity                -2.411203e+01  1.784050e-03 -5.267540e-02
## volatile.acidity             -3.856933e+00  1.339362e-04  6.921074e-03
## citric.acid                   1.603646e+00  4.190011e-05 -7.706052e-03
## residual.sugar                1.332449e+02  7.882813e-03 -2.044981e-01
## chlorides                    -5.537142e-01  3.809423e-05  2.518380e-04
## free.sulfur.dioxide           7.232620e+02  1.368770e-03 -4.162490e-01
## total.sulfur.dioxide          3.194720e+03  5.490564e-03 -2.166696e+00
## density                       5.490564e-03  8.992040e-06  5.634423e-06
## pH                           -2.166696e+00  5.634423e-06  2.585252e-02
## sulphates                    -2.319079e+00  1.157845e-04  4.596760e-03
## alcohol                      -1.791465e+01 -2.456181e-03  2.325216e-02
##                          sulphates       alcohol
## fixed.acidity          0.0577916979 -1.475940e-01
## volatile.acidity       0.0055363458 -7.391211e-03
## citric.acid            0.0012152190 -1.818756e-03
## residual.sugar        -0.1316345842 -2.039567e+00
## chlorides              0.0020623093 -1.073521e-02
## free.sulfur.dioxide   -0.4977561009 -3.807165e+00
## total.sulfur.dioxide -2.3190787089 -1.791465e+01
## density                0.0001157845 -2.456181e-03
## pH                     0.0045967600  2.325216e-02
## sulphates              0.0221431880 -5.376291e-04
## alcohol               -0.0005376291  1.422561e+00
```

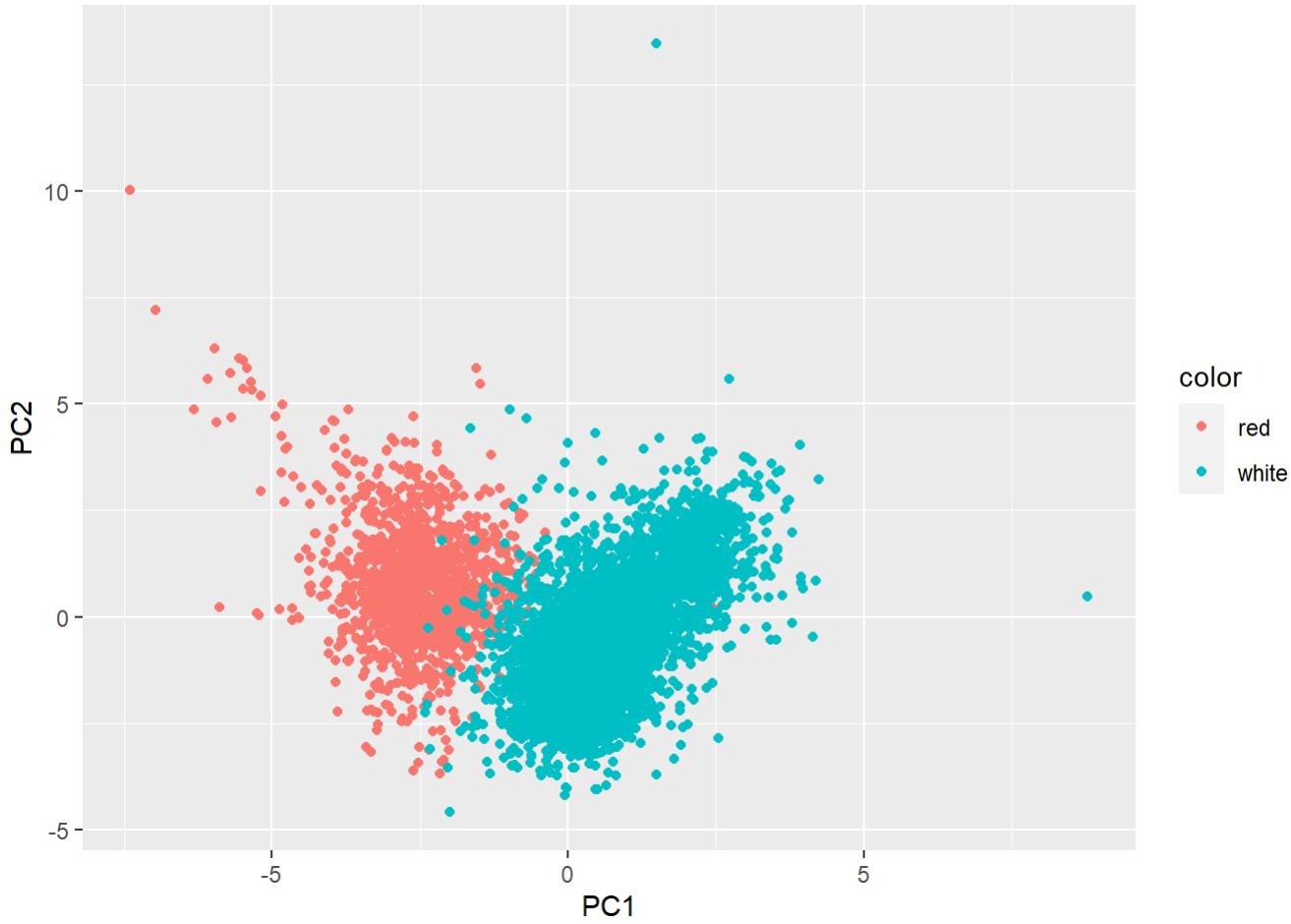
The below table helps us get a sense of the 11 Principal Components formed and the variability explained by them. We can see that the first 4 components can explain 73% variability in the data. After which each additional feature gives a lesser incremental value in variance explainability.

```
## 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
```

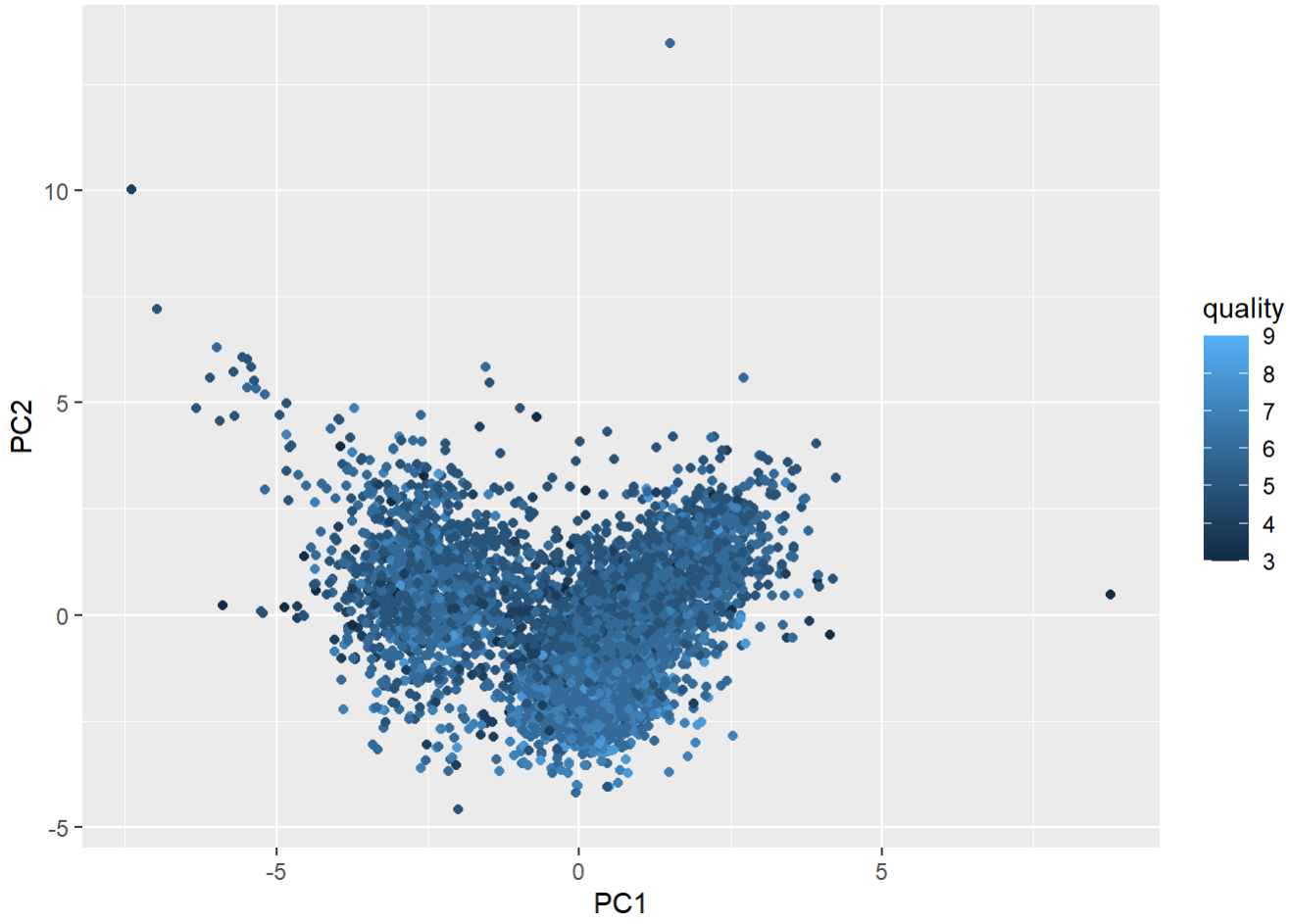
```
## [1] 6497   11
```

```
##         PC1       PC2      PC3       PC4        PC5        PC6         PC7
## 1 -3.205749 0.4164913 2.722027 0.7967162 -0.2028619  0.2273453 -0.32552850
## 2 -3.038817 1.1073769 2.046795 0.7701656  1.3225536 -1.6549941  0.05955408
## 3 -3.071657 0.8788968 1.742445 0.8021955  0.7620531 -0.8483083  0.16765673
##         PC8         PC9       PC10        PC11 fixed.acidity volatile.acidity
## 1 0.5672348  0.07122341 0.10803795  0.02745798           7.4             0.70
## 2 0.5145630 -0.42909559 0.26812827 -0.01547017           7.8             0.88
## 3 0.4209200 -0.27101087 0.08682522  0.05414179           7.8             0.76
##   citric.acid residual.sugar chlorides free.sulfur.dioxide total.sulfur.dioxide
## 1        0.00            1.9     0.076                  11                   34
## 2        0.00            2.6     0.098                  25                   67
## 3        0.04            2.3     0.092                  15                   54
##   density   pH sulphates alcohol quality color
## 1  0.9978 3.51      0.56     9.4       5   red
## 2  0.9968 3.20      0.68     9.8       5   red
## 3  0.9970 3.26      0.65     9.8       5   red
```

To get a visual sense of the the PCs and their differentiation of the Wine Color, we have plotted the below scatter plot with PC1 and PC2 (Since they explain 50% variability). We can see a clear distinction between red and white wines.



However, no clear differentiation is observed in Wine Quality basis PC1 And PC2



K-Means Clustering

We are running the K-Means clustering model with all the available features.

We start with using K=2 to identify clusters that can group Red and White wines separately.

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1    -0.2804833       -0.3953082   0.1143429      0.1998380 -0.3119753
## 2     0.8286464        1.1678795  -0.3378091     -0.5903919  0.9216848
##   free.sulfur.dioxide total.sulfur.dioxide    density         pH  sulphates
## 1           0.2814861            0.4018607 -0.2306934 -0.1920315 -0.2853595
## 2          -0.8316090           -1.1872380  0.6815493  0.5673286  0.8430523
##       alcohol
## 1  0.02562065
## 2 -0.07569241
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.85167903           0.27458385           0.33524928
##       residual.sugar            chlorides  free.sulfur.dioxide
##           6.39402555           0.04510424          35.52152864
## total.sulfur.dioxide              density                   pH
##         138.45848785           0.99400486           3.18762464
##            sulphates              alcohol
##           0.48880511          10.52235888
```

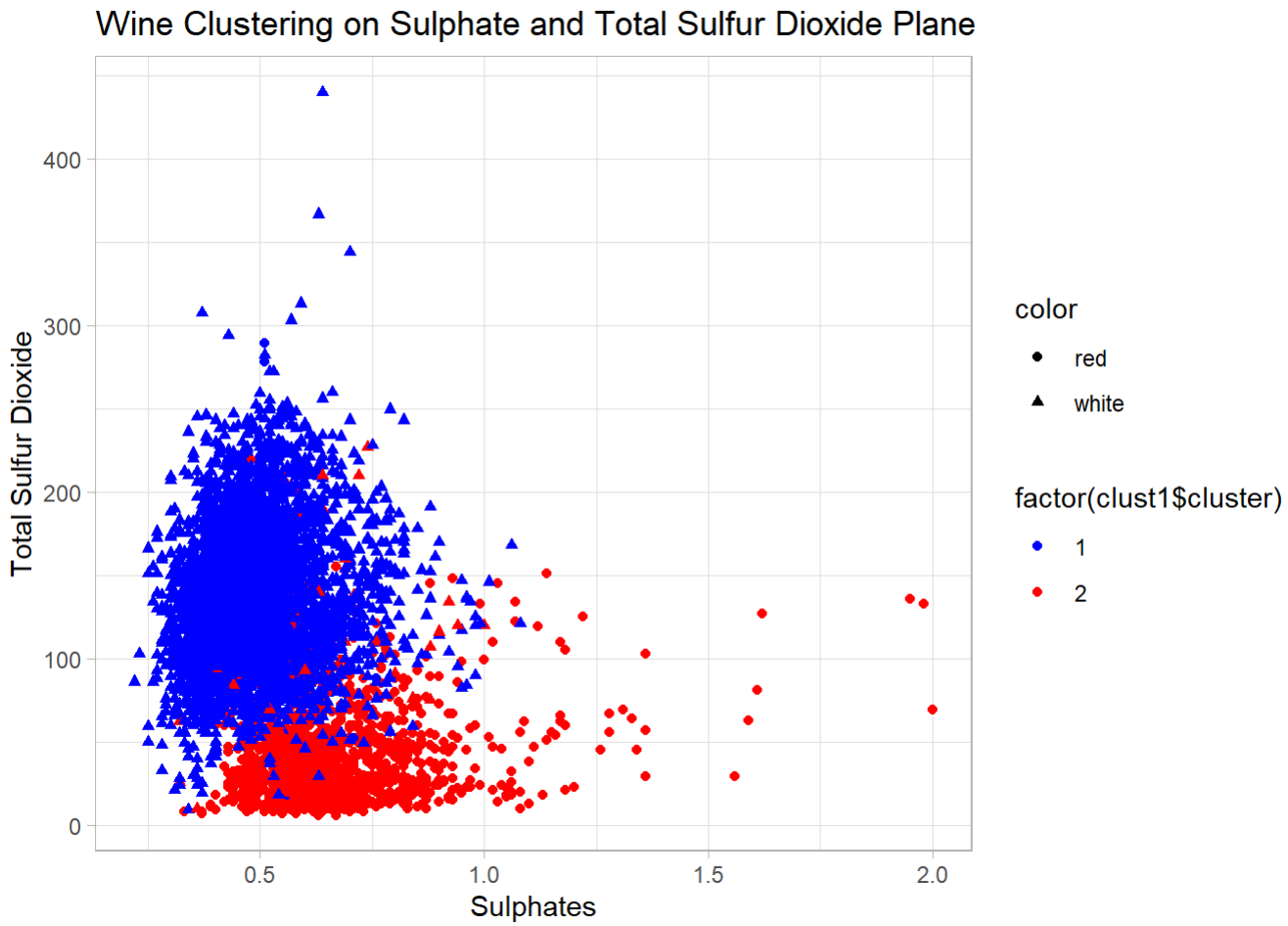
```
##        fixed.acidity     volatile.acidity          citric.acid
##            8.2895922            0.5319416            0.2695435
##       residual.sugar            chlorides  free.sulfur.dioxide
##            2.6342666            0.0883238           15.7647596
## total.sulfur.dioxide              density                   pH
##           48.6396835            0.9967404            3.3097200
##            sulphates              alcohol
##            0.6567194           10.4015216
```

The below confusion matrix indicates that we are predicting wine color with 99% accuracy.

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

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

The below scatter plot is showing an example plotted on 2 of the features, where we can see a clear distinction between red and white wine




```
## Warning: did not converge in 10 iterations
```

Now, we are running a model with K=7 to try and form clusters that will indicate Wine quality groups.

```
##   fixed.acidity volatile.acidity citric.acid residual.sugar  chlorides
## 1   -0.55828664       -0.2627045 -0.03800665     -0.4668366 -0.5923117
## 2   -0.17152985       -0.3520670  0.31703650      1.4642874 -0.1604910
## 3    0.08776382        1.6941374 -1.26666091     -0.6272135  0.6581520
## 4    0.12409998       -0.4754132  0.26320642     -0.3054724 -0.2241985
## 5    2.04888035        0.4270597  0.95009634     -0.5822833  0.7812790
## 6    0.78338135        1.1610561  0.74885032     -0.2525151  6.9430163
## 7   -0.59824549       -0.5065584 -0.16120993     -0.2649791 -0.2688307
##   free.sulfur.dioxide total.sulfur.dioxide    density           pH   sulphates
## 1         -0.09911421          -0.15157775 -1.3458179  0.004105864 -0.29498430
## 2          0.93069109           0.99591770  0.9183853 -0.503026231 -0.28011637
## 3         -0.79748224          -1.15886988  0.5012553  0.957881442  0.41494723
## 4         -0.27330327           0.05872296 -0.4784721 -0.775477963 -0.48884865
## 5         -0.91766038          -1.30732692  0.9545307  0.011548662  1.22776875
## 6         -0.43780488          -0.55648679  0.9098882 -0.694033817  2.98622679
## 7          0.39733684           0.54762591 -0.2894575  0.757922182  0.03182378
##       alcohol
## 1  1.43244651
## 2 -0.87652177
## 3 -0.25024574
## 4 -0.01837911
## 5  0.14206767
## 6 -0.77057475
## 7 -0.18084111
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.49152542           0.29641525           0.31311017
##       residual.sugar            chlorides  free.sulfur.dioxide
##           3.22211864           0.03528305          28.76610169
## total.sulfur.dioxide              density                   pH
##         107.17711864           0.99066097           3.21916102
##            sulphates              alcohol
##           0.48737288          12.20029661
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.99292998           0.28170292           0.36470428
##       residual.sugar            chlorides  free.sulfur.dioxide
##          12.41002719           0.05041128          47.04452753
## total.sulfur.dioxide              density                   pH
##         172.03569001           0.99745057           3.13762067
##            sulphates              alcohol
##           0.48958532           9.44636302
```

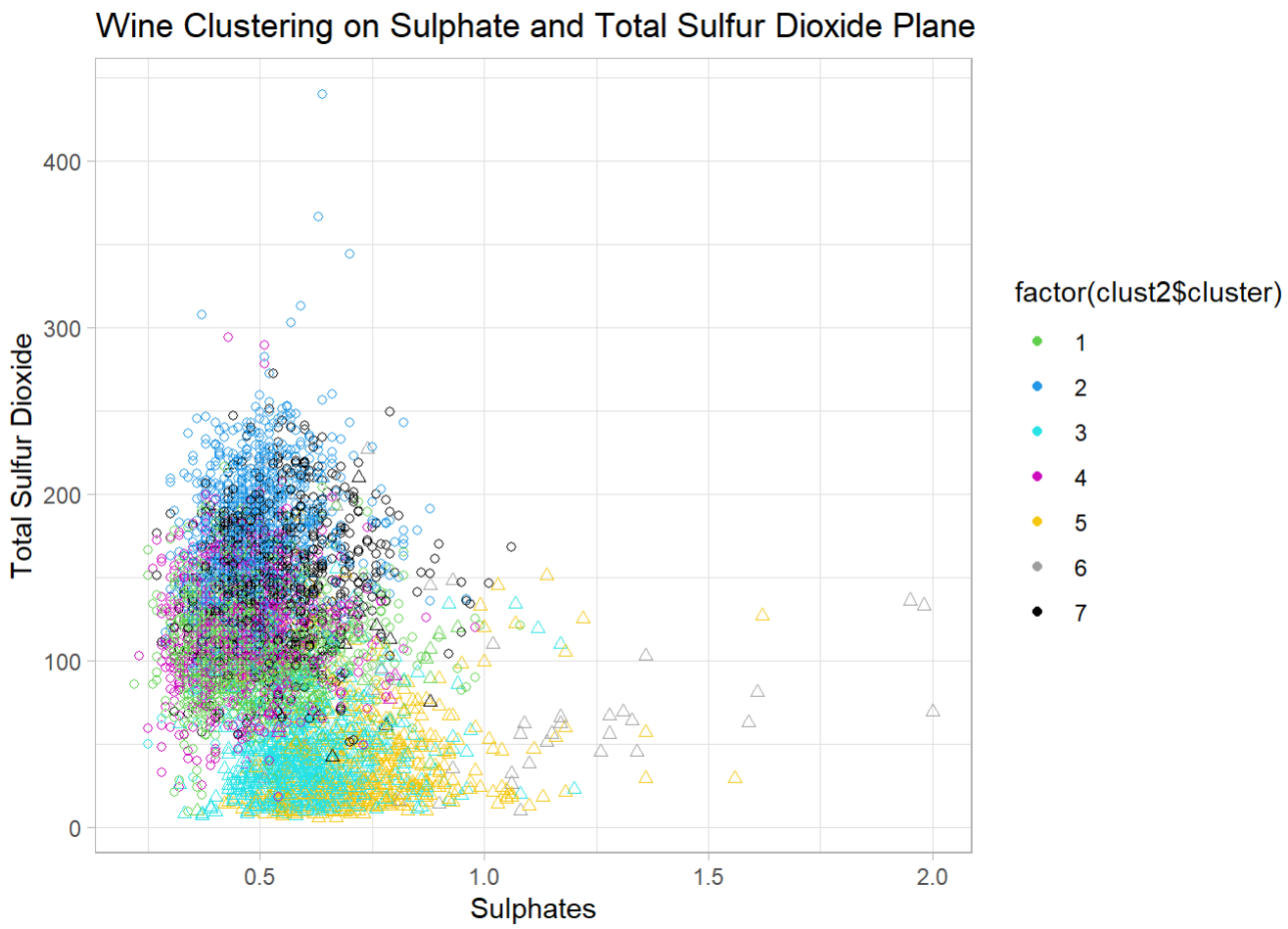
```
##        fixed.acidity     volatile.acidity          citric.acid
##            7.3290870            0.6185828            0.1345648
##       residual.sugar            chlorides  free.sulfur.dioxide
##            2.4590764            0.0790913           16.3704883
## total.sulfur.dioxide              density                   pH
##           50.2430998            0.9961997            3.3725159
##            sulphates              alcohol
##            0.5930149           10.1933298
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           7.37619447           0.26139564           0.35688181
##       residual.sugar            chlorides  free.sulfur.dioxide
##           3.98985750           0.04817938          25.67435038
## total.sulfur.dioxide              density                   pH
##         119.06370495           0.99326185           3.09381391
##            sulphates              alcohol
##           0.45852473          10.46987985
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           9.87154472           0.40997561           0.45669919
##       residual.sugar            chlorides  free.sulfur.dioxide
##           2.67284553           0.08340488          14.23739837
## total.sulfur.dioxide              density                   pH
##          41.85203252           0.99755896           3.22035772
##            sulphates              alcohol
##           0.71396748          10.66124661
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##            8.2309091            0.5308182            0.4274545
##       residual.sugar            chlorides  free.sulfur.dioxide
##            4.2418182            0.2992727           22.7545455
## total.sulfur.dioxide              density                   pH
##           84.2909091            0.9974251            3.1069091
##            sulphates              alcohol
##            0.9756364            9.5727273
```

```
##        fixed.acidity     volatile.acidity          citric.acid
##           6.43972142           0.25626801           0.29520653
##       residual.sugar            chlorides  free.sulfur.dioxide
##           4.18251681           0.04661575          37.57780980
## total.sulfur.dioxide              density                   pH
##          146.69740634           0.99382865           3.34036503
##            sulphates              alcohol
##           0.53600384          10.27610951
```



Wine Color -

1. PCA - We can easily conclude that wine color can be differentiated basis PCA.
2. K-means - This as well gives us good results which can be concluded from the Confusion matrix as a 99% accuracy is achieved.

Wine Quality -

1. PCA - PCA fails to give us a clear distinction between the 7 wine quality ratings available (As seen in the plot above)
2. K Means - The plot above indicates that clusters are overlapping one another thus it is not possible to differentiate between the clusters. Also, confusion matrix cannot be created here since, we cannot map the Wine Quality ratings to the clusters as K-means gives clusters without labels.

The Reuters corpus

For this problem we will be identifying authorship of documents using classification. We will use a Naive Bayes classification initially and then try to improve the accuracy using alternative methods.

Reading the TRAIN DATA into a corpus

```
# code taken from Prof Scott's R file
author_dirs = Sys.glob('data/ReutersC50/C50train/*')

file_list = NULL
labels = NULL
for(author in author_dirs) {
    author_name = substring(author, first=26)
    files_to_add = Sys.glob(paste0(author, '/*.txt'))
    file_list = append(file_list, files_to_add)
    labels = append(labels, rep(author_name, length(files_to_add)))
}

all_docs = lapply(file_list, readerPlain)


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


# creating the train corpus from all the files in train folder
train_Corpus = Corpus(VectorSource(all_docs))



# Preprocessing (removing stop words, numbers, white spaces and converting to lower)
train_Corpus = tm_map(train_Corpus, content_transformer(tolower)) # make everything lowercase
train_Corpus = tm_map(train_Corpus, content_transformer(removeNumbers)) # remove numbers
train_Corpus = tm_map(train_Corpus, content_transformer(removePunctuation)) # remove punctuation
train_Corpus = tm_map(train_Corpus, content_transformer(stripWhitespace)) ## remove excess white
-space
train_Corpus = tm_map(train_Corpus, content_transformer(removeWords), stopwords("SMART"))


#creating the document term matrix using tfidf weighting for normalization
DTM_train = DocumentTermMatrix(train_Corpus,control=list(weighting=weightTfIdf, bounds = list(gl
obal = c(5, Inf))))

DTM_train = removeSparseTerms(DTM_train, 0.95)

DTM_train = as.matrix(DTM_train)
DTM_train = as.data.frame(DTM_train)
```

Reading the TEST DATA into a corpus

```
author_dirs = Sys.glob('data/ReutersC50/C50test/*')
file_list = NULL
test_labels = NULL
author_names = NULL
for(author in author_dirs) {
  author_name = substring(author, first=25)
  author_names = append(author_names, author_name)
  files_to_add = Sys.glob(paste0(author, '/*.txt'))
  file_list = append(file_list, files_to_add)
  test_labels = append(test_labels, rep(author_name, length(files_to_add)))
}
all_docs = lapply(file_list, readerPlain)
names(all_docs) = file_list
names(all_docs) = sub('.txt', '', names(all_docs))

# performing a similar preprocessing as train
test_corpus = Corpus(VectorSource(all_docs))
test_corpus = tm_map(test_corpus, content_transformer(tolower))
test_corpus = tm_map(test_corpus, content_transformer(removeNumbers))
test_corpus = tm_map(test_corpus, content_transformer(removePunctuation))
test_corpus = tm_map(test_corpus, content_transformer(stripWhitespace))
test_corpus = tm_map(test_corpus, content_transformer(removeWords), stopwords("en"))

# creating document term matrix for test using tfidf weighting
DTM_test = DocumentTermMatrix(test_corpus, control=list(weighting=weightTfIdf, bounds = list(glo
bal = c(5, Inf))))
DTM_test = removeSparseTerms(DTM_test, 0.95)
DTM_test = as.matrix(DTM_test)
DTM_test = as.data.frame(DTM_test)
```

Naive Bayes for predicting authorship

```
set.seed(007)
# correcting train and test labels - removing any unwanted file
# paths from the name of authors
labels = sapply(str_split(labels, '/'), tail, 1)
test_labels = sapply(str_split(test_labels, '/'), tail, 1)

DTM_train = cbind(DTM_train,labels)
DTM_test = cbind(DTM_test,test_labels)

nB = naiveBayes(as.factor(labels)~., data=DTM_train)
nB_predictions = predict(nB, DTM_test[,-ncol(DTM_test)], type="class")


predicted_labels= factor(as.numeric(as.factor(nB_predictions)), levels=1:50)

obs_labels = factor(as.numeric(as.factor(DTM_test$test_labels)),levels=1:50)

caret::confusionMatrix(obs_labels, predicted_labels)$overall['Accuracy']
```

```
## Accuracy
##    0.402
```

```
# the overall accuracy is 40.2% for naive bayes


# calculating the best a worst predictions by naive bayes
cfm = caret::confusionMatrix(obs_labels, predicted_labels)
acc = as.tibble(cfm$byClass)
min_idx = which.min(acc$`Balanced Accuracy`)
max_idx = which.max(acc$`Balanced Accuracy`)


DTM_test$test_labels[as.numeric(as.factor(DTM_test$test_labels)) == min_idx][1]
```

```
## [1] "JaneMacartney"
```

```
# the worst predicted author is JaneMacartney
min(acc$`Balanced Accuracy`)
```

```
## [1] 0.524055
```

```
# with minimum accuracy of 52.40%


DTM_test$test_labels[as.numeric(as.factor(DTM_test$test_labels)) == max_idx][1]
```

```
## [1] "JimGilchrist"
```

```
# the best predicted author is JimGilchrist
max(acc$`Balanced Accuracy`)
```

```
## [1] 0.9622132
```

```
# with maximum accuracy of 96.22%
```

We will try to improve this accuracy using random forest

Random Forest for predicting authorship

```
set.seed(007)
registerDoParallel(cores = 6)
testTrees = c(10,50,75,100,200,400)
TreeClass = foreach( i = 1:length(testTrees),.combine = 'c') %dopar%
{
   model_RF = randomForest::randomForest(x=DTM_train[,!(colnames(DTM_train) == "labels")], y=as.
factor(labels),ntree = testTrees[i])
  pred_RF = predict(model_RF, data=DTM_test[,!(colnames(DTM_test) == "test_labels")])
  # print(pred_RF)
  predicted_labels= factor(as.numeric(pred_RF),levels=1:51)
  obs_labels = factor(as.numeric(as.factor(DTM_test$test_labels)),levels=1:51)
  caret::confusionMatrix(obs_labels, predicted_labels)$overall['Accuracy']
}

print(TreeClass)
```

```
##  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy  Accuracy
## 0.4504432 0.6768000 0.7028000 0.7216000 0.7564000 0.7768000
```

```
# 400 trees fetch an overall accuracy of >75% therefore we train random forest on 400 trees.

model_RF = randomForest::randomForest(x=DTM_train[,!(colnames(DTM_train) == "labels")], y=as.fac
tor(labels),ntree = 400)

pred_RF = predict(model_RF, data=DTM_test[,!(colnames(DTM_test) == "test_labels")])

predicted_labels= factor(as.numeric(pred_RF),levels=1:50)


obs_labels = factor(as.numeric(as.factor(DTM_test$test_labels)),levels=1:50)

caret::confusionMatrix(obs_labels, predicted_labels)$overall['Accuracy']
```

```
## Accuracy
##   0.7796
```

```
# the overall accuracy is 77.96% for RFM with 400 trees

# calculating the best a worst predictions by naive bayes
cfm = caret::confusionMatrix(obs_labels, predicted_labels)
acc = as.tibble(cfm$byClass)
min_idx = which.min(acc$`Balanced Accuracy`)
max_idx = which.max(acc$`Balanced Accuracy`)

DTM_test$test_labels[as.numeric(as.factor(DTM_test$test_labels)) == min_idx][1]
```

```
## [1] "ScottHillis"
```

```
# the worst predicted author is ScottHillis
min(acc$`Balanced Accuracy`)
```

```
## [1] 0.7012311
```

```
# with minimum accuracy of 70.12%


DTM_test$test_labels[as.numeric(as.factor(DTM_test$test_labels)) == max_idx][1]
```

```
## [1] "HeatherScoffield"
```

```
# the best predicted author is AlanCrosby
max(acc$`Balanced Accuracy`)
```

```
## [1] 0.9756441
```

```
# with maximum accuracy of 97.56%

# The accuracy for JaneMacartney (which was 52.40% in Naive Bayes) has increased significantly t
o 78.85%, showing us that RFM is a better predictor for the overall set of authors
fac = factor(DTM_test$test_labels)
acc$`Balanced Accuracy`[mapLevels(x=fac)['JaneMacartney'][[1]]]
```

```
## [1] 0.7885705
```

```
# This could be improved even more if we tune for parameters like max depth, or mtry(number of c
olumns) in the random forest iteration
```

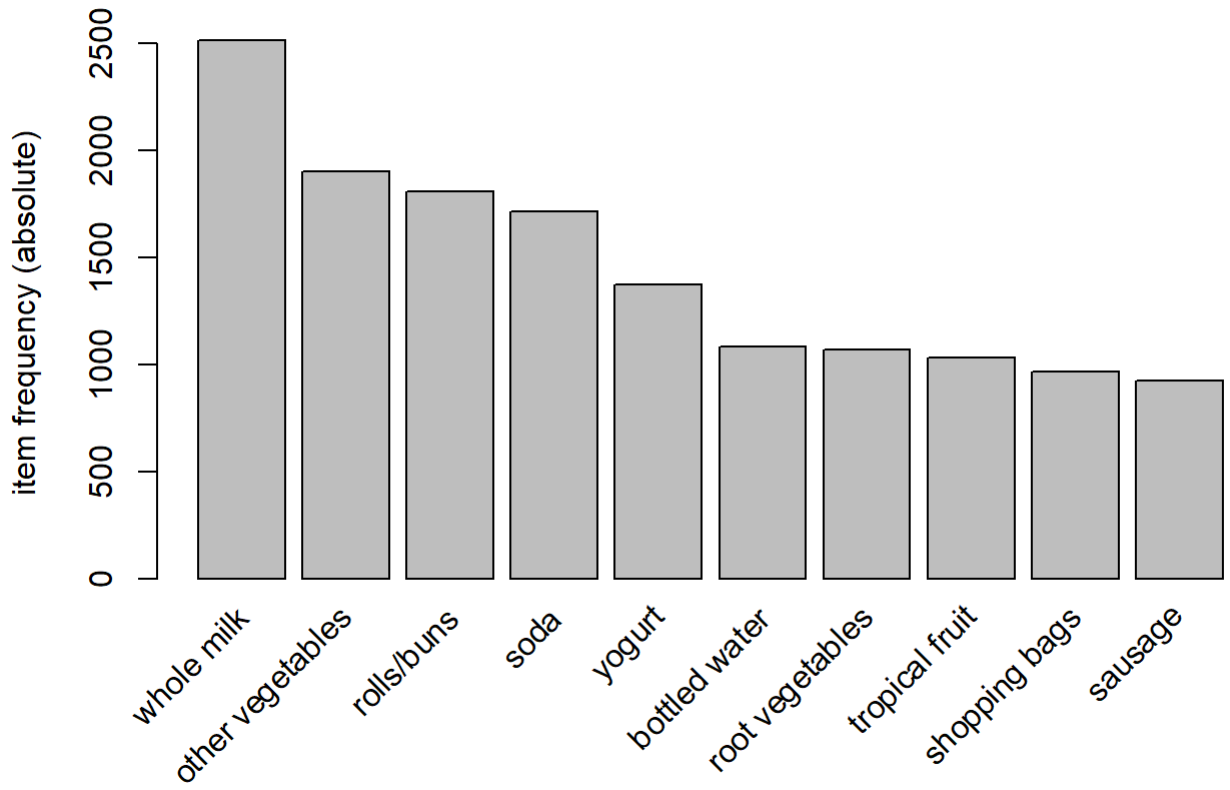
Association rule mining

Reading the grocery transaction level data and getting the summary

```
## 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
```

The most frequent items being bought are whole milk, other vegetables, rolls/buns, soda and yogurt

```
## transactions in sparse format with
##  9835 transactions (rows) and
##  169 items (columns)
```



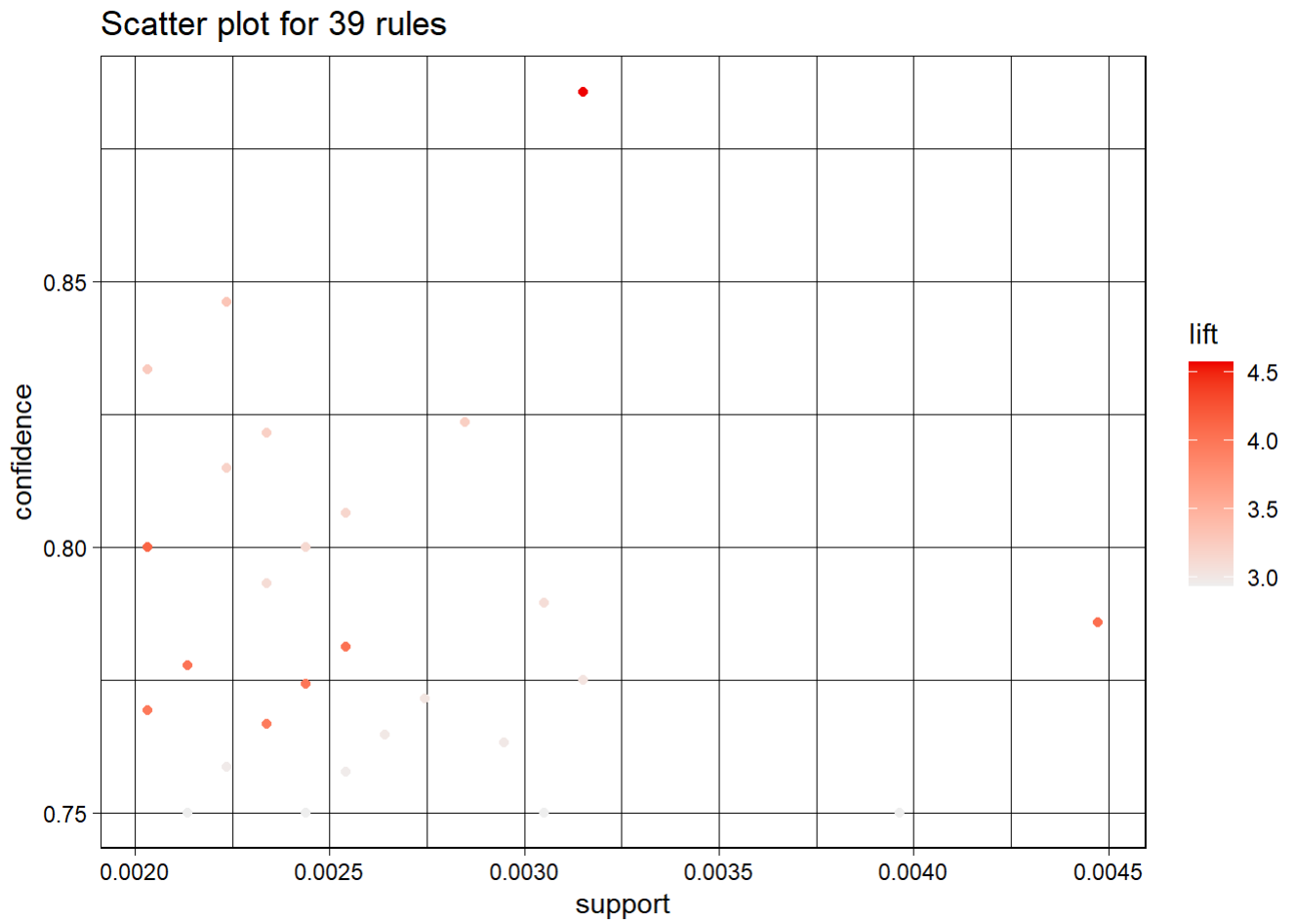
The top item purchased was whole milk with about 2,500 of the 9,836 transactions in the basket

```
## Apriori
##
## Parameter specification:
##  confidence minval smax arem  aval originalSupport maxtime support minlen
##        0.75    0.1    1 none FALSE            TRUE       5   0.002      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: 19
##
## set item appearances ...[0 item(s)] done [0.00s].
## set transactions ...[169 item(s), 9835 transaction(s)] done [0.00s].
## sorting and recoding items ... [147 item(s)] done [0.00s].
## creating transaction tree ... done [0.00s].
## checking subsets of size 1 2 3 4 5 done [0.00s].
## writing ... [39 rule(s)] done [0.00s].
## creating S4 object  ... done [0.00s].
```

```
##     lhs                                rhs          support     confidence
## [1] {rice, root vegetables}          => {whole milk} 0.002440264 0.7741935
## [2] {herbs, tropical fruit}          => {whole milk} 0.002338587 0.8214286
## [3] {herbs, rolls/buns}              => {whole milk} 0.002440264 0.8000000
## [4] {butter milk, whipped/sour cream} => {whole milk} 0.002948653 0.7631579
## [5] {butter, onions}                 => {whole milk} 0.003050330 0.7500000
##     coverage    lift     count
## [1] 0.003152008 3.029922 24
## [2] 0.002846975 3.214783 23
## [3] 0.003050330 3.130919 24
## [4] 0.003863752 2.986732 29
## [5] 0.004067107 2.935237 30
```

```
## set of 39 rules
##
## rule length distribution (lhs + rhs):sizes
##  3  4  5
##  6 24  9
##
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   3.000   4.000   4.000   4.077   4.000   5.000
##
## summary of quality measures:
##     support           confidence        coverage             lift
##  Min.   :0.002034   Min.   :0.7500   Min.   :0.002440   Min.   :2.935
##  1st Qu.:0.002237   1st Qu.:0.7667   1st Qu.:0.002847   1st Qu.:3.000
##  Median :0.002440   Median :0.7742   Median :0.003050   Median :3.104
##  Mean   :0.002539   Mean   :0.7850   Mean   :0.003241   Mean   :3.327
##  3rd Qu.:0.002694   3rd Qu.:0.8000   3rd Qu.:0.003457   3rd Qu.:3.637
##  Max.   :0.004474   Max.   :0.8857   Max.   :0.005694   Max.   :4.578
##      count
##  Min.   :20.00
##  1st Qu.:22.00
##  Median :24.00
##  Mean   :24.97
##  3rd Qu.:26.50
##  Max.   :44.00
##
## mining info:
##        data ntransactions support confidence
##  grocery_df          9835   0.002       0.75
##                                                                                call
##  apriori(data = grocery_df, parameter = list(support = 0.002, confidence = 0.75))
```

We need a very low support (of 0.002) and a relatively low confidence (of 0.75) for apriori to learn association rules. This iteration gives us 39 rules.



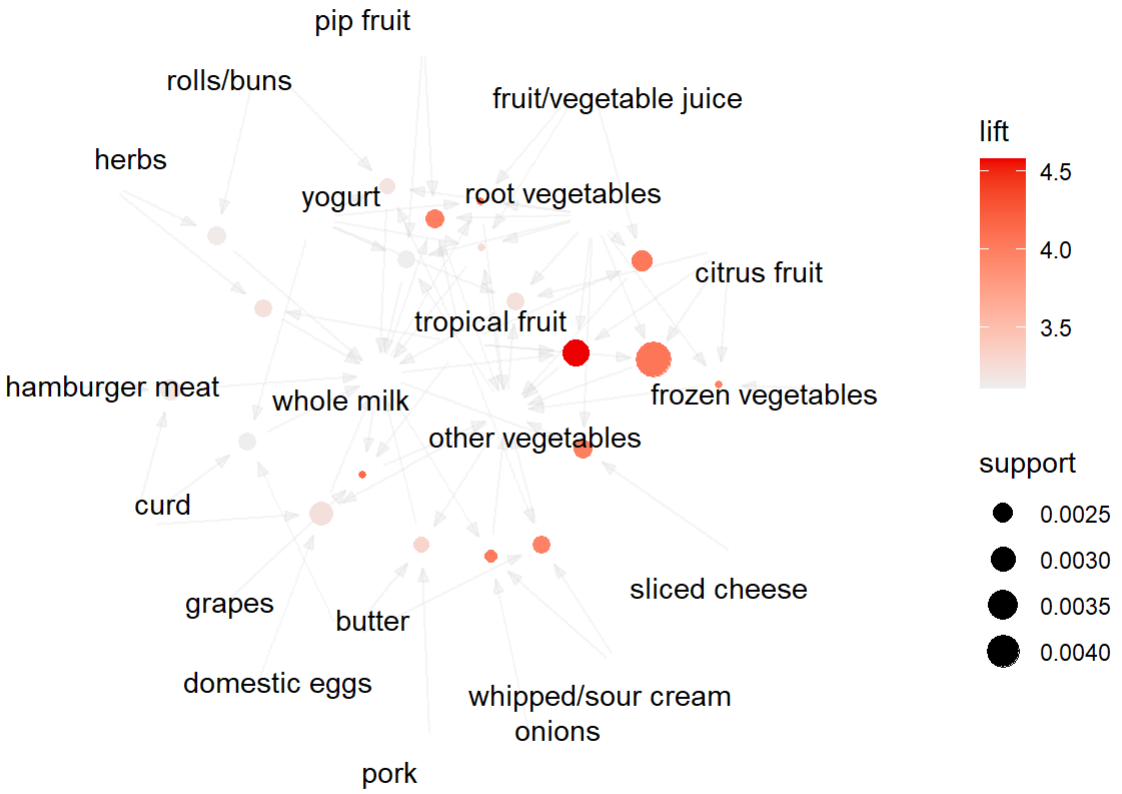
This plot shows high lift rules have lower support and confidence. Subsetting for lift greater than 4 gives use the following

```
##     lhs                         rhs                   support confidence    coverage     lif
t count
## [1] {grapes,
##      tropical fruit,
##      whole milk}             => {other vegetables} 0.002033554  0.8000000 0.002541942 4.13452
4    20
## [2] {root vegetables,
##      sliced cheese,
##      whole milk}             => {other vegetables} 0.002440264  0.7741935 0.003152008 4.00115
3    24
## [3] {onions,
##      whipped/sour cream,
##      whole milk}             => {other vegetables} 0.002135231  0.7777778 0.002745297 4.01967
7    21
## [4] {fruit/vegetable juice,
##      root vegetables,
##      tropical fruit}         => {other vegetables} 0.002541942  0.7812500 0.003253686 4.03762
2    25
## [5] {citrus fruit,
##      root vegetables,
##      tropical fruit}         => {other vegetables} 0.004473818  0.7857143 0.005693950 4.06069
4    44
## [6] {fruit/vegetable juice,
##      root vegetables,
##      whole milk,
##      yogurt}                 => {other vegetables} 0.002033554  0.8000000 0.002541942 4.13452
4    20
## [7] {pip fruit,
##      root vegetables,
##      tropical fruit,
##      whole milk}             => {other vegetables} 0.002440264  0.7741935 0.003152008 4.00115
3    24
## [8] {citrus fruit,
##      root vegetables,
##      tropical fruit,
##      whole milk}             => {other vegetables} 0.003152008  0.8857143 0.003558719 4.57750
9    31
```

This shows that people generally tend to buy milk, fruits yogurt and vegetables together.

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

```
## 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 plot reinforces the statement we make earlier that people tend to buy veggies, fruits and dairy products together.

We can assume the following based on the association rule model:

1. Whole Milk, yogurt and fruit and vegetable juices occur with high confidence indicating that people are probably buying these things together to make nutritious smoothies.
2. root vegetables and fruits occur with other vegetables indicating that people tend to make conscious nutritious choices.
3. citrus fruits and tropical fruits occur together, suggesting that people are preferring fruits that are native to hot and humid climates (tropical countries).
4. whole milk is associated with other vegetables majority of time with lift>4 indicating that there is 4 times a chance of people buying other vegetables when they buy whole milk.