EE7206: MACHINE LEARNING

PROJECT

NAME: THENNAKOON T.M.B.S.B

REG NO: EG/2015/2781

DATE: 22/09/2019

Throughout the project stock “ES0158252033” was selected for applying as it has highest number of rows in the dataset.

Q1 – Identifying Trade Outliers and Trader Outliers

The R file related to this section is trade_outliers_k_means.R.

Clustering algorithms can be used to identify the outliers in the given the dataset. Clustering algorithms such as K-Means Clustering, Hierarchical Clustering, Isolation forests and DB Scan Algorithms.

Identifying the Trade Outliers

Using K Means Clustering

First the Within Cluster Sum of Squares (WCSS) was calculated for Executed Price of the Selected dataset and the Elbow method was used to determine the optimum number of clusters.

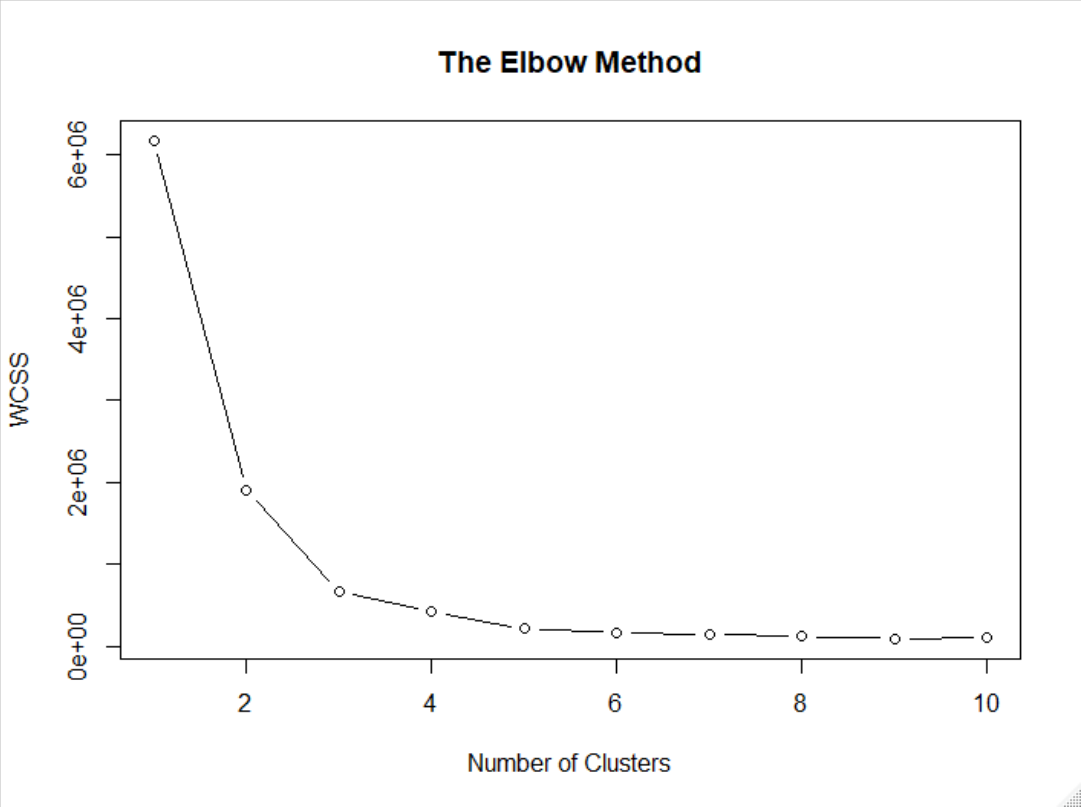

Figure 1: Number of Clusters vs WCSS for Trade Quantities

It is clear from the above figure the optimum of clusters to the which the above dataset can be broken into is 3 as the elbow of the graph is at 3.

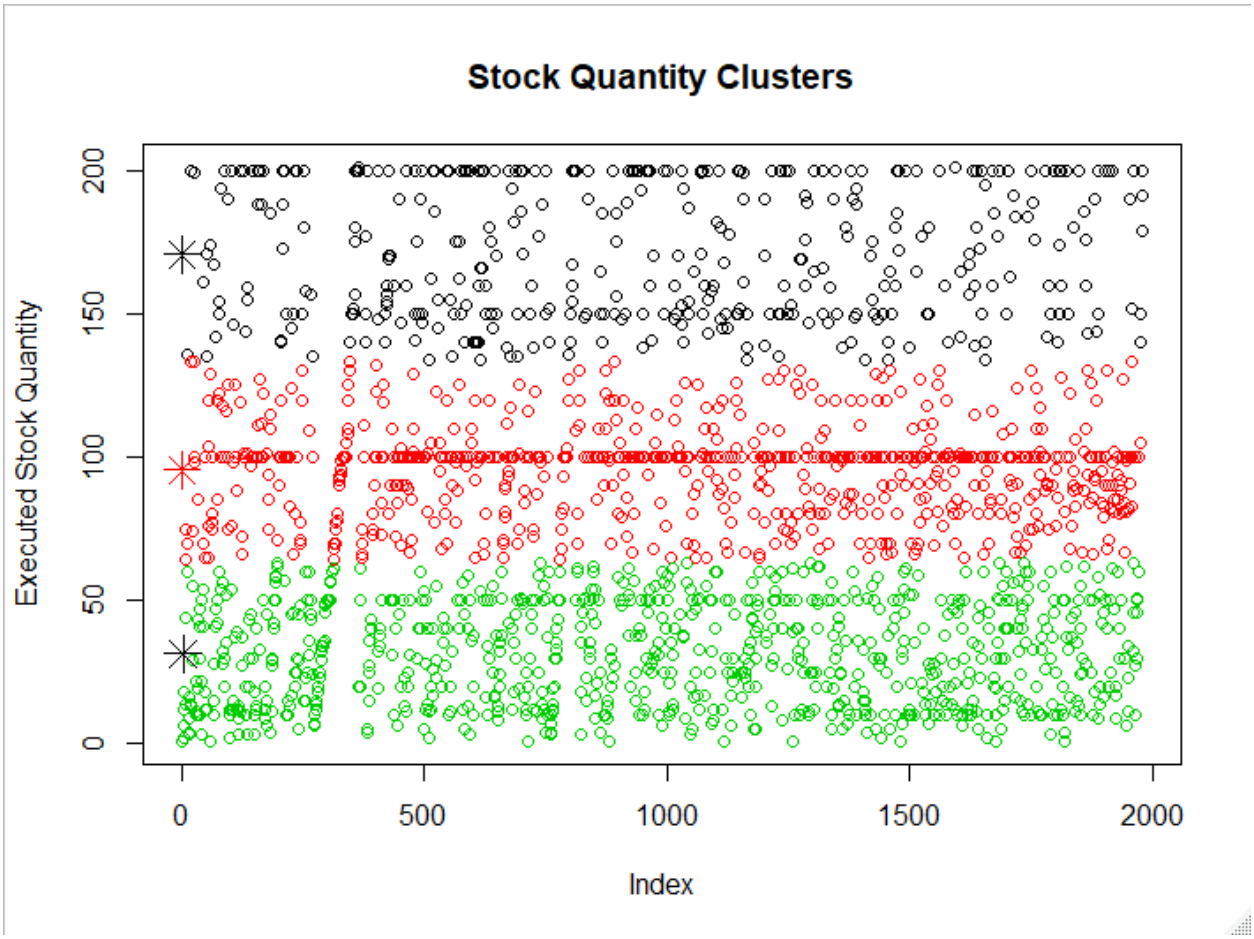

Figure 2: Results from K means clustering

‘*’ shows the cluster centers

```
> kmeans$centers[, 1]
       1        2        3
170.65848  95.44552  31.33608
> |
```

Figure 3: Cluster Centroids

By assuming this as a normal distribution we can separate the outlier cluster using mean and the standards deviations. The clusters whose centroids are beyond the value, mean + 3* standard deviation can be considered as an outlier cluster.

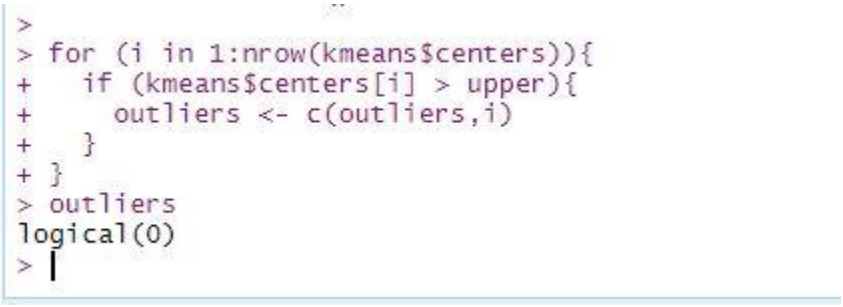

Figure 4: Code for finding outlier centroids

This method detects no outliers for this dataset.

Trader Outliers

Using K Means Clustering

The R file related to this section is buyer_outliers_k_means.R.

First the Within Cluster Sum of Squares (WCSS) was calculated for Executed Price of the Selected dataset and the Elbow method was used to determine the optimum number of clusters.

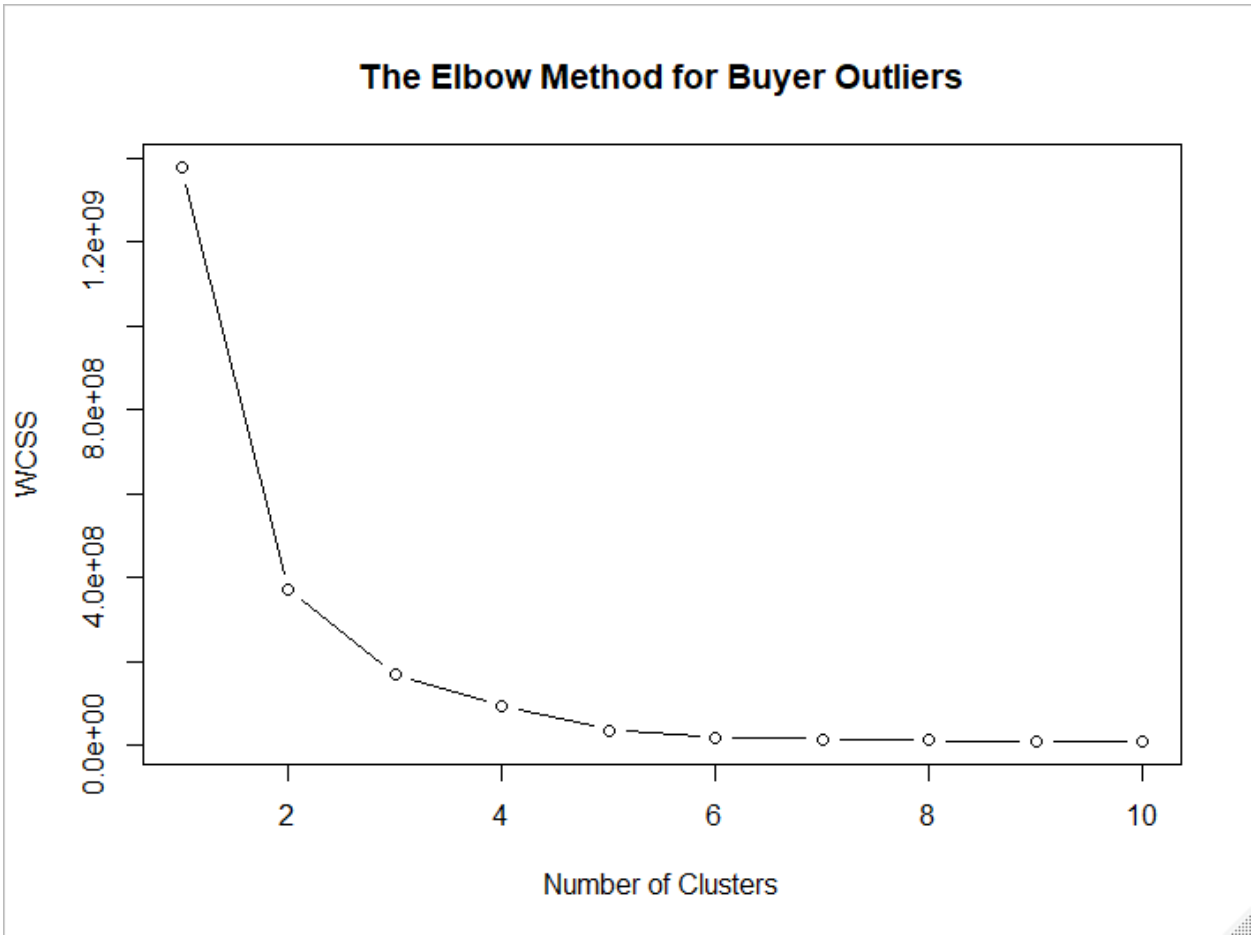

Figure 5: Number of Clusters vs WCSS Buyer Outliers

It is clear from the above figure the optimum of clusters to the which the above dataset can be broken into is 4 as the elbow of the graph is at 4.

Figure 6: Results from K Means Clustering

```
# plot cluster centers
points(kmeans$centers[, 1], col = 1:2, pch = 8, cex = 2)
```

Figure 7: Cluster Centroids

By assuming this as a normal distribution we can separate the outlier cluster using mean and the standards deviations. The clusters whose centroids are beyond the value, mean + 3* standard deviation can be considered as an outlier cluster

```
>
> #Separating outliers
> outliers_set <- data.frame()
> outliers_values <- vector()
> for (i in 1:2000) {
+   if(df_trades$Executed.Qty[i] >upper) {
+     outliers_values<- c(outliers,df_trades$Executed.Qty[i] )
+     filtered_trades =subset(df_trades, df_trades$Executed.Qty == df_trades$Executed.Qty[i])
+     outliers_set <- rbind(outliers_set,filtered_trades)
+
+   }
+ }
> outliers
[1] 1
>
```

Figure 8: Code detecting outliers

The code detected cluster one as the outliers which can be clearly observed from the cluster plot as well.

Q2 – Identifying Collusive Trader Groups

The code related to this section is collusive_trader.R

First the variation of price of the stock 'ES0158252033' was observed.

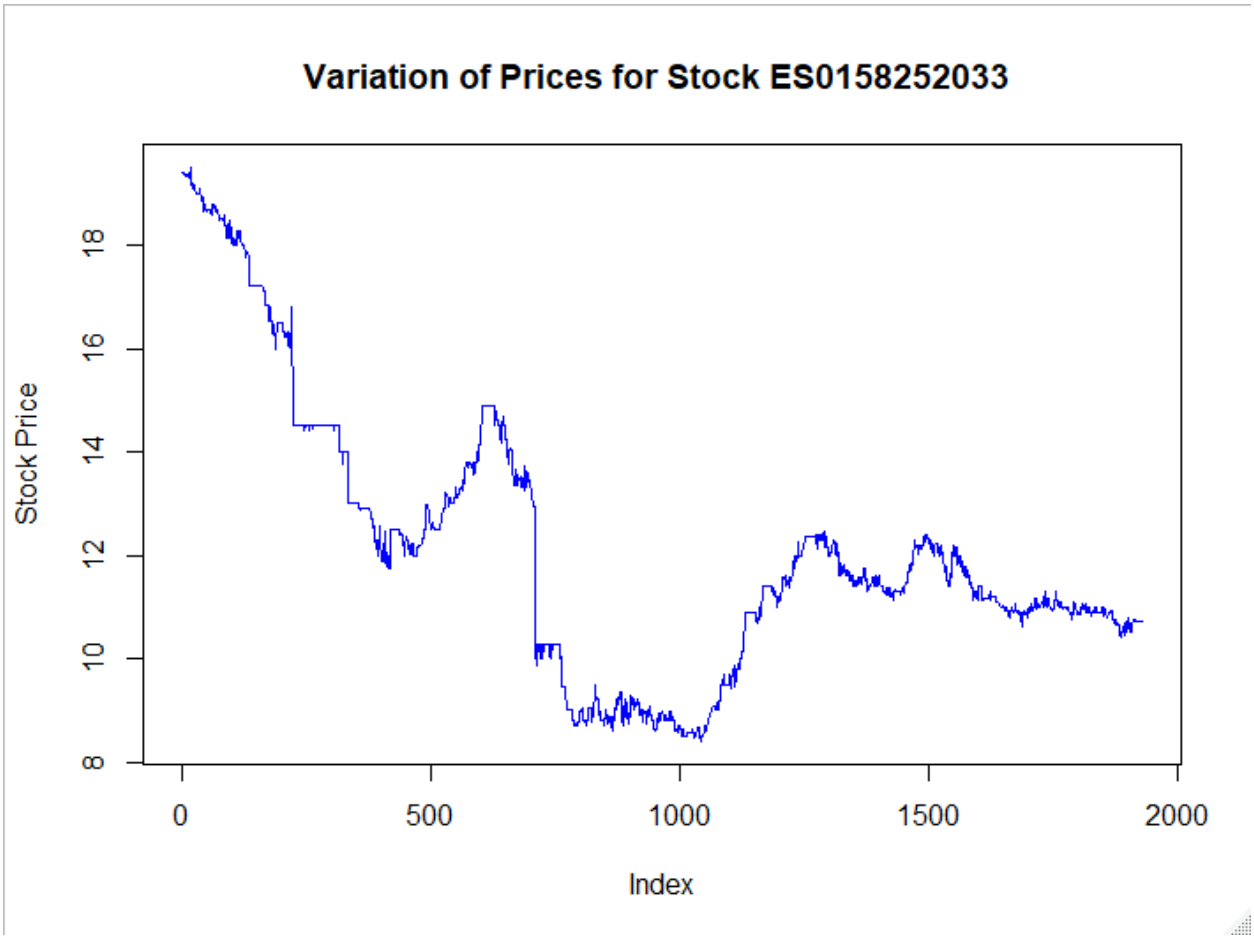

Figure 9: Variation of the stock price of the Selected Stock over time

Then a trailing moving average for the executed price was calculated for identifying sudden rises and drops in stock prices. The trailing average was selected instead of a centered moving average as price change should be compared with the historic values.

The 50 points trailing moving average was selected because it strikes a balance between identifying noise points while maintaining the data structure.

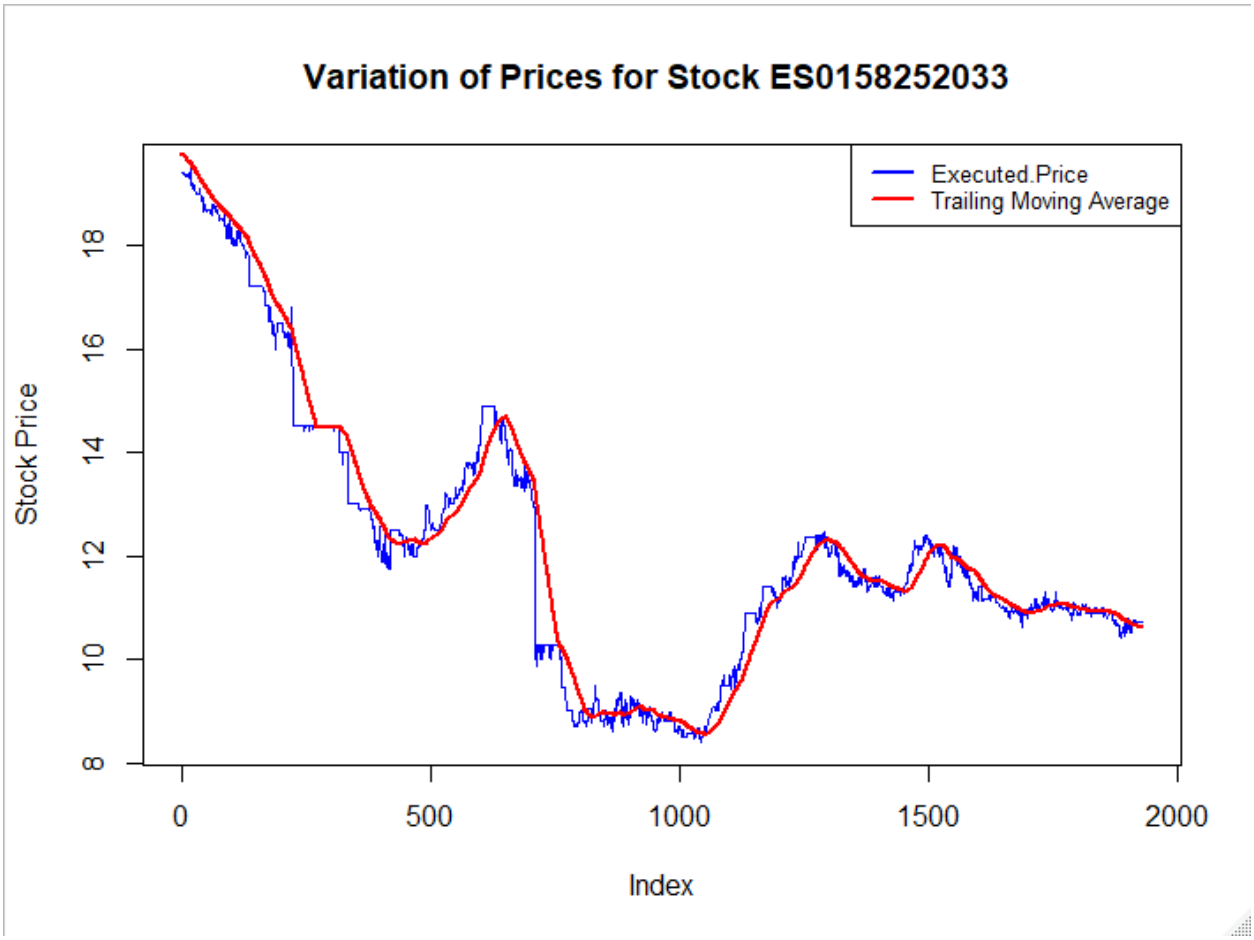

Figure 10: Trailing Moving Average Curve

Then percentage change of price with trailing moving average was calculated with the intention of identify sudden rises and drops in price. The points where changes are greater than 2% was selected condition for selecting sudden price changes through trial and error.

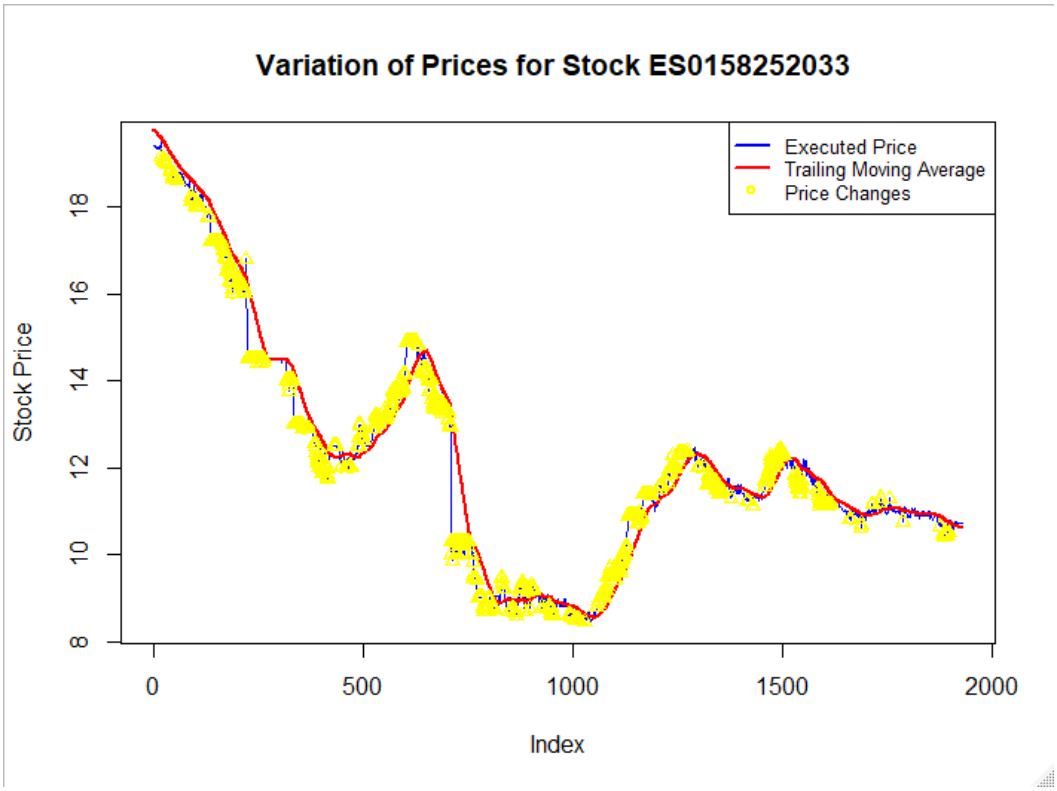

Figure 11: Detected Price Changes using Trailing Moving Averages

Applying Apriori Algorithm to find Collusive Traders.

First a sparse matrix of Buyers and sellers of the suspected transactions

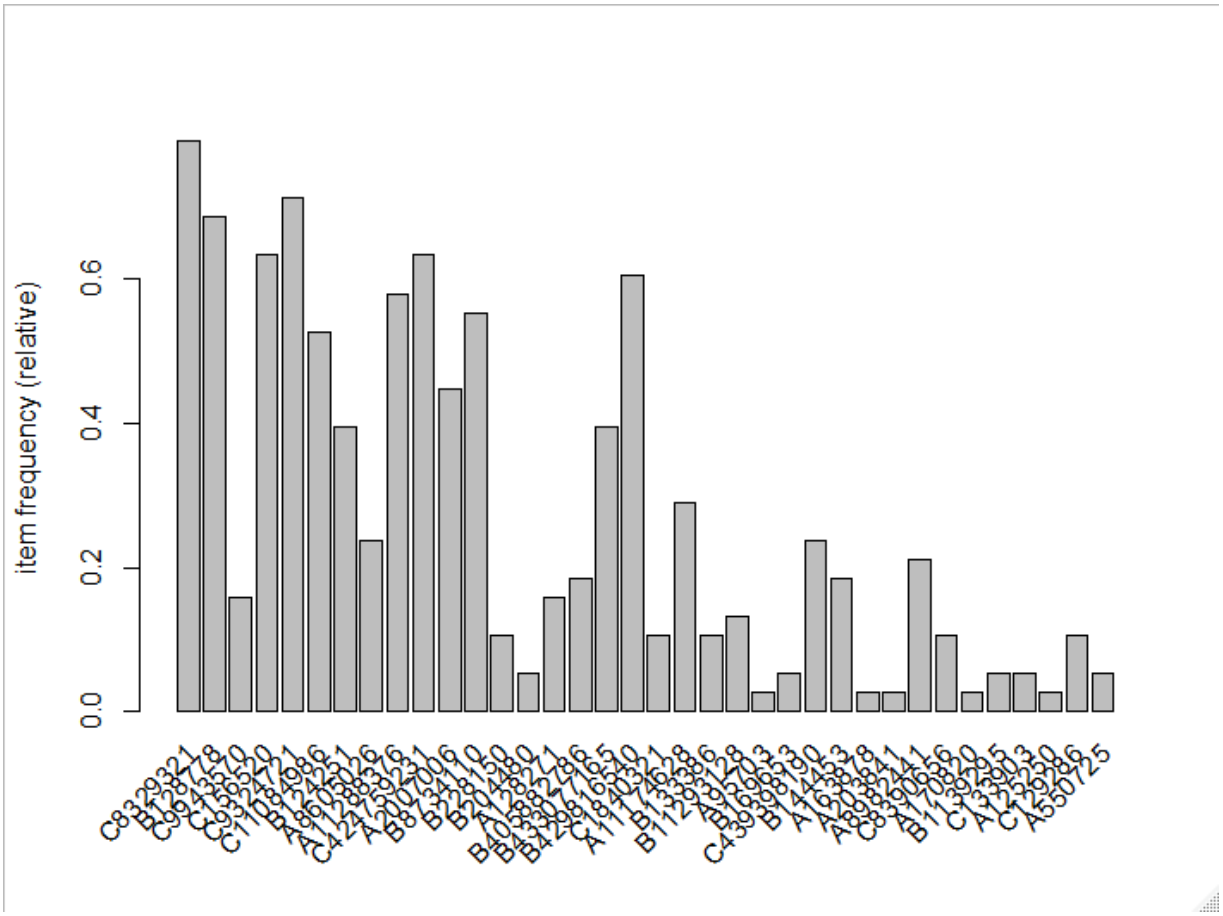

Figure 12: Item Frequency Plot

Apriori algorithm applied using arules library. The minimum support was set to 0.5 and the minimum confidence was set to 0.8. These values were selected by trial and error.

```
> inspect(sort(rules,by ='lift')[1:10])
     lhs                          rhs          support   confidence lift     count
[1]  {B128778,C424759231}      => {B8734110}   0.5000000 0.9500000  1.719048 19
[2]  {C8329321,B429816540}     => {C156520}    0.5263158 1.0000000  1.583333 20
[3]  {C8329321,B128778}        => {B8734110}   0.5263158 0.8695652  1.573499 20
[4]  {B128778,C9324721}        => {B8734110}   0.5000000 0.8636364  1.562771 19
[5]  {C8329321,C156520}        => {B429816540} 0.5263158 0.9090909  1.501976 20
[6]  {B128778}                 => {B8734110}   0.5526316 0.8076923  1.461538 21
[7]  {B8734110}                => {B128778}    0.5526316 1.0000000  1.461538 21
[8]  {C424759231,B8734110}     => {B128778}    0.5000000 1.0000000  1.461538 19
[9]  {C9324721,B8734110}       => {B128778}    0.5000000 1.0000000  1.461538 19
[10] {C8329321,B8734110}       => {B128778}    0.5263158 1.0000000  1.461538 20
> |
```

Figure 13: Collusive Traders arranged according to the highest lift