Customer Segmentation using RFM - Clustering using K-means

2020-01-29

Introduction:

One of the most important tasks for any business is to know their customers. In today’s world every business needs to offer personalized products and services to its customers or risk losing them.

Customers are both similar and different. It is impossible to have individualized products and services for each customer. Hence the need to segment customers with similar characteristics and have tailored offerings to each group.

There are many characteristics on which customers can be segmented. Common characteristics used are customer behaviour, demography and interests.

Data like customer purchase date and value are readily available with vendors. It therefore makes sense to use them for targeted marketing. Recency of purchase, Frequency of purchases and Monetary value of purchases - popularly referred to as RFM (Receny-Frequency-Monetary) are one of the most effective methods used for customer segmentation.

Clustering algorithms do not necessarily know the outcomes and are interested in discovering groups. K-means is one of the most popular ways to segment customers using unsupervised clustering techniques.

Given a set of observations (x1, x2, …, xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition n observations into k (≤ n) sets S = {S1, S2, …, Sk} so as to minimize the within-cluster sum of squares… more about k-means

We will use k-means for customer segmentation.

Data:

We will use Superstore Orders data for this analysis.

Load dependent libraries

```r
if(!require(tidyverse)) install.packages("tidyverse", repos = "http://cran.us.r-project.org")
if(!require(caret)) install.packages("caret", repos = "http://cran.us.r-project.org")
if(!require(gridExtra)) install.packages("gridExtra", repos = "http://cran.us.r-project.org")
```

Load data from CSV file

```r
# Load store orders data from csv file
orders <-
  read_csv("https://raw.githubusercontent.com/madankundapur/DataAnalytics/master/Data/SuperstoreOrders.csv")
```

The dataset has 9994 observations with 21 variables and contains store orders data for the United States.

```r
str(orders)
```

```
## Classes 'spec_tbl_df', 'tbl_df', 'tbl' and 'data.frame': 9994 obs. of  21 variables:
##  $ Row ID       : num  1 2 3 4 5 6 7 8 9 10 ...
##  $ Order ID     : chr  "CA-2017-152156" "CA-2017-152156" "CA-2017-138688" "US-2016-108966" ...
##  $ Order Date   : chr  "11/8/2017" "11/8/2017" "6/12/2017" "10/11/2016" ...
##  $ Ship Date    : chr  "11/11/2017" "11/11/2017" "6/16/2017" "10/18/2016" ...
##  $ Ship Mode    : chr  "Second Class" "Second Class" "Second Class" "Standard Class" ...
##  $ Customer ID  : chr  "CG-12520" "CG-12520" "DV-13045" "SO-20335" ...
##  $ Customer Name: chr  "Claire Gute" "Claire Gute" "Darrin Van Huff" "Sean O'Donnell" ...
##  $ Segment      : chr  "Consumer" "Consumer" "Corporate" "Consumer" ...
##  $ Country      : chr  "United States" "United States" "United States" "United States" ...
##  $ City         : chr  "Henderson" "Henderson" "Los Angeles" "Fort Lauderdale" ...
##  $ State        : chr  "Kentucky" "Kentucky" "California" "Florida" ...
##  $ Postal Code  : num  42420 42420 90036 33311 33311 ...
##  $ Region       : chr  "South" "South" "West" "South" ...
##  $ Product ID   : chr  "FUR-BO-10001798" "FUR-CH-10000454" "OFF-LA-10000240" "FUR-TA-10000577" ...
##  $ Category     : chr  "Furniture" "Furniture" "Office Supplies" "Furniture" ...
##  $ Sub-Category : chr  "Bookcases" "Chairs" "Labels" "Tables" ...
##  $ Product Name : chr  "Bush Somerset Collection Bookcase" "Hon Deluxe Fabric Upholstered Stacking
Chairs, Rounded Back" "Self-Adhesive Address Labels for Typewriters by Universal" "Bretford CR4500
Series Slim Rectangular Table" ...
##  $ Sales        : num  262 731.9 14.6 957.6 22.4 ...
##  $ Quantity     : num  2 3 2 5 2 7 4 6 3 5 ...
##  $ Discount     : num  0 0 0 0.45 0.2 0 0 0.2 0.2 0 ...
##  $ Profit       : num  41.91 219.58 6.87 -383.03 2.52 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   `Row ID` = col_double(),
##   ..   `Order ID` = col_character(),
##   ..   `Order Date` = col_character(),
##   ..   `Ship Date` = col_character(),
##   ..   `Ship Mode` = col_character(),
##   ..   `Customer ID` = col_character(),
##   ..   `Customer Name` = col_character(),
##   ..   Segment = col_character(),
##   ..   Country = col_character(),
##   ..   City = col_character(),
##   ..   State = col_character(),
##   ..   `Postal Code` = col_double(),
##   ..   Region = col_character(),
##   ..   `Product ID` = col_character(),
##   ..   Category = col_character(),
##   ..   `Sub-Category` = col_character(),
##   ..   `Product Name` = col_character(),
##   ..   Sales = col_double(),
##   ..   Quantity = col_double(),
##   ..   Discount = col_double(),
##   ..   Profit = col_double()
##   .. )
```

Data variable names have spaces in them and as a practise it is good to avoid spaces. Also note that the variable names are in proper casing - we will retain and follow that convention for naming data variables.

```
names(orders)<-str_replace_all(names(orders), c(" " = ""))
```

OrderDate variable is of type 'character'. Changing it to 'date'

```
orders$OrderDate <- as.Date(orders$OrderDate, "%m/%d/%Y")
class(orders$OrderDate)
```

```
## [1] "Date"
```

To keep data tidy check for duplicates and filter them out.

```
duplicates <- which(duplicated(orders))
duplicates
```

```
## integer(0)
```

```
# No duplicates exist in data
rm(duplicates)
```

Data that we need for RFM analysis is *OrderDate* and *Sales* amount by customer. The dataset has order details at the product level which we don't need. Let us aggregate *Sales* amount and select only necessary variables for further analysis

```
orders <- orders %>%
  group_by(CustomerID, OrderID , OrderDate) %>%
  summarize(Sales = sum(Sales)) %>%
  select(CustomerID, OrderID , OrderDate, Sales)

# Checking if the orders are equal to the observations in the dataset
length(unique(orders$OrderID ))
```

```
## [1] 5009
```

```
nrow(orders)
```

```
## [1] 5009
```

Order dates range from Jan 2015 through Dec 2018. Compute maximum date from the dataset. This will help compute *DaysSincePurchase* and *Recency*. Note that a day is added to the maximum date to ensure that there are no zeroes calculated (applies to purchases made on the last day).

```
range(orders$OrderDate)
```

```
## [1] "2015-01-03" "2018-12-30"
```

```
max_date <- max(orders$OrderDate)+1
```

Compute *PurchaseYear* - which is year part of order date and *DaysSincePurchase* - which is the difference between order date and the maximum date in the dataset

```
orders <- orders %>%
  mutate(PurchaseYear = as.numeric(format(OrderDate, "%Y")),
         DaysSincePurchase = as.numeric(difftime(max_date, OrderDate,"days")))

rm(max_date)
```

Compute Recency, Frequency and Monetary Value:

For each customer compute RFM values:

- *Recency* is the duration in days since the last purchase made by the customer
- *Frequency* is the number of distinct orders by customer
- *Monetary* value is total sales amount for the customer

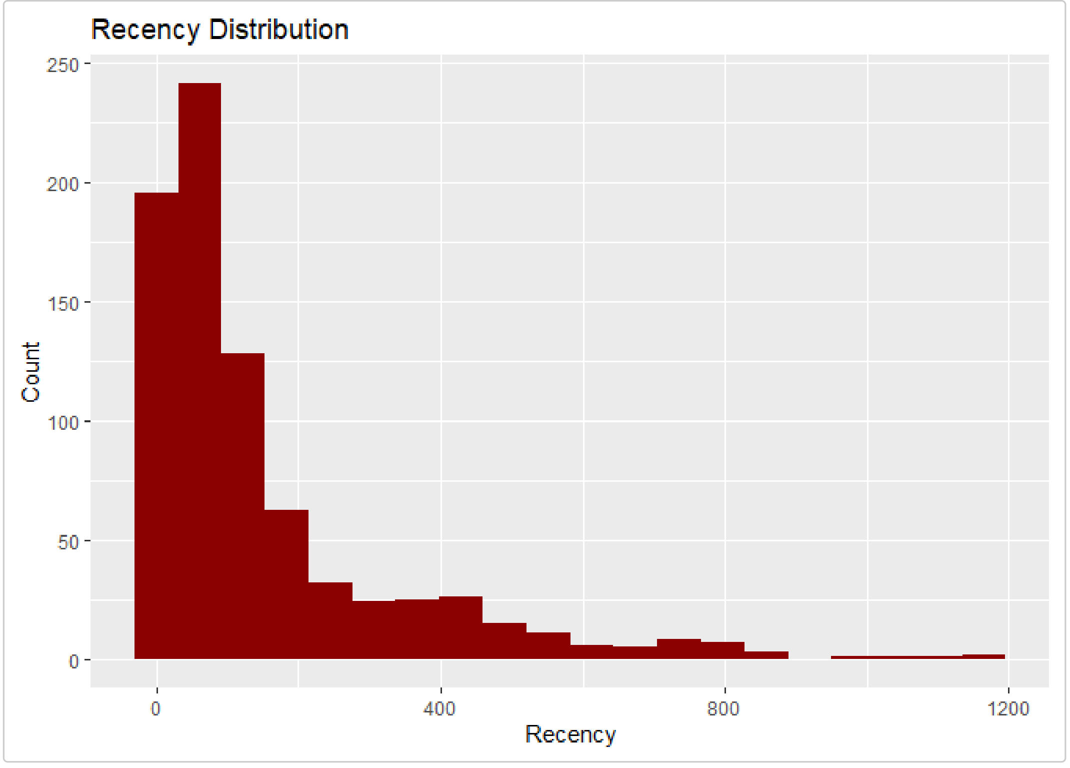
```
customers <- orders %>%
  group_by(CustomerID) %>%
  summarise(Recency = min(DaysSincePurchase),
            Frequency = n_distinct(OrderID),
            Monetary = sum(Sales))

knitr::kable(summary(customers))
```

CustomerID	Recency	Frequency	Monetary
Length:793	Min. : 1.0	Min. : 1.000	Min. : 4.833
Class :character	1st Qu.: 31.0	1st Qu.: 5.000	1st Qu.: 1146.050
Mode :character	Median : 76.0	Median : 6.000	Median : 2256.394
NA	Mean : 147.8	Mean : 6.317	Mean : 2896.848
NA	3rd Qu.: 184.0	3rd Qu.: 8.000	3rd Qu.: 3785.276
NA	Max. :1166.0	Max. :17.000	Max. :25043.050

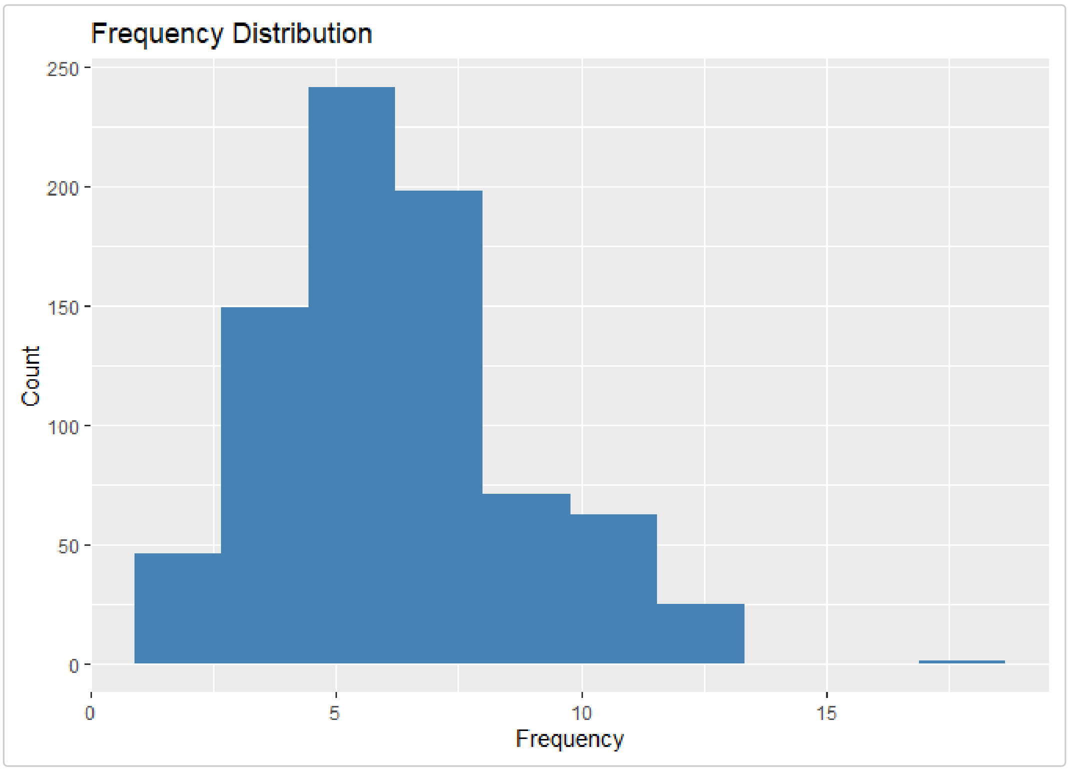
Plot distribution for Recency, Frequency and Monetary Value to explore RFM data

```
customers %>% ggplot(aes(Recency))  +
  geom_histogram(bins=20,fill = "darkred") +
  labs(x = "Recency", y = "Count", title = "Recency Distribution")
```



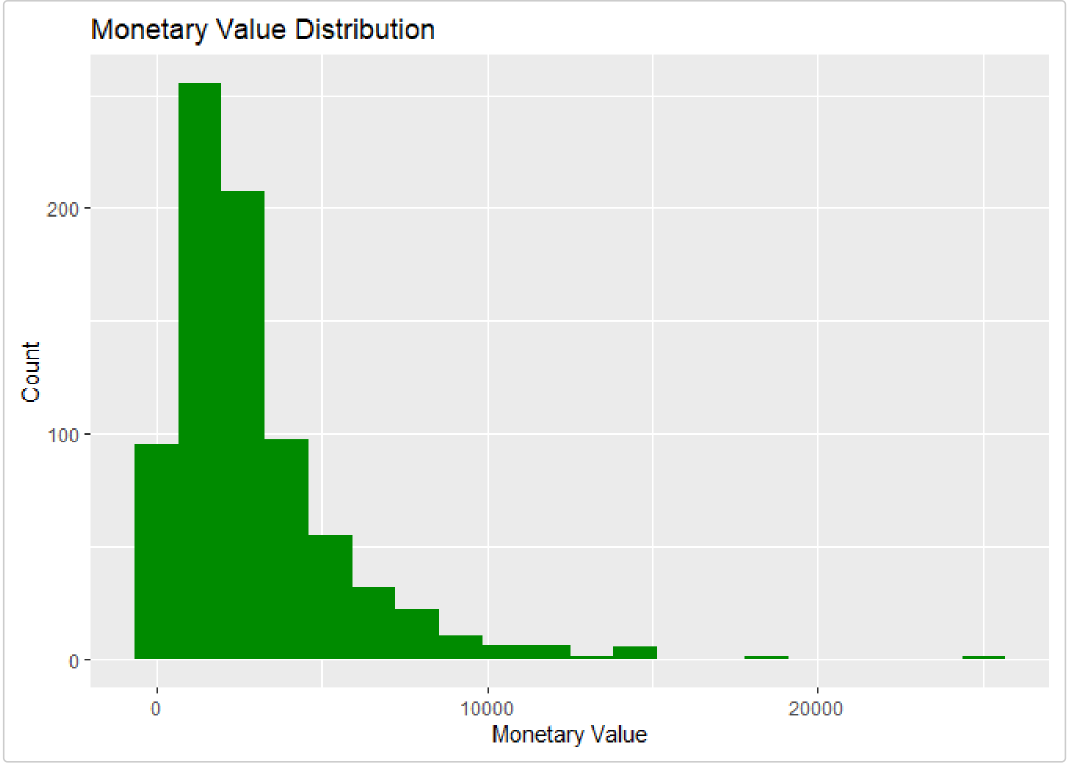
From the Recency plot, more than 80% of customers have been active in the last one year, which is a good sign.

```
customers %>% ggplot(aes(Frequency))  +
  geom_histogram(bins=10,fill = "steelblue")+
  labs(x = "Frequency", y = "Count", title = "Frequency Distribution")
```



From the Frequency plot, the values are more-or-less distributed and the range is between 1 and 13 with an outlier of 17.

```
customers %>% ggplot(aes(Monetary))  +
  geom_histogram(bins=20,fill = "green4") +
  labs(x = "Monetary Value", y = "Count", title = "Monetary Value Distribution")
```



From the Monetary value plot, more than 97% of customers have spent less than $10000 across years.

Since the scale of values are very different for Recency, Frequency and Monetary. Let us remove the skew and standardise the RFM values.

```
customers$RecencyZ <- scale(log(customers$Recency), center=TRUE, scale=TRUE)
customers$FrequencyZ <- scale(log(customers$Frequency), center=TRUE, scale=TRUE)
customers$MonetaryZ <- scale(log(customers$Monetary), center=TRUE, scale=TRUE)
```

We now have a tidy dataset with 793 observations of 8 variables to work with.

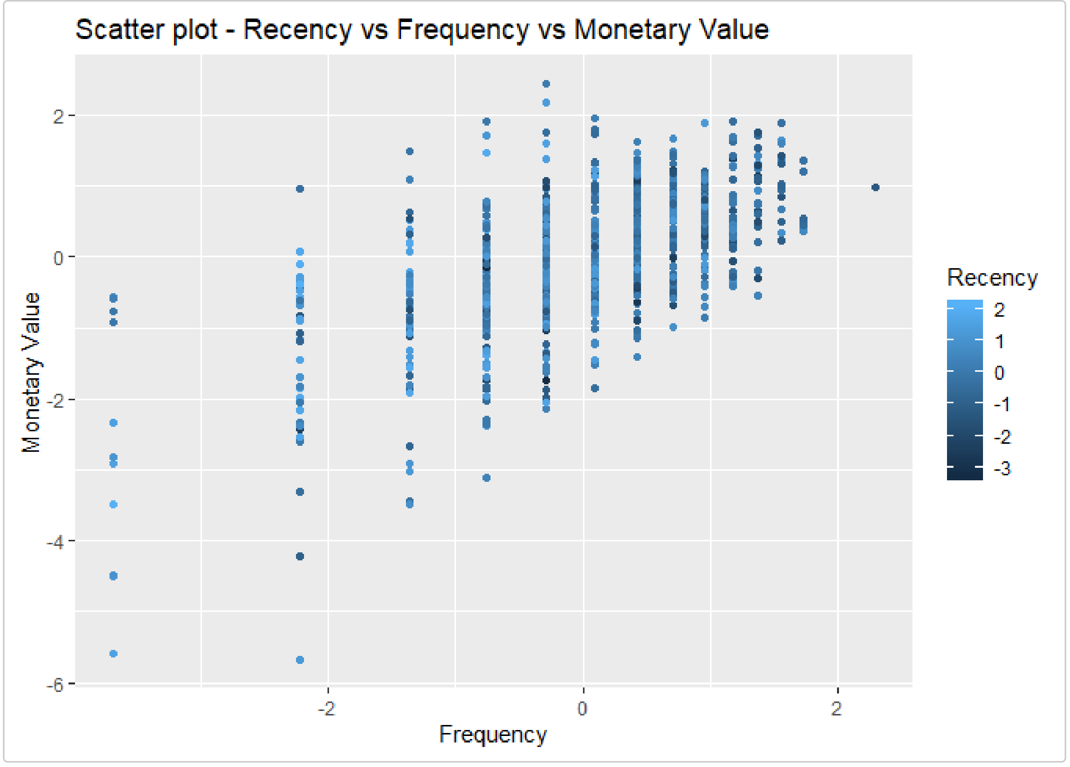
Analysis:

As mentioned earlier, for the purpose of this analysis we will use k-means clustering algorithm.

To use k-means clustering algorithm we have to pre-define k, which is the number of clusters we want to define. The k-means algorithm is iterative. The first step is to define k centers. Then each observation is assigned to the cluster with the closest center to that observation. In the second step the centers are redefined using the observation in each cluster. The column means are used to define a centroid. We repeat these steps until the centers converge.

Let us first see how the scatter plot looks for RFM values in the final dataset

```
customers %>% ggplot(aes(x=FrequencyZ,y=MonetaryZ)) +
  geom_point(aes(colour = RecencyZ)) +
  scale_colour_gradient(name="Recency") +
  labs(x = "Frequency", y = "Monetary Value", title = "Scatter plot - Recency vs Frequency vs Monetary
 Value")
```



K-means clustering

Note that the plot shows high-frequency and high monetary value customers in the top right with recency indicated in dark shades of blue. Similarly, low-frequency, low monetary value customers are in the bottom-left with recency indicated in the lighter shades of blue.

Looking at the entire plot it is hard find clusters. But the data points also don't seem to be distributed continuously. We therefore need to assume the clusters to extract and see which is the best fit. Let us assume a maximum value of 10 for cluster centers and loop through to arrive at the best cluster.

We will print the medians of RFM grouped by the cluster levels to analyze and find the best cluster.

Each cluster also provides us with the following information.

- *totss* is the total sum of squares
- *withinss* is the vector of within-cluster sum of squares, one component per cluster
- *tot.withinss* is the total within-cluster sum of squares, i.e. sum(withinss)
- *betweenss* is the between-cluster sum of squares, i.e. totss-tot.withinss

Also, let us temporarily persist *tot.withinss* and plot it to identify the best cluster.

```
j<- 10

# data frame to hold cluster components
ss <- data.frame(K=integer(),
                 TWSS=numeric())

# ensure customers dataset is a data frame
customers <- as.data.frame(customers)

# loop to create upto 10 clusters
for (i in 1:j ) {

  set.seed(1, sample.kind="Rounding")

  # Run k-means with i centers, assume nstart =25
  km <- kmeans(customers[,c(5:7)], centers = i, nstart = 25)

  # Adding cluster data  to customers dataset for each i in different variables
  col_nm <- paste("C", i, sep="")
  customers[,(col_nm)] <- factor(km$cluster, levels = c(1:i))

  # Find medians for RFM grouped by cluster and print them
  med <- customers %>%
      group_by(Cluster = customers[,(col_nm)]) %>%
      summarize(Recency=round(median(Recency), 0),
              Frequency=round(median(Frequency),1),
              Monetary=round(median(Monetary),2))
  print(paste(i, "Clusters", sep=" "))
  print(med)
  # store cluster info
  ss[i,("K")] <- i
  ss[i,("TWSS")] <- km$tot.withinss
}
```

```
## [1] "1 Clusters"
## # A tibble: 1 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            76         6    2256.
## [1] "2 Clusters"
## # A tibble: 2 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            49         7    3087.
## 2 2           208         4     935.
## [1] "3 Clusters"
## # A tibble: 3 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            27         8    3101.
## 2 2           120         6    2459.
## 3 3           204         3     574.
## [1] "4 Clusters"
## # A tibble: 4 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            65         8    4047.
## 2 2            16         7    2217
## 3 3           160         5    1553.
## 4 4           284         2     348.
## [1] "5 Clusters"
## # A tibble: 5 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           296         2     219.
## 2 2           303         5    1418.
## 3 3            44         5     904.
## 4 4            98         7    3492.
## 5 5            22         8    3101.
## [1] "6 Clusters"
## # A tibble: 6 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            56         5     913.
## 2 2            38         8    3336.
## 3 3           153         7    2965.
## 4 4             7         7    2550.
## 5 5           407         4    1057.
## 6 6           227         2     131.
## [1] "7 Clusters"
## # A tibble: 7 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           246         2     84.0
## 2 2            42         4    862.
## 3 3           175         5   1791.
## 4 4           453         3    886.
## 5 5            34         8   2929.
## 6 6           108         8   5013.
## 7 7             7         7   2562.
## [1] "8 Clusters"
## # A tibble: 8 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1            35         9   3013.
## 2 2           265         2     79.8
## 3 3            40         4    741.
## 4 4           136         8   4873.
## 5 5           203         5   1523.
## 6 6             7         7   2528.
## 7 7           445         3    893.
## 8 8            51         5   3079.
## [1] "9 Clusters"
## # A tibble: 9 x 4
##   Cluster Recency Frequency Monetary
##   <fct>     <dbl>     <dbl>    <dbl>
## 1 1           378         5   2558.
## 2 2            126         6   1082.
## 3 3             99         8   4642.
## 4 4             31         4    742.
## 5 5            430         3    863.
## 6 6             55         5   2932.
## 7 7              6         7   2493.
## 8 8             29         8   3037.
## 9 9            265         2     79.8
## [1] "10 Clusters"
## # A tibble: 10 x 4
##    Cluster Recency Frequency Monetary
##    <fct>     <dbl>     <dbl>    <dbl>
##  1 1            31         4    740.
##  2 2           419         3    917.
##  3 3            30         9   4493.
##  4 4           192         5    839.
##  5 5            59         5   3021.
##  6 6           346         5   2367.
##  7 7           265         2     79.8
##  8 8             7         7   2493.
##  9 9           143         8   4511.
## 10 10           56         7   1860.
```

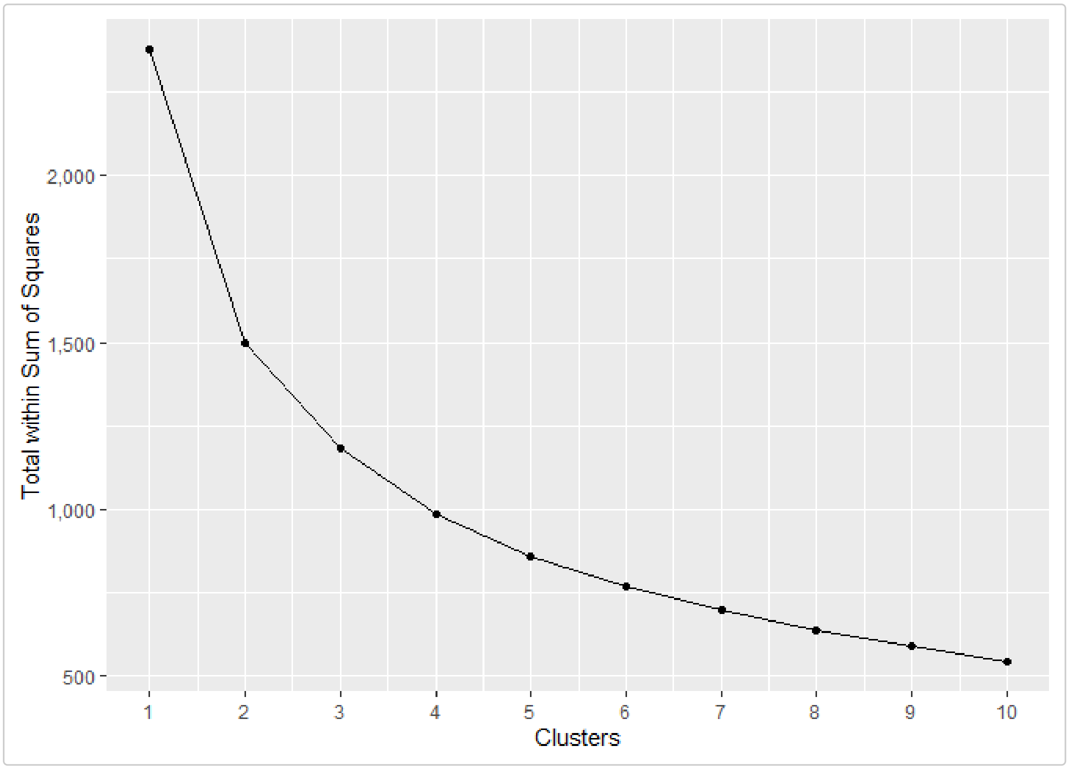
In the 2-cluster resultset, we find that high-recency, high-frequency and high-value customers are in one cluster. Low-recency, low-frequency and low-value customers are in the other cluster.

In the 3-cluster resultset, we find that high-recency, high-frequency and high-value customers are in cluster 1. Medium-recency, medium-frequency and medium-value customers are in cluster 2. Low-recency, low-frequency and low-value customers are in cluster 3.

In the 4-cluster resultset, cluster 1 has high-recency, high-frequency and high-value customers; cluster 2 has high-recency, high-frequency and medium-value customers; cluster 3 has medium-recency, medium-frequency and medium-value customers; cluster 4 has low-recency, low-frequency and low-value customers.

3-cluster resultset seems interpretable. Let us plot 'Total within Sum of Squares' against k to see if we can find an elbow at cluster 3. An 'elbow' indicates the most optimal k.

```
# Plot sum within sum of squares
ss %>% ggplot(aes(x = K, y = TWSS)) +
  geom_point() +
  geom_line()+
  scale_y_continuous(labels = scales::comma)+
  scale_x_continuous(breaks = 1:j)+
  xlab("Clusters")+
  ylab("Total within Sum of Squares")
```



We do find the bend at cluster 3. or is it at cluster 2? Let us plot cluster solutions from 2 to 5 and see how they look visually.

```
# color palette for the scatter plot
palette <- c('darkred','steelblue','green4','orange', "cyan")
```

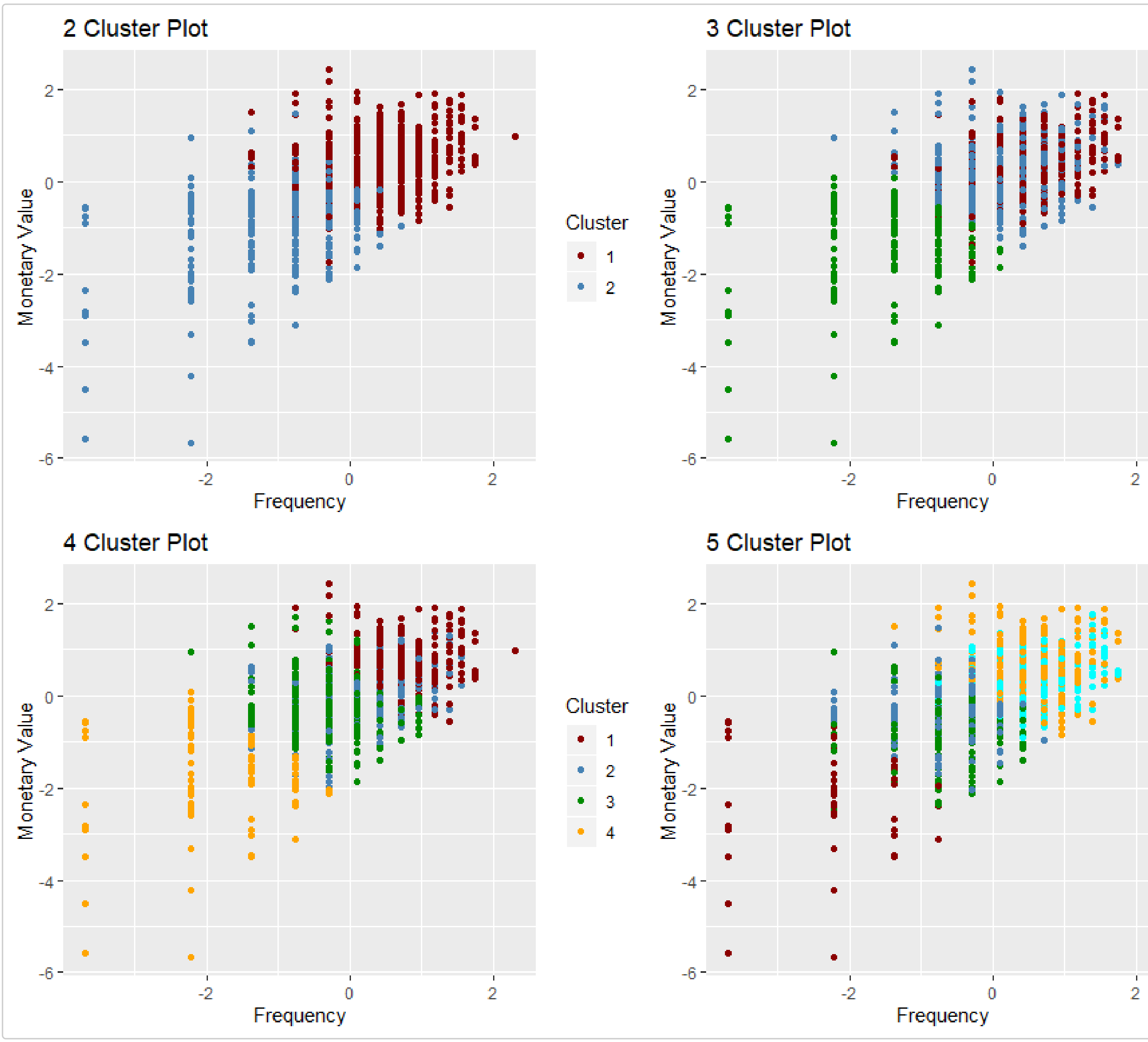
```r
# Plot RFM flor cluster 2
p1 <- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C2))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("2 Cluster Plot", sep=" "))

# Plot RFM flor cluster 3
p2<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C3))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("3 Cluster Plot", sep=" "))

# Plot RFM flor cluster 4
p3<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C4))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("4 Cluster Plot", sep=" "))

# Plot RFM flor cluster 5
p4<- customers %>% ggplot( aes(x = FrequencyZ, y = MonetaryZ))+
  geom_point(aes(colour = C5))+
  scale_colour_manual(name = "Cluster", values=palette)+
  xlab("Frequency")+
  ylab("Monetary Value")+
  ggtitle(paste("5 Cluster Plot", sep=" "))
```

```r
# Arrange plots
grid.arrange(p1, p2, p3, p4, ncol=2, nrow = 2)
```



Cluster 2 may be simplistic. Cluster 3 does look good. It may not always be possible to find distinct clusters since it depends on the data.

```
rm(ss, med, col_nm, i, j, km, palette, p1, p2, p3, p4)
```

Conclusion:

```
customers %>%
  group_by(C3) %>%
  summarize(Rmean = mean(Recency),
            Fmean = mean(Frequency),
            Mmean=mean(Monetary),
            Msum=sum(Monetary))
```

```
## # A tibble: 3 x 5
##   C3    Rmean Fmean Mmean     Msum
##   <fct> <dbl> <dbl> <dbl>    <dbl>
## 1 1      26.7  7.86 3704. 1040759.
## 2 2     184.   6.40 3184. 1165498.
## 3 3     291.   3.14  623.   90944.
```

If we look at the mean values of RFM it is clear that the customer segments are not distinct, which is perhaps the case in real life situations. It is important however to tailor an action plan based on the findings.

R-Markdown