Assignment 4

Daniele Melotti

1/13/2022

1) Import and examine the data

a) Import the CSV file into R using fread() and take a look at the data (e.g., dim, head, summary, etc.).

```
# Requiring all the necessary packages:
require(data.table)
require(lubridate)
require(dplyr)
require(ggplot2)
require(gridExtra)

data <- fread("onlineRetail.csv")
data <- na.omit(data)

# Number of variables and rows:
dim(data)
```

```
## [1] 406829      8
```

```
# Take a look at the data:
head(data)
```

```
##    InvoiceNo StockCode                         Description Quantity
## 1:    536365    85123A  WHITE HANGING HEART T-LIGHT HOLDER        6
## 2:    536365     71053                 WHITE METAL LANTERN        6
## 3:    536365    84406B      CREAM CUPID HEARTS COAT HANGER        8
## 4:    536365    84029G KNITTED UNION FLAG HOT WATER BOTTLE        6
## 5:    536365    84029E      RED WOOLLY HOTTIE WHITE HEART.        6
## 6:    536365     22752        SET 7 BABUSHKA NESTING BOXES        2
##     InvoiceDate UnitPrice CustomerID        Country
## 1: 12/1/10 8:26      2.55      17850 United Kingdom
## 2: 12/1/10 8:26      3.39      17850 United Kingdom
## 3: 12/1/10 8:26      2.75      17850 United Kingdom
## 4: 12/1/10 8:26      3.39      17850 United Kingdom
## 5: 12/1/10 8:26      3.39      17850 United Kingdom
## 6: 12/1/10 8:26      7.65      17850 United Kingdom
```

```
# Structure of the data:
str(data)
```

```
## Classes 'data.table' and 'data.frame':   406829 obs. of  8 variables:
##  $ InvoiceNo  : chr  "536365" "536365" "536365" "536365" ...
##  $ StockCode  : chr  "85123A" "71053" "84406B" "84029G" ...
##  $ Description: chr  "WHITE HANGING HEART T-LIGHT HOLDER" "WHITE METAL LANTERN" "CREAM CUPID HEARTS C
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
```

```
##  $ InvoiceDate: chr  "12/1/10 8:26" "12/1/10 8:26" "12/1/10 8:26" "12/1/10 8:26" ...
##  $ UnitPrice  : num  2.55 3.39 2.75 3.39 3.39 7.65 4.25 1.85 1.85 1.69 ...
##  $ CustomerID : int  17850 17850 17850 17850 17850 17850 17850 17850 17850 13047 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "United Kingdom" "United Kingdom" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Summary stats:
summary(data)
```

```
##    InvoiceNo          StockCode         Description           Quantity
##  Length:406829      Length:406829      Length:406829      Min.   :-80995.00
##  Class :character   Class :character   Class :character   1st Qu.:     2.00
##  Mode  :character   Mode  :character   Mode  :character   Median :     5.00
##                                                           Mean   :    12.06
##                                                           3rd Qu.:    12.00
##                                                           Max.   : 80995.00
##  InvoiceDate          UnitPrice          CustomerID      Country
##  Length:406829      Min.   :    0.00   Min.   :12346   Length:406829
##  Class :character   1st Qu.:    1.25   1st Qu.:13953   Class :character
##  Mode  :character   Median :    1.95   Median :15152   Mode  :character
##                     Mean   :    3.46   Mean   :15288
##                     3rd Qu.:    3.75   3rd Qu.:16791
##                     Max.   :38970.00   Max.   :18287
```

b) Examine the data by printing out the unique number of customers, the unique number of products purchased, as well as the unique number of transactions.

```
# Unique number of customers:
length(unique(data$CustomerID))
```

```
## [1] 4372
```

```
# Unique number of products purchased:
length(unique(data$StockCode))
```

```
## [1] 3684
```

```
# Unique number of transactions:
length(unique(data$InvoiceNo))
```

```
## [1] 22190
```

2) Compute the RFM Variables

c) Convert the InvoiceDate into a date obj. then create a variable called Recency by computing the number of days until the last day of the purchase in the dataset (i.e. Dec. 09, 2011) since last purchase for each customer.

```
data$InvoiceDate <- as_date(mdy_hm(data$InvoiceDate))

# Creating the Recency variable:
last_day <- max(data$InvoiceDate)

data_R <- data %>%
  group_by(CustomerID) %>%
  summarise(last_purchase = max(InvoiceDate)) %>%
  mutate(Recency = last_day - last_purchase)
```

d) Create a variable called Frequency and Monetary for each customer in the data.

```r
data_FM <- data %>%
  group_by(CustomerID) %>%
  summarise(Frequency = length(unique(InvoiceNo)),
            Monetary = sum(Quantity * UnitPrice))
```

3) Removing Outliers (i.e., Winsorizing)

e) Visualize the RFM variables with box plots.

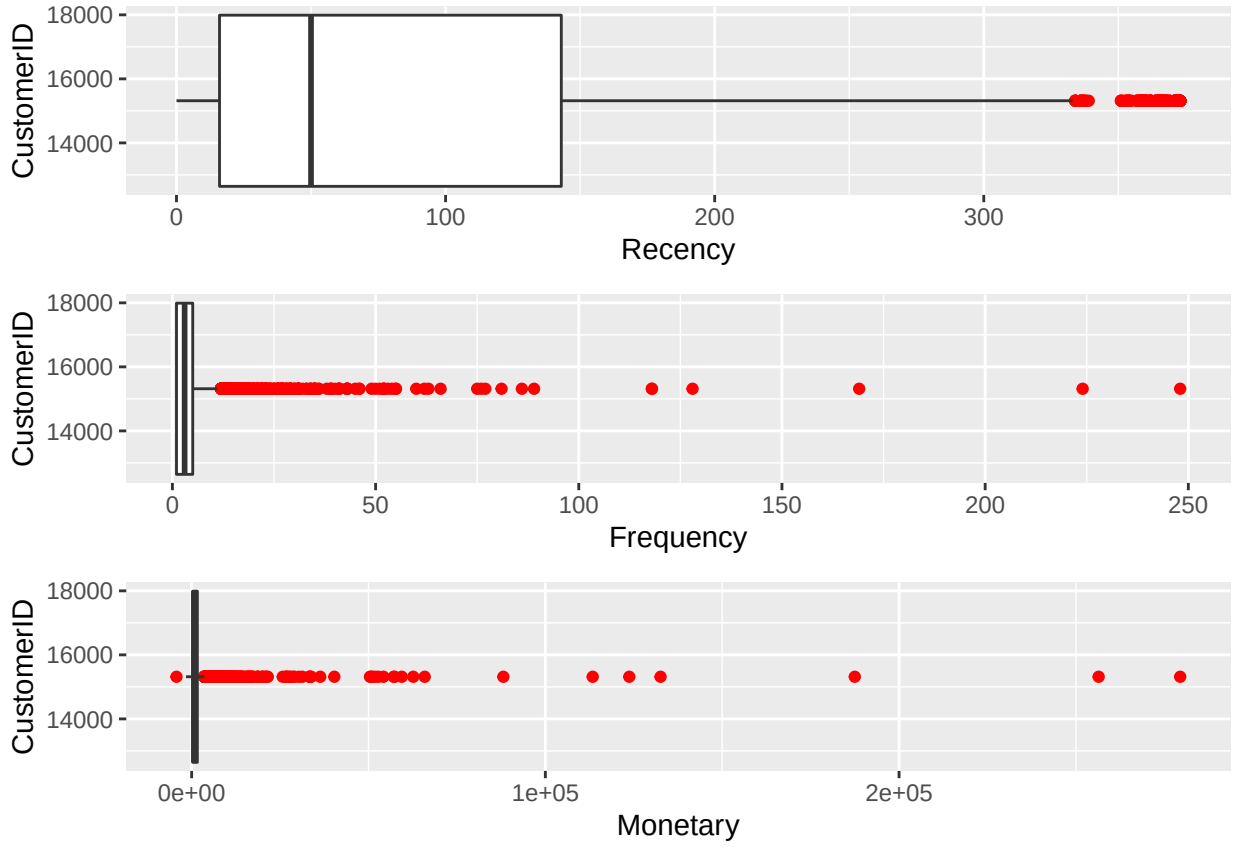
```r
# Joining R, F and M in the same dataset:
RFM <- data_R %>%
  full_join(data_FM, by = c("CustomerID" = "CustomerID"))

# Creating the boxplots:
p1 <- ggplot(RFM, aes(CustomerID, Recency)) +
  geom_boxplot(outlier.colour = "red") +
  coord_flip()

p2 <- ggplot(RFM, aes(CustomerID, Frequency)) +
  geom_boxplot(outlier.colour = "red") +
  coord_flip()

p3 <- ggplot(RFM, aes(CustomerID, Monetary)) +
  geom_boxplot(outlier.colour = "red") +
  coord_flip()

grid.arrange(p1, p2, p3)
```



f) It seems that there are extreme values in the RFM variables. Remove these extreme values/outliers by keeping only the values that are within the 99th percentile:

```
# These are the 99th percentiles for each variable:
quantile(RFM$Recency, 0.99, type = 1)
```

```
## Time difference of 368 days
```

```
quantile(RFM$Frequency, 0.99, type = 1)
```

```
## 99%
##  36
```

```
quantile(RFM$Monetary, 0.99, type = 1)
```

```
##      99%
## 17588.26
```

```
# Removing the outliers:
RFM <- RFM %>%
  filter(Recency <= quantile(Recency, .99)) %>%
  filter(Frequency <= quantile(Frequency, .99)) %>%
  filter(Monetary <= quantile(Monetary, .99))

RFM <- data.table(RFM)

summary(RFM)
```

```
##    CustomerID    last_purchase          Recency          Frequency
```

```
##  Min.   :12346   Min.   :2010-12-06   Length:4243       Min.   : 1.000
##  1st Qu.:13822   1st Qu.:2011-07-21   Class :difftime   1st Qu.: 1.000
##  Median :15308   Median :2011-10-20   Mode  :numeric    Median : 3.000
##  Mean   :15304   Mean   :2011-09-09                     Mean   : 4.306
##  3rd Qu.:16780   3rd Qu.:2011-11-22                     3rd Qu.: 5.000
##  Max.   :18287   Max.   :2011-12-09                     Max.   :36.000
##     Monetary
##  Min.   :-4287.6
##  1st Qu.:  292.6
##  Median :  639.9
##  Mean   : 1245.3
##  3rd Qu.: 1522.1
##  Max.   :11341.1
```

4) Scaling the variables

g) To prep the data for clustering, we will need to scale the features/variables. Create another data.table.obj. called RFM_scaled which contains the CustomerID and the standardized RFM variables.

```
RFM_Scaled <- RFM

RFM_Scaled$Recency <- scale(RFM_Scaled$Recency)
RFM_Scaled$Frequency <- scale(RFM_Scaled$Frequency)
RFM_Scaled$Monetary <- scale(RFM_Scaled$Monetary)

# Leaving RFM_Scaled with only the CustomerID variable and the RFM:
RFM_Scaled <- RFM_Scaled %>%
  select(-last_purchase)
```

5) Running K-Means Clustering

h) Convert RFM_Scaled to a matrix. (p.s., do not forget to remove the CustomerID from the matrix).

```
RFM_Clust <- RFM_Scaled %>%
  select(-CustomerID) %>%
  as.matrix()
```

i) Set seed at 2021 and run K-Means clustering (set k = 4).

```
set.seed(2021)
km.out <- kmeans(RFM_Clust, centers = 4)
```

j) Attach the cluster numbers (i.e. km.out$cluster) onto RFM_Scaled.

```
RFM_Clust <- cbind(RFM_Clust, km.out$cluster)
colnames(RFM_Clust) <- c(colnames(RFM_Clust)[1:3], "Cluster")

head(RFM_Clust)
```

```
##         Recency  Frequency   Monetary Cluster
## [1,]  2.4050342 -0.4844697 -0.7678390       4
## [2,] -0.9062341  0.5661308  1.8896237       1
## [3,] -0.1578670 -0.0642295  0.3403045       3
## [4,] -0.7422085 -0.6945899  0.3158324       3
## [5,]  2.2512602 -0.6945899 -0.5616544       4
## [6,] -0.5576796  1.4066113  0.1850310       1
```

6) Examining the Clusters

k) Compute the average of RFM for each cluster. Do we observe any difference between the clusters? Can we label them? Which of the clusters do you think are the most suitable for us to run target marketing campaigns and how?

```
RFM_Clust <- data.table(RFM_Clust)

RFM_Clust %>%
  group_by(Cluster) %>%
  summarise(Mean_Rec = mean(Recency),
            Mean_Fre = mean(Frequency),
            Mean_Mon = mean(Monetary))
```

```
## # A tibble: 4 x 4
##   Cluster Mean_Rec Mean_Fre Mean_Mon
##     <dbl>    <dbl>    <dbl>    <dbl>
## 1       1   -0.624    0.829    0.799
## 2       2   -0.725    3.01     3.13
## 3       3   -0.422   -0.361   -0.369
## 4       4    1.59    -0.540   -0.524
```

As we can see, there are some differences between clusters. Cluster 2 seems to be the one with the most active purchasers, as the frequency is the highest as well as the amount of money spent, while recency is very low (meaning that their last purchase was very recent). Oppositely, Cluster 4 is the one with the least active purchasers, where the value of recency is the highest (meaning that they did no purchase for a long time), while frequency and monetary are very low, meaning that they purchase rarely and spend relatively less money than customers from other clusters. In the middle, we find cluster 1 and 3, with cluster 1 holding customers that spend relatively more and more often, with a more recent latest purchase.

l) Based on the list of top selling products, you could further develop your target marketing strategies. Print out the the top 5 most selling products in terms of sales revenue (i.e., sum of sales amount = quantity x unit price) for each cluster.

```
t1 <- RFM_Scaled

# Creating a table holding scaled values, cluster numbers and CustomerID:
t2 <- cbind(t1, cluster = km.out$cluster)

# Performing an inner join between the initial raw data (with NA omitted) and table t2:
t3 <- inner_join(x = t2, y = data, by = "CustomerID")

# Adding a new column called amount = (unit price x quantity):
str(t3)
```

```
## Classes 'data.table' and 'data.frame':   341074 obs. of  12 variables:
##  $ CustomerID : int  12346 12346 12347 12347 12347 12347 12347 12347 12347 12347 ...
##  $ Recency    : num  2.405 2.405 -0.906 -0.906 -0.906 ...
##   ..- attr(*, "scaled:center")= num 90.4
##   ..- attr(*, "scaled:scale")= num 97.5
##  $ Frequency  : num  -0.484 -0.484 0.566 0.566 0.566 ...
##   ..- attr(*, "scaled:center")= num 4.31
##   ..- attr(*, "scaled:scale")= num 4.76
##  $ Monetary   : num  -0.768 -0.768 1.89 1.89 1.89 ...
##   ..- attr(*, "scaled:center")= num 1245
##   ..- attr(*, "scaled:scale")= num 1622
##  $ cluster    : int  4 4 1 1 1 1 1 1 1 1 ...
```

```
##  $ InvoiceNo  : chr  "541431" "C541433" "537626" "537626" ...
##  $ StockCode  : chr  "23166" "23166" "85116" "22375" ...
##  $ Description: chr  "MEDIUM CERAMIC TOP STORAGE JAR" "MEDIUM CERAMIC TOP STORAGE JAR" "BLACK CANDELA
##  $ Quantity   : int  74215 -74215 12 4 12 36 12 12 12 12 ...
##  $ InvoiceDate: Date, format: "2011-01-18" "2011-01-18" ...
##  $ UnitPrice  : num  1.04 1.04 2.1 4.25 3.25 0.65 1.25 1.25 1.25 1.25 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "Iceland" "Iceland" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
all_data <- t3 %>%
    mutate(Amount = Quantity * UnitPrice,
                    InvoiceNo = as.factor(InvoiceNo),
                    StockCode = as.factor(StockCode),
                    CustomerID = as.factor(CustomerID))
str(t3)
```

```
## Classes 'data.table' and 'data.frame':   341074 obs. of  12 variables:
##  $ CustomerID : int  12346 12346 12347 12347 12347 12347 12347 12347 12347 12347 ...
##  $ Recency    : num  2.405 2.405 -0.906 -0.906 -0.906 ...
##   ..- attr(*, "scaled:center")= num 90.4
##   ..- attr(*, "scaled:scale")= num 97.5
##  $ Frequency  : num  -0.484 -0.484 0.566 0.566 0.566 ...
##   ..- attr(*, "scaled:center")= num 4.31
##   ..- attr(*, "scaled:scale")= num 4.76
##  $ Monetary   : num  -0.768 -0.768 1.89 1.89 1.89 ...
##   ..- attr(*, "scaled:center")= num 1245
##   ..- attr(*, "scaled:scale")= num 1622
##  $ cluster    : int  4 4 1 1 1 1 1 1 1 1 ...
##  $ InvoiceNo  : chr  "541431" "C541433" "537626" "537626" ...
##  $ StockCode  : chr  "23166" "23166" "85116" "22375" ...
##  $ Description: chr  "MEDIUM CERAMIC TOP STORAGE JAR" "MEDIUM CERAMIC TOP STORAGE JAR" "BLACK CANDELA
##  $ Quantity   : int  74215 -74215 12 4 12 36 12 12 12 12 ...
##  $ InvoiceDate: Date, format: "2011-01-18" "2011-01-18" ...
##  $ UnitPrice  : num  1.04 1.04 2.1 4.25 3.25 0.65 1.25 1.25 1.25 1.25 ...
##  $ Country    : chr  "United Kingdom" "United Kingdom" "Iceland" "Iceland" ...
##  - attr(*, ".internal.selfref")=<externalptr>
```

```
# Subsetting for each cluster:
cluster1 <- subset(all_data, cluster == 1)
cluster2 <- subset(all_data, cluster == 2)
cluster3 <- subset(all_data, cluster == 3)
cluster4 <- subset(all_data, cluster == 4)

# Printing out the top5 selling products for each cluster:
cluster1 %>% group_by(StockCode) %>%
    arrange(-Amount) %>%
    select(CustomerID, StockCode, Amount) %>%
    head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   StockCode [4]
##   CustomerID StockCode Amount
##   <fct>      <fct>      <dbl>
## 1 12536      M          4161.
## 2 12536      M          4161.
## 3 15195      22413      3861
```

```
## 4 12798      23084      3652.
## 5 18087      22053      3203.
```

```
cluster2 %>% group_by(StockCode) %>%
    arrange(-Amount) %>%
    select(CustomerID, StockCode, Amount) %>%
    head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   StockCode [1]
##   CustomerID StockCode Amount
##   <fct>      <fct>      <dbl>
## 1 12744      M          3949.
## 2 15502      M          3156.
## 3 12744      M          2383.
## 4 12744      M          2119.
## 5 12744      M          2053.
```

```
cluster3 %>% group_by(StockCode) %>%
    arrange(-Amount) %>%
    select(CustomerID, StockCode, Amount) %>%
    head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   StockCode [4]
##   CustomerID StockCode  Amount
##   <fct>      <fct>       <dbl>
## 1 16446      23843     168470.
## 2 17846      M           2033.
## 3 16986      85099B      1790
## 4 17553      62018       1250
## 5 12669      M           1136.
```

```
cluster4 %>% group_by(StockCode) %>%
    arrange(-Amount) %>%
    select(CustomerID, StockCode, Amount) %>%
    head(5)
```

```
## # A tibble: 5 x 3
## # Groups:   StockCode [5]
##   CustomerID StockCode Amount
##   <fct>      <fct>      <dbl>
## 1 12346      23166     77184.
## 2 15098      22502     38970
## 3 12755      22328      3794.
## 4 13135      22197      3096
## 5 16692      21621      1118.
```