AS4: Exploratory Data Analysis with Clustering

By: Kevin Karnadi Kirmansjah 紀維鑫 - 109006241

---

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.)

```
data = fread('onlineRetail.csv')
data = na.omit(data)
```

```
dim(data)
```

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

```
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
```

```
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.

```
n_customers = length(unique(data$CustomerID))
n_products = length(unique(data$StockCode))
n_transactions = length(unique(data$InvoiceNo))
```

```
cat('Unique number of customers:', n_customers, '\n')
```

```
## Unique number of customers: 4372
```

```
cat('Unique number of products purchased:', n_products, '\n')
```

```
## Unique number of products purchased: 3684
```

```
cat('Unique number of transactions:', n_transactions)
```

```
## Unique number of transactions: 22190
```

2) Compute the RFM Variables

```
data_RFM = data.frame(unique(data$CustomerID))
colnames(data_RFM) = c("CustomerID")
```

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 purchase in the dataset (i.e., Dec. 09, 2011) since last purchase for each customer.

```
data$InvoiceDate = as.Date(mdy_hm(data$InvoiceDate))
lastday = ymd('20111209')

data_RFM = data %>% group_by(CustomerID) %>% summarise(lastpurchase = max(InvoiceDate))
data_RFM$Recency = lastday - data_RFM$lastpurchase
```

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

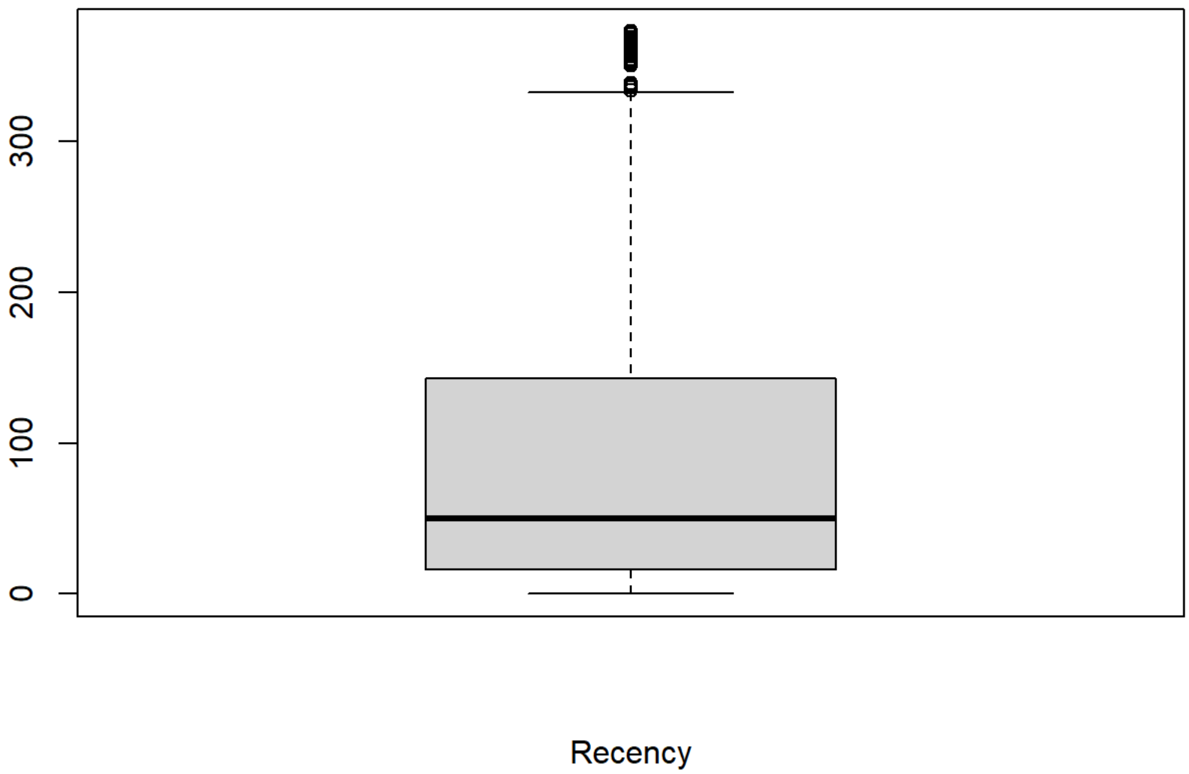
```
data$Amount = data$UnitPrice * data$Quantity
data_RFM$Frequency = with(data, as.numeric(by(InvoiceNo, CustomerID, function(x) length(unique
(x)))))
data_RFM$Monetary = with(data, as.numeric(by(Amount, CustomerID, function(x) sum(x))))
head(data_RFM)
```

```
## # A tibble: 6 × 5
##   CustomerID lastpurchase Recency  Frequency Monetary
##        <int> <date>       <drtn>       <dbl>    <dbl>
## 1      12346 2011-01-18   325 days         2       0
## 2      12347 2011-12-07     2 days         7    4310
## 3      12348 2011-09-25    75 days         4    1797.
## 4      12349 2011-11-21    18 days         1    1758.
## 5      12350 2011-02-02   310 days         1     334.
## 6      12352 2011-11-03    36 days        11    1545.
```

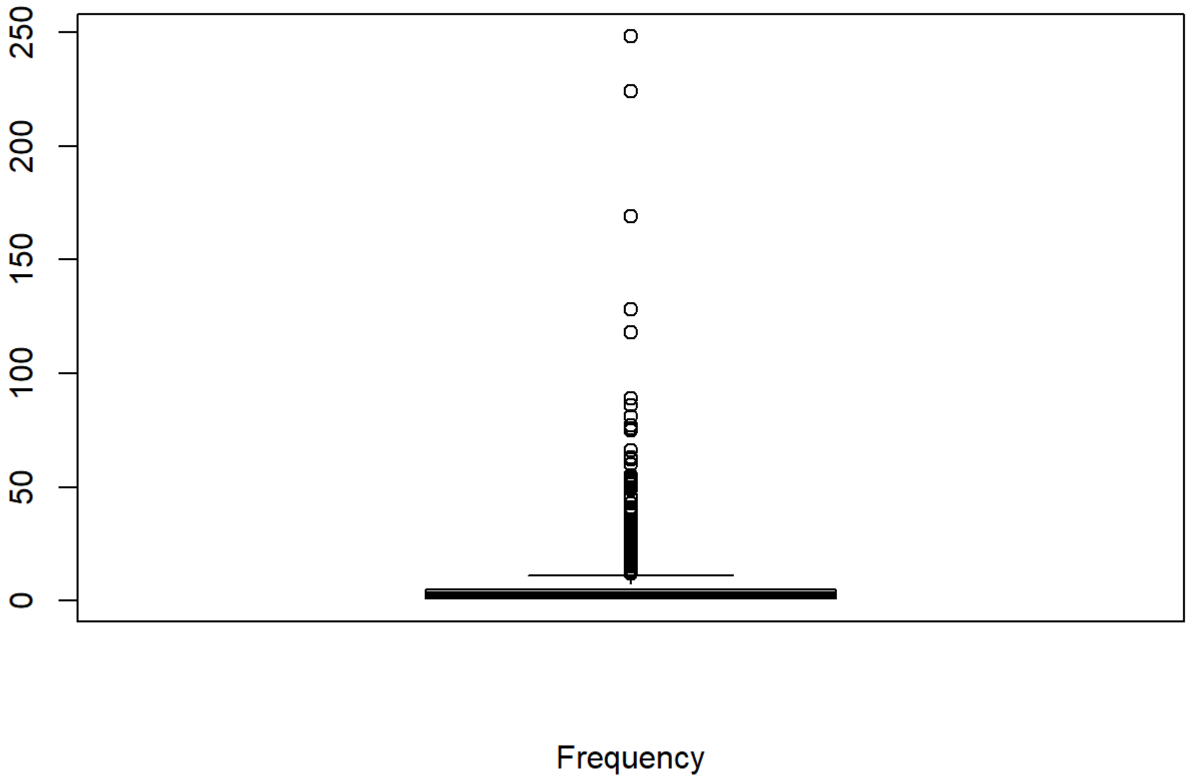
3) Removing Outliers (i.e., Winsorizing)

e) Visualize the RFM variables with box plots.

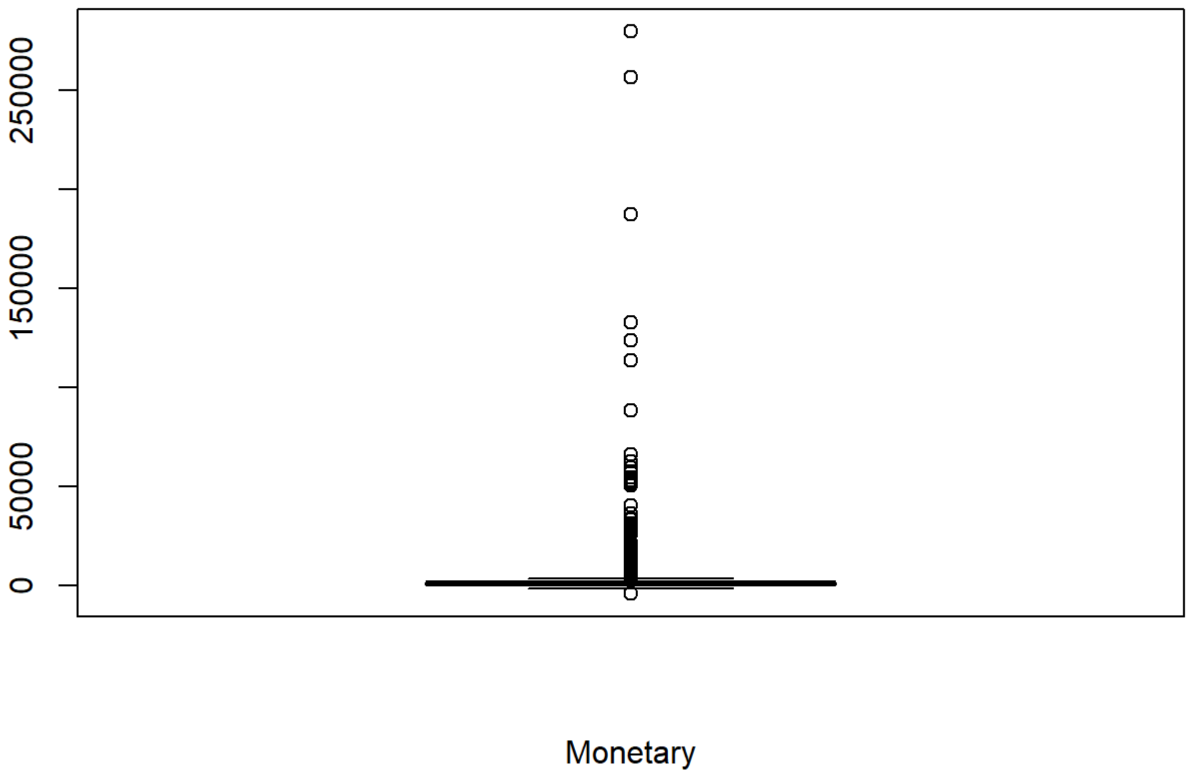
```
boxplot(data_RFM$Recency, xlab='Recency')
```




```
boxplot(data_RFM$Frequency, xlab='Frequency')
```



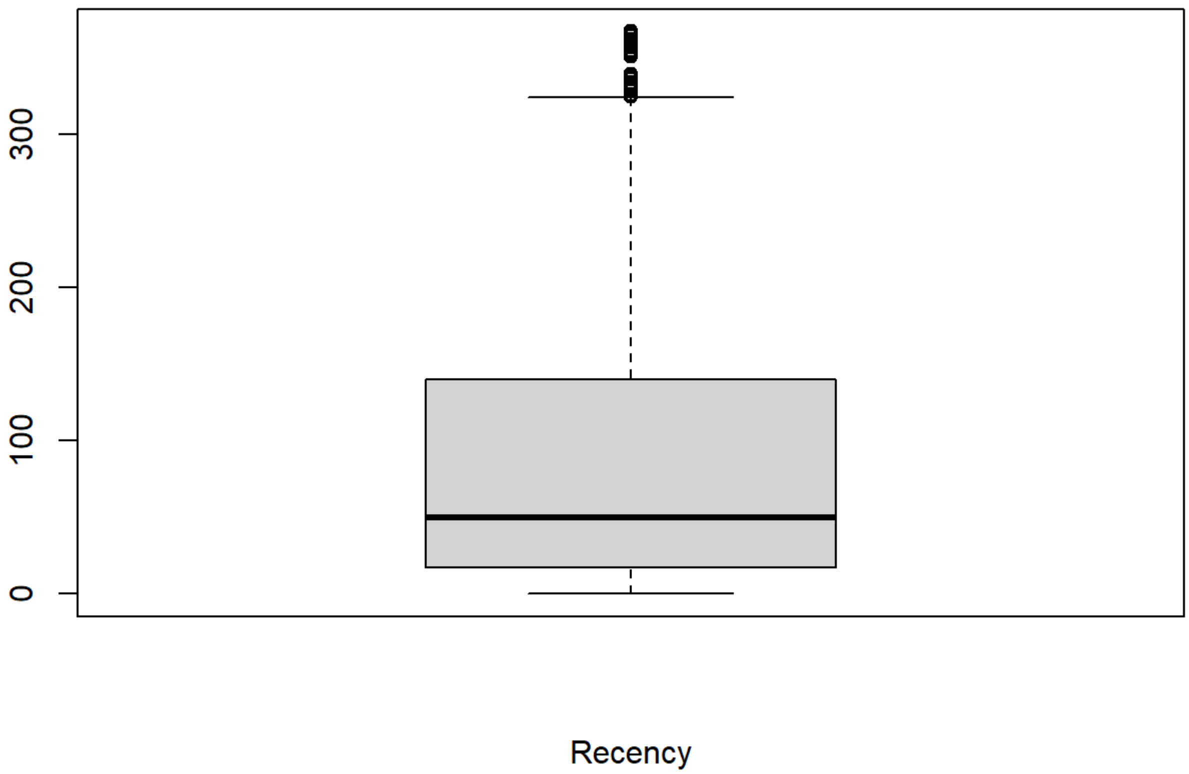

```
boxplot(data_RFM$Monetary, xlab='Monetary')
```



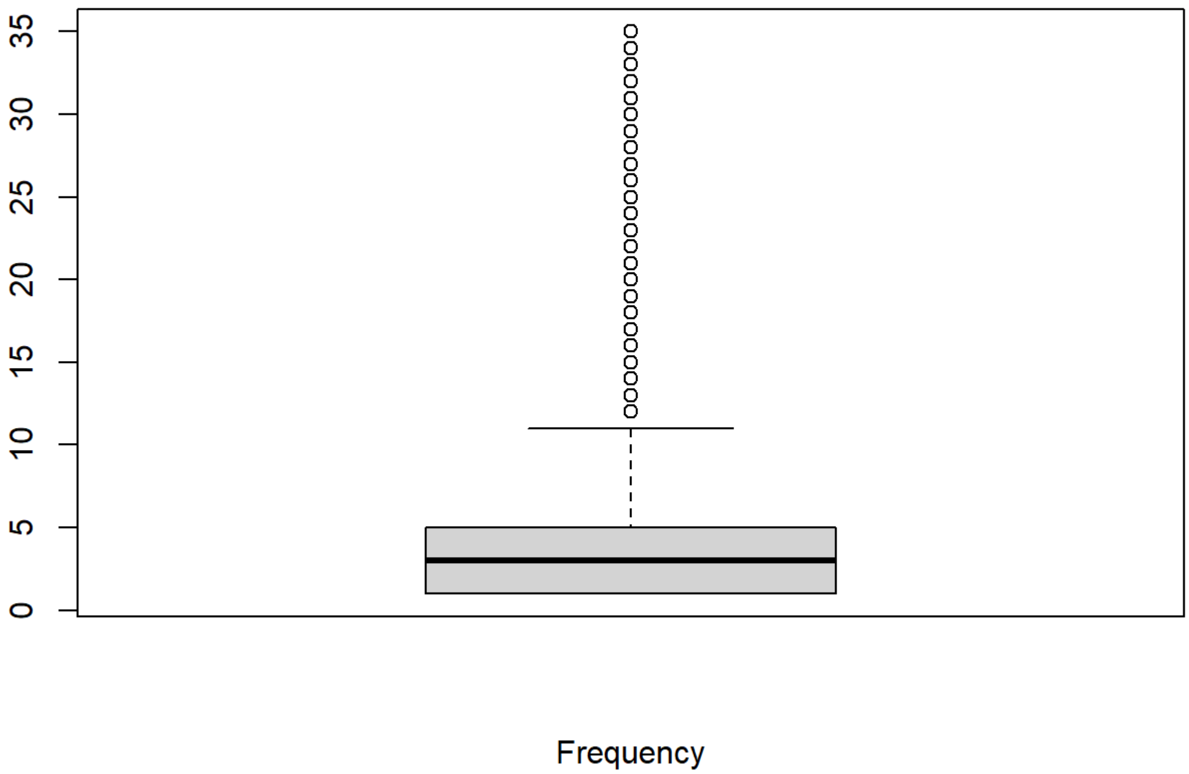
f) It seems that there are extreme values in the RFM variables. Remove these extreme values/outliers by keeping only the values that are within the 99th percentile.

```
data_RFM = data_RFM[data_RFM$Recency < quantile(as.numeric(data_RFM$Recency), 0.99),]
data_RFM = data_RFM[data_RFM$Frequency < quantile(data_RFM$Frequency, 0.99),]
data_RFM = data_RFM[data_RFM$Monetary < quantile(data_RFM$Monetary, 0.99),]
```

```
boxplot(data_RFM$Recency, xlab='Recency')
```




```
boxplot(data_RFM$Frequency, xlab='Frequency')
```




```
boxplot(data_RFM$Monetary, xlab='Monetary')
```

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 = select(data_RFM, CustomerID, Recency, Frequency, Monetary)
RFM_Scaled$Recency = scale(RFM_Scaled$Recency)
RFM_Scaled$Frequency = scale(RFM_Scaled$Frequency)
RFM_Scaled$Monetary = scale(RFM_Scaled$Monetary)
```

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_Matrix = as.matrix(RFM_Scaled[,-1])
```

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

```
set.seed(2021)
km.out = kmeans(RFM_Matrix, centers=4)
km.out
```

```
## K-means clustering with 4 clusters of sizes 2184, 227, 825, 1000
##
## Cluster means:
##      Recency  Frequency   Monetary
## 1 -0.4194622 -0.3687218 -0.3751290
## 2 -0.7253881  2.9842607  3.0883185
## 3 -0.6207827  0.8098898  0.7797611
## 4  1.5929143 -0.5402979 -0.5250696
##
## Clustering vector:
##    [1] 4 3 1 1 4 3 4 4 4 3 3 1 3 3 4 2 1 1 4 1 3 1 1 4 1 1 4 3 1 3 3 4 1 4 3 1 1
##   [38] 1 1 3 1 1 1 4 4 1 1 3 1 3 2 4 1 1 4 3 1 1 1 1 3 4 1 4 1 2 3 1 2 3 1 3 1 2
##   [75] 3 4 1 3 1 1 4 1 3 4 2 1 1 1 3 3 3 1 1 1 3 3 1 1 2 3 2 1 2 1 1 3 2 3 3 1 4
##  [112] 2 1 1 4 3 1 1 3 4 3 4 1 4 4 1 1 4 1 1 1 4 4 1 3 3 1 3 1 1 3 3 1 1 3 1 1 1
##  [149] 1 1 3 1 3 1 1 1 1 4 4 1 4 1 3 1 1 4 3 4 3 4 4 1 2 2 1 1 4 4 1 1 3 1 4 1 4
##  [186] 2 3 4 1 1 1 1 2 4 1 1 3 1 3 3 3 3 4 1 1 1 4 1 1 3 1 1 3 4 3 1 1 1 3 1 4 4
##  [223] 1 4 2 3 1 1 1 3 1 2 1 4 1 1 1 3 1 3 1 3 4 1 1 4 4 1 1 3 1 1 1 1 3 3 4 4 1
##  [260] 3 1 3 4 1 3 3 3 1 2 3 3 3 1 4 1 1 1 1 1 3 1 1 1 3 2 4 3 2 1 2 1 3 1 1 4 1
##  [297] 1 2 3 1 1 1 3 3 1 1 4 4 4 4 4 1 1 1 2 3 3 1 3 4 4 1 2 1 1 1 1 4 2 4 4 1 3
##  [334] 1 3 1 3 3 1 1 1 1 4 4 4 1 1 4 3 4 1 4 4 1 1 1 1 4 4 1 4 1 1 1 3 1 1 4 2 4
##  [371] 1 4 4 3 4 1 2 3 2 1 3 1 4 1 1 1 4 3 1 3 4 1 1 1 4 3 4 4 1 4 4 1 1 3 1 1 4
##  [408] 1 1 1 1 1 4 1 4 1 1 4 4 1 3 1 3 3 3 3 4 3 1 3 1 4 1 1 1 3 1 1 1 3 1 1 1 1
##  [445] 1 4 1 4 3 3 3 1 3 1 1 2 4 3 1 3 1 1 1 1 1 4 4 4 2 1 4 1 1 1 4 2 1 1 1 1 1
##  [482] 3 4 1 2 4 1 2 4 1 4 3 2 3 3 1 1 2 3 1 3 2 1 1 1 1 4 1 1 1 1 1 4 4 1 1 2 3
##  [519] 1 2 4 1 1 4 4 1 4 1 1 1 1 2 4 4 1 1 1 4 3 1 4 2 1 1 2 3 1 2 1 4 2 1 1 1 1
##  [556] 1 3 3 1 3 3 4 4 1 3 3 1 1 1 4 4 3 4 3 3 1 3 1 3 1 4 1 1 1 2 1 4 1 4 1 3 1
##  [593] 1 1 4 1 1 4 1 1 1 1 1 1 3 1 1 2 1 3 3 4 1 4 1 1 1 1 1 1 3 1 1 1 3 2 1 1 1
##  [630] 3 4 1 1 1 4 4 1 2 4 4 1 4 3 3 1 4 1 1 4 1 1 4 1 1 3 1 1 1 4 1 1 1 1 1 4 4
##  [667] 2 3 2 2 3 2 4 1 1 1 1 1 1 1 1 1 1 4 3 4 1 4 4 1 3 1 4 4 1 4 4 3 1 1 1 1 1
##  [704] 1 1 3 3 1 3 1 1 1 2 1 3 4 1 1 1 3 1 4 4 2 4 4 3 4 1 1 1 1 1 1 1 1 1 4 4 1
##  [741] 3 1 1 1 1 4 4 4 1 3 1 1 1 3 4 3 4 1 1 1 4 3 4 1 3 4 1 1 1 1 3 1 1 1 1 2 1
##  [778] 1 1 1 1 3 1 1 1 1 1 1 3 1 3 4 1 1 3 1 3 3 4 4 1 3 2 1 1 4 1 1 1 1 3 1 4 4
##  [815] 4 3 4 3 4 1 4 4 1 1 2 1 3 1 4 4 3 1 4 3 1 4 1 1 1 3 4 3 4 3 3 4 1 1 1 1 3
##  [852] 1 1 3 1 1 3 1 1 3 1 4 1 4 3 3 1 1 1 1 3 4 2 3 1 1 1 4 1 1 1 1 1 4 1 4 3 4
##  [889] 1 1 2 3 1 4 4 4 1 3 1 3 1 1 3 1 1 1 2 4 1 1 1 1 1 3 1 1 4 1 4 1 1 4 3 2 3
##  [926] 3 3 1 1 1 1 1 1 4 1 3 1 3 4 3 1 1 3 1 4 1 3 4 1 4 1 4 2 1 1 4 1 4 4 3 1 4
##  [963] 4 3 4 1 1 1 4 3 1 4 3 4 3 1 4 4 4 2 1 4 4 4 4 3 4 1 1 1 3 4 3 1 1 1 1 4 1
## [1000] 3 1 4 1 4 1 3 3 1 4 1 1 4 1 4 4 3 2 3 1 1 1 4 1 1 3 3 1 1 1 1 4 1 3 4 1 1
## [1037] 1 1 1 4 1 3 4 3 1 1 4 3 4 1 4 1 1 1 3 1 1 4 1 1 1 3 4 1 1 4 3 1 4 1 4 1 4
## [1074] 3 1 1 1 1 1 1 3 1 2 4 1 4 3 3 4 1 1 1 3 3 3 2 1 4 3 1 1 1 3 1 3 1 4 1 4 1
## [1111] 1 3 1 4 1 3 1 4 1 4 3 1 3 3 1 1 1 1 4 1 3 1 1 1 1 4 1 1 4 1 4 4 4 1 1 1 3
## [1148] 1 1 4 3 1 4 4 3 1 4 1 1 1 1 4 4 2 3 1 4 1 3 4 1 1 1 3 3 1 2 1 3 1 4 1 1 3
## [1185] 3 3 2 4 1 1 3 3 1 4 4 1 1 1 3 1 1 4 4 1 1 1 1 3 2 1 1 4 4 1 1 3 4 1 1 4 3
## [1222] 4 3 1 1 1 1 1 2 2 4 2 2 3 4 1 4 3 1 1 1 1 1 1 1 4 4 3 2 1 1 4 2 1 1 3 1 3
## [1259] 3 4 1 3 1 1 2 1 3 3 4 1 1 4 3 1 3 1 1 1 1 4 3 1 2 1 1 1 1 1 3 3 1 4 4 1 1
## [1296] 4 4 1 1 2 4 4 3 1 4 1 1 1 1 3 4 1 3 1 2 1 4 3 2 2 1 2 4 1 1 4 3 1 1 4 1 3
## [1333] 3 3 1 3 3 1 2 1 1 4 3 4 1 1 3 3 1 4 3 1 1 3 1 4 3 1 4 4 3 4 1 4 4 1 3 4 3
## [1370] 1 1 3 4 1 1 4 4 1 1 1 3 4 3 1 1 3 1 4 1 3 3 2 1 1 3 2 1 1 4 3 1 3 3 4 3 1
## [1407] 3 1 4 1 1 4 4 2 1 1 1 3 1 1 4 4 4 3 1 1 1 1 4 1 4 4 4 4 1 4 1 1 1 3 1 3 2
## [1444] 4 4 4 4 4 1 1 1 1 1 4 1 3 1 2 4 3 1 2 1 3 1 3 1 3 3 3 3 4 1 4 3 2 1 1 1 1
## [1481] 1 3 4 1 2 1 3 4 3 3 1 1 4 4 3 3 1 3 1 1 1 1 4 1 1 4 3 4 4 1 1 3 1 4 3 1 1
## [1518] 3 1 1 1 4 1 4 1 4 3 4 1 1 3 1 4 4 3 1 1 4 4 3 1 1 1 3 3 1 2 3 3 1 3 4 1 1
```

```
## [1555] 3 1 1 1 1 1 3 3 1 1 3 4 3 2 3 4 1 1 1 1 3 4 2 1 3 3 4 4 1 1 4 1 3 1 1 2 1
## [1592] 2 3 1 1 1 4 3 3 1 1 1 1 1 4 1 4 3 1 1 1 1 3 1 1 4 1 4 1 1 4 1 4 4 3 4 1 1
## [1629] 4 3 4 1 1 1 4 4 1 1 3 3 1 1 4 1 4 4 1 1 1 3 3 4 1 1 3 1 1 1 1 3 1 2 4 4 4
## [1666] 3 1 3 1 4 1 1 2 4 1 3 1 4 3 4 1 4 1 3 2 1 3 4 1 2 1 3 1 3 1 1 1 3 1 4 1 1
## [1703] 4 3 1 3 2 2 1 3 3 4 3 3 1 4 3 1 1 1 2 1 1 1 1 4 1 4 1 1 3 1 2 4 1 1 3 1 1
## [1740] 4 1 4 1 1 1 1 1 1 1 3 4 2 1 1 2 1 4 1 1 1 3 3 1 4 4 1 1 1 4 4 1 3 4 1 1 3
## [1777] 1 1 4 3 3 3 4 2 4 1 3 1 3 1 1 1 1 1 1 1 1 1 1 4 3 1 4 1 4 1 3 1 1 1 1 4 1
## [1814] 4 4 4 4 1 1 2 4 1 3 3 1 3 1 4 3 1 1 1 1 1 1 1 1 4 1 1 4 1 1 1 4 1 4 2 1 2 1
## [1851] 1 3 1 1 2 1 1 4 1 1 1 2 4 1 4 1 1 1 1 1 1 1 1 1 1 1 3 3 4 1 4 1 4 1 1 4 1 1
## [1888] 1 2 1 4 1 1 1 1 4 1 1 4 3 1 2 3 1 1 1 1 1 1 4 3 3 1 1 1 4 4 1 2 4 2 4 1 4
## [1925] 3 1 4 1 1 4 3 3 1 1 4 3 1 1 1 4 1 1 1 1 4 4 1 4 4 1 3 4 1 4 3 1 1 1 4 1 4
## [1962] 1 1 4 3 3 4 1 4 3 1 3 3 1 1 3 1 4 1 1 3 1 1 2 1 3 3 4 1 1 3 4 1 4 3 1 4 2
## [1999] 1 4 4 1 1 2 2 1 1 1 3 1 4 1 1 4 1 4 3 4 1 4 3 4 1 4 1 1 1 2 1 1 2 3 1 1 1
## [2036] 3 3 4 1 1 1 4 1 4 4 3 1 1 2 4 3 4 4 4 4 4 4 1 3 4 1 4 3 4 1 1 1 3 3 1 1 1
## [2073] 4 1 2 2 1 1 1 1 4 4 4 3 1 4 1 1 1 4 3 1 3 3 1 1 1 1 1 4 4 3 1 1 3 1 2 2 4
## [2110] 3 1 3 3 1 2 4 3 1 1 4 1 1 4 1 4 1 4 1 3 1 1 4 1 1 1 4 4 4 1 1 1 1 1 1 1 4
## [2147] 4 1 4 3 4 1 3 2 1 4 4 1 3 1 3 1 4 3 1 3 1 1 3 3 4 3 4 1 4 4 4 1 1 3 4 1 4
## [2184] 1 1 1 4 4 1 1 4 3 1 4 1 4 3 1 1 1 4 1 4 3 1 1 1 1 1 1 1 1 4 1 1 4 3 1 4 4
## [2221] 4 1 1 1 3 4 4 4 1 4 1 1 4 2 1 1 1 1 1 1 3 1 1 1 2 4 3 3 1 1 3 1 3 1 1 1 2
## [2258] 4 1 3 1 4 4 4 1 4 3 1 4 3 1 3 1 1 1 1 1 3 4 3 1 1 3 1 4 4 1 1 1 4 3 1 2 1
## [2295] 1 1 4 1 4 2 1 1 1 4 3 1 1 1 1 3 1 3 1 4 1 4 4 1 3 1 1 4 1 4 4 1 1 3 3 1 1
## [2332] 1 2 3 1 1 1 3 4 1 4 1 3 1 2 1 1 3 1 3 1 1 3 1 3 1 3 1 1 1 4 1 1 4 4 1 4 4
## [2369] 4 1 1 1 1 4 1 1 4 1 3 4 1 1 4 1 4 1 4 3 3 3 1 3 1 1 1 1 1 1 3 1 1 4 1 2 4
## [2406] 3 1 4 1 4 1 4 1 1 3 4 1 1 1 1 4 2 1 1 3 4 4 1 3 1 4 1 4 4 1 3 2 1 1 1 1 1
## [2443] 1 1 1 3 3 4 1 3 1 1 1 4 1 1 3 1 1 1 1 4 1 4 4 1 3 1 3 3 1 4 1 1 1 4 3 1 1
## [2480] 1 1 1 1 2 2 1 1 4 1 3 3 1 4 3 3 1 4 1 1 4 4 1 3 1 1 1 4 3 1 1 1 1 1 1 4 1
## [2517] 3 1 1 4 1 3 1 1 1 3 3 1 1 3 1 1 2 4 1 1 3 1 4 4 1 1 1 1 4 4 4 4 4 3 1 1 3
## [2554] 1 3 1 4 3 4 3 4 4 1 4 4 4 1 3 1 4 1 1 2 4 1 4 4 1 1 1 3 1 1 3 2 1 1 1 1 1
## [2591] 1 4 3 4 3 1 4 1 1 1 3 1 3 1 1 3 1 1 1 3 1 1 3 3 1 4 1 1 1 4 3 1 1 1 3 1 1
## [2628] 3 1 4 1 1 1 1 1 4 1 2 4 1 4 1 1 3 3 1 1 1 1 4 1 1 1 4 1 3 3 4 1 4 4 1 1 1
## [2665] 4 4 3 4 1 1 3 1 4 1 1 4 1 1 1 4 4 3 1 1 1 3 1 4 1 3 1 4 1 1 4 4 3 4 4 1 1
## [2702] 1 1 1 4 4 3 1 3 3 1 4 1 1 1 1 4 3 4 2 4 1 4 1 3 4 3 3 1 4 1 2 4 4 1 2 1 3
## [2739] 1 1 3 1 1 1 1 1 2 1 3 1 3 1 3 2 1 1 1 2 4 1 1 3 4 1 1 1 1 1 1 3 1 1 1 4 1
## [2776] 4 4 3 1 4 1 1 4 4 1 4 1 1 1 1 1 3 3 3 1 4 1 4 1 3 4 4 3 3 4 4 3 1 1 4 2 1
## [2813] 1 4 3 1 4 4 1 1 1 4 3 1 4 1 3 1 4 1 1 3 1 4 3 1 1 1 4 3 1 1 4 4 1 1 4 1 1
## [2850] 3 3 1 3 1 4 4 1 3 1 1 4 1 1 3 4 1 4 1 2 1 4 1 3 3 3 1 3 1 1 3 1 1 1 1 1 1
## [2887] 1 1 4 4 3 1 1 1 1 1 4 1 4 2 3 1 1 1 1 1 2 4 4 1 4 1 3 3 1 1 1 1 1 1 1 1 4
## [2924] 4 3 4 1 1 1 4 1 1 3 1 3 1 1 1 1 1 1 4 4 4 1 1 3 3 4 2 1 4 1 3 1 3 1 1 1 4
## [2961] 4 3 1 3 1 1 1 1 4 4 1 1 1 1 4 1 1 3 4 4 1 1 1 1 1 1 4 1 1 1 1 4 1 1 4 2 4 1
## [2998] 1 4 3 3 1 1 1 1 1 4 4 3 3 1 4 2 1 1 3 1 2 1 1 1 4 1 4 1 3 1 1 1 1 4 4 1 1
## [3035] 4 4 1 1 2 1 1 1 1 1 4 1 1 1 1 3 3 4 4 1 1 4 1 4 3 4 1 1 4 1 4 2 1 3 1 3 4
## [3072] 1 3 1 1 1 4 4 4 1 1 2 1 1 3 2 1 4 1 4 1 1 1 1 2 1 2 1 1 3 4 1 3 1 1 3 1 1
## [3109] 1 4 1 1 4 4 2 2 1 1 3 3 3 3 2 4 1 4 3 3 1 4 2 1 4 1 4 1 2 1 1 1 3 1 4 4 1
## [3146] 3 3 2 2 1 1 1 1 1 1 4 4 1 1 1 1 4 3 4 4 2 1 4 1 3 1 1 3 1 1 1 2 1 4 3 1 1
## [3183] 1 3 3 4 1 3 4 1 1 4 1 1 4 1 1 1 1 4 1 4 3 2 1 3 3 1 4 1 1 1 1 4 1 4 1 4 1
## [3220] 1 3 4 3 1 2 1 1 1 4 4 1 1 1 1 4 1 1 1 1 1 1 3 1 2 3 1 1 3 1 1 4 1 1 1 4 1 1
## [3257] 4 1 4 1 3 1 1 1 2 3 1 1 1 3 3 4 1 1 1 4 3 3 1 2 1 1 2 3 1 2 1 1 1 1 3 3 3
## [3294] 4 2 4 1 3 1 1 1 1 3 4 1 4 1 1 1 4 4 3 4 1 1 1 4 1 3 4 1 3 3 2 1 3 1 1 4 3
## [3331] 3 4 1 1 3 1 1 1 1 1 1 1 1 2 1 1 1 1 1 4 1 1 1 1 1 1 4 1 1 1 1 1 1 1 1 4 1
## [3368] 1 2 3 4 1 1 3 1 3 4 2 4 3 1 4 2 3 1 3 1 1 1 1 1 1 1 1 1 3 1 3 1 3 3 1 4 1
## [3405] 3 3 1 1 4 4 2 3 4 1 3 1 4 4 1 4 1 4 4 1 1 4 3 3 1 3 4 2 4 4 1 1 3 1 4 1 1
## [3442] 1 1 1 1 3 3 1 3 4 1 1 4 1 3 4 1 4 4 1 1 3 1 4 3 1 1 3 3 4 1 1 1 3 3 1 1 3
```

```
## [3479] 4 3 1 4 3 1 3 1 1 4 4 1 3 1 3 3 1 1 4 1 1 1 2 4 1 2 1 4 1 1 1 4 1 1 1 1 1
## [3516] 4 1 4 1 1 1 1 3 1 4 4 1 1 3 1 3 3 1 1 4 3 1 1 1 1 1 1 4 2 4 1 1 1 1 3 3 1
## [3553] 1 3 1 4 1 1 1 1 4 3 2 1 4 4 1 1 3 1 1 4 1 1 1 1 4 1 4 1 1 1 4 3 3 4 3 1 1
## [3590] 3 1 4 4 1 2 1 1 1 1 1 3 1 1 4 4 4 2 1 1 3 3 1 1 4 4 4 1 1 3 1 1 2 1 2 1 3
## [3627] 1 1 3 1 1 1 4 1 1 3 1 4 1 3 4 3 4 1 4 3 3 1 4 4 4 1 1 1 1 1 1 3 4 1 1 3 1
## [3664] 4 1 1 1 1 4 1 3 3 4 1 4 1 4 1 4 1 1 3 4 1 4 4 2 1 3 1 1 1 1 4 3 1 1 1 1 4
## [3701] 4 3 1 3 3 1 1 4 4 1 1 1 4 1 4 1 4 4 1 1 3 1 4 1 1 1 3 1 1 3 4 1 1 1 4 2 4
## [3738] 1 4 2 4 1 1 1 3 1 3 3 1 1 3 4 1 4 3 1 4 1 1 2 3 2 1 1 4 1 4 1 1 1 1 3 1 3
## [3775] 4 1 1 3 1 1 4 1 1 3 4 1 3 1 1 3 2 1 4 3 1 1 3 1 3 1 1 3 1 3 4 3 3 3 1 3 4
## [3812] 1 1 3 1 3 3 2 4 1 1 1 1 3 4 4 3 4 3 1 4 3 2 4 4 4 1 1 4 4 2 4 2 1 3 3 1 3
## [3849] 1 3 2 1 1 1 1 1 3 3 1 3 4 1 4 3 4 3 2 3 1 1 1 4 1 4 1 1 3 4 1 1 1 4 1 1 4
## [3886] 1 1 1 1 1 4 3 1 1 4 1 1 3 3 1 1 1 2 1 3 1 1 3 4 1 3 3 4 1 1 1 1 1 3 3 1 4
## [3923] 1 1 3 4 2 1 4 3 2 4 3 4 3 1 2 1 3 1 3 4 3 4 4 1 4 4 4 1 1 1 1 1 3 4 4 4 1
## [3960] 4 3 1 1 4 1 4 4 1 1 1 4 1 4 1 1 1 1 2 1 4 3 1 1 1 3 1 1 1 1 1 3 4 3 1 1 4
## [3997] 1 4 1 1 3 4 4 1 1 4 1 1 3 1 4 4 4 1 1 2 1 4 1 4 4 1 1 4 1 4 4 1 4 3 1 1 4
## [4034] 1 1 3 4 4 4 1 1 1 1 1 1 1 4 1 1 4 1 1 1 1 4 1 2 1 1 3 1 4 4 1 2 4 1 1 3 4
## [4071] 1 3 1 1 4 3 4 4 3 3 3 4 3 1 1 1 1 4 4 3 1 2 3 3 4 1 3 1 1 1 1 4 4 2 1 1 4
## [4108] 3 1 2 4 4 4 3 1 1 1 1 4 1 4 1 1 4 2 4 1 1 3 3 1 1 1 1 1 1 1 1 1 4 1 3 1 4
## [4145] 3 3 1 1 3 2 1 1 4 3 3 3 3 1 4 1 4 3 1 4 4 1 4 1 1 2 1 4 4 1 1 1 3 1 4 4 1
## [4182] 1 1 4 3 4 1 1 2 4 2 2 4 1 2 3 4 1 4 1 1 1 4 1 3 3 3 1 1 1 4 3 1 1 4 3 3 3
## [4219] 1 1 1 1 1 4 1 3 1 1 1 1 1 4 4 1 3 1
##
## Within cluster sum of squares by cluster:
## [1] 732.3106 845.5302 808.7352 558.0792
##  (between_SS / total_SS =  76.8 %)
##
## Available components:
##
## [1] "cluster"      "centers"      "totss"        "withinss"     "tot.withinss"
## [6] "betweenss"    "size"         "iter"         "ifault"
```

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

```
RFM_Scaled = cbind(RFM_Scaled, "Cluster"=km.out$cluster)
data_RFM = cbind(data_RFM, "Cluster"=km.out$cluster)
```

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?

```
group_by(data_RFM, Cluster) %>%
  summarise(N = n(),
            Mean_Recency = mean(Recency),
            Mean_Frequency = mean(Frequency),
            Mean_Monetary = mean(Monetary))
```

```
## # A tibble: 4 × 5
##   Cluster     N Mean_Recency   Mean_Frequency Mean_Monetary
##     <int> <int> <drtn>                  <dbl>         <dbl>
## 1       1  2184  49.36172 days           2.55          639.
## 2       2   227  19.64317 days          18.4          6210.
## 3       3   825  29.80485 days           8.13         2497.
## 4       4  1000 244.85000 days           1.74          397.
```

Customers from cluster 1 can be considered as the most common group of customers, because it it the largest cluster, with average levels of recency, frequency, and monetary.

Customers from cluster 2 can be considered as the highest-spending customers. Not to mention, their recency levels are also low, meaning that they have recently made a transaction from the shop.

Customers from cluster 3 can be considered as the semi-high-spending customers, because its recency, monetary, and frequency levels are just a little lower than cluster 2.

Customers from cluster 4 has the highest level of recency, meaning that they haven't made a transaction from the shop for a very long time.

I personally think that the marketing campaigns should be targetted into customers from cluster 4, because they haven't made a transaction for a very long time, and it would be very great if we can get their interest towards our products again.

Maybe we can do this by using a strategy such as old customer discount, which gives a 50% discount code for customers who haven't made a transaction for more than 180 days.

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

```
Cluster_Data = data_RFM[,c('CustomerID', 'Cluster')]
data_joined = left_join(data, Cluster_Data, by=c('CustomerID'='CustomerID'))
Cluster_Sales = data_joined %>%
  na.omit() %>%
  select(StockCode, Description, Amount, Cluster) %>%
  group_by(StockCode, Description, Cluster) %>%
  summarise(Total_Sales=sum(Amount), .groups='drop')
```

Cluster 1:

```
subset(Cluster_Sales, Cluster=="1") %>%
  arrange(desc(Total_Sales)) %>%
  select(-'Cluster') %>%
  head(5)
```

```
## # A tibble: 5 × 3
##   StockCode Description                        Total_Sales
##   <chr>     <chr>                                    <dbl>
## 1 POST      POSTAGE                                 14173.
## 2 22423     REGENCY CAKESTAND 3 TIER                13995.
## 3 85123A    WHITE HANGING HEART T-LIGHT HOLDER      12702.
## 4 84879     ASSORTED COLOUR BIRD ORNAMENT           10858.
## 5 85099B    JUMBO BAG RED RETROSPOT                 10547.
```

Cluster 2:

```
subset(Cluster_Sales, Cluster=="2") %>%
  arrange(desc(Total_Sales)) %>%
  select(-'Cluster') %>%
  head(5)
```

```
## # A tibble: 5 × 3
##   StockCode Description                        Total_Sales
##   <chr>     <chr>                                    <dbl>
## 1 22423     REGENCY CAKESTAND 3 TIER                24107.
## 2 47566     PARTY BUNTING                           14571.
## 3 85099B    JUMBO BAG RED RETROSPOT                 13666.
## 4 85123A    WHITE HANGING HEART T-LIGHT HOLDER      13187.
## 5 POST      POSTAGE                                 12283.
```

Cluster 3:

```
subset(Cluster_Sales, Cluster=="3") %>%
  arrange(desc(Total_Sales)) %>%
  select(-'Cluster') %>%
  head(5)
```

```
## # A tibble: 5 × 3
##   StockCode Description                        Total_Sales
##   <chr>     <chr>                                    <dbl>
## 1 POST      POSTAGE                                 26330.
## 2 22423     REGENCY CAKESTAND 3 TIER                22714.
## 3 85123A    WHITE HANGING HEART T-LIGHT HOLDER      19201.
## 4 47566     PARTY BUNTING                           17908.
## 5 85099B    JUMBO BAG RED RETROSPOT                 15616.
```

Cluster 4:

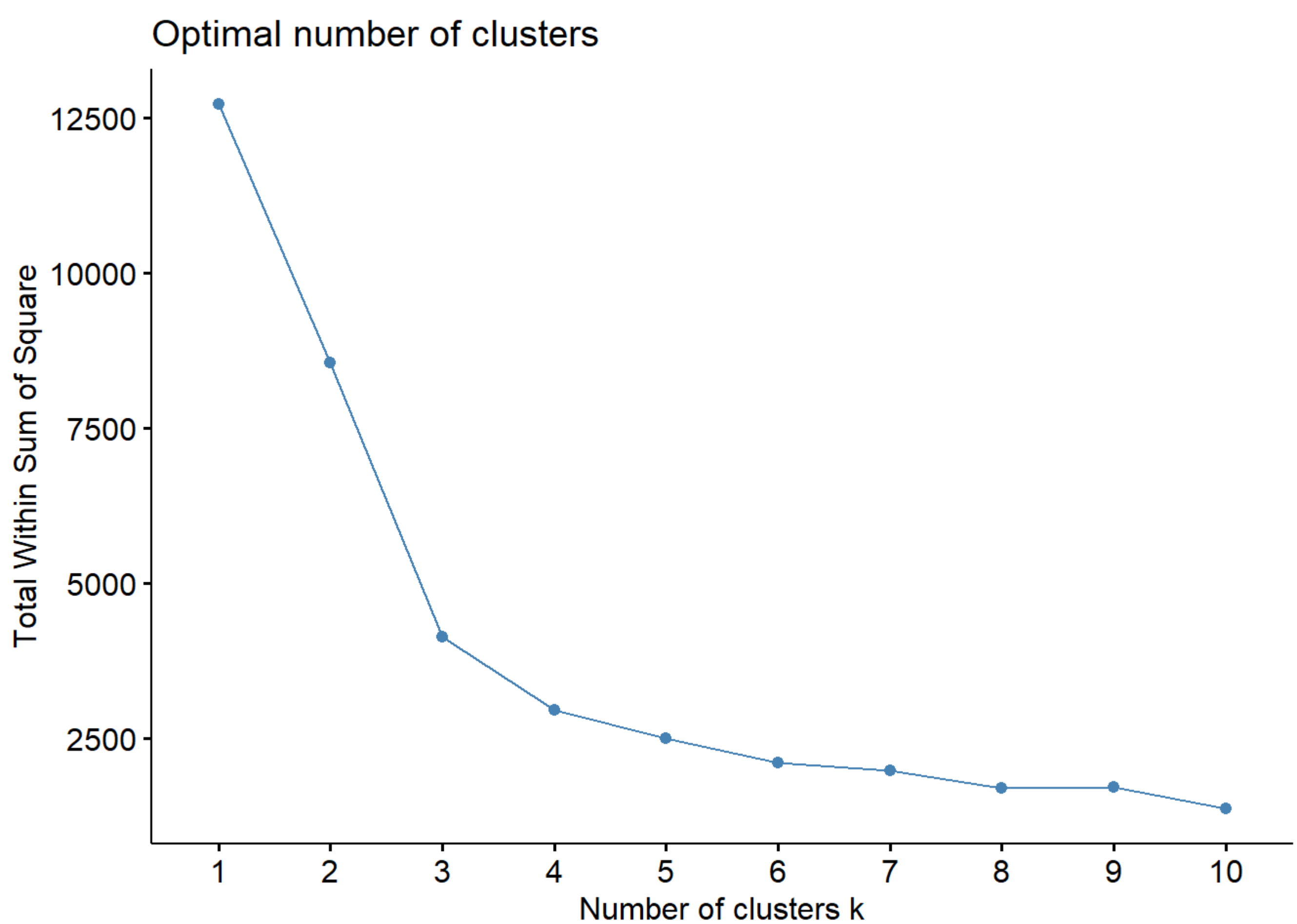
```
subset(Cluster_Sales, Cluster=="4") %>%
  arrange(desc(Total_Sales)) %>%
  select(-'Cluster') %>%
  head(5)
```

```
## # A tibble: 5 × 3
##   StockCode Description                        Total_Sales
##   <chr>     <chr>                                    <dbl>
## 1 22502     PICNIC BASKET WICKER 60 PIECES          39620.
## 2 22423     REGENCY CAKESTAND 3 TIER                 6992.
## 3 85123A    WHITE HANGING HEART T-LIGHT HOLDER       5410.
## 4 47566     PARTY BUNTING                            4778
## 5 POST      POSTAGE                                  4100.
```

EC3) When using k-means clustering, the number of clusters should be predetermined, and this should be firmly backed by domain knowledge or a proven theory. However, we could also take a data-driven approach by using methods such as the Elbow method or the Silhouette method which can easily be done using the packages like factoextra and NbClust. Explain whether k = 4 is a reasonable decision using the Elbow/Silhouette method.

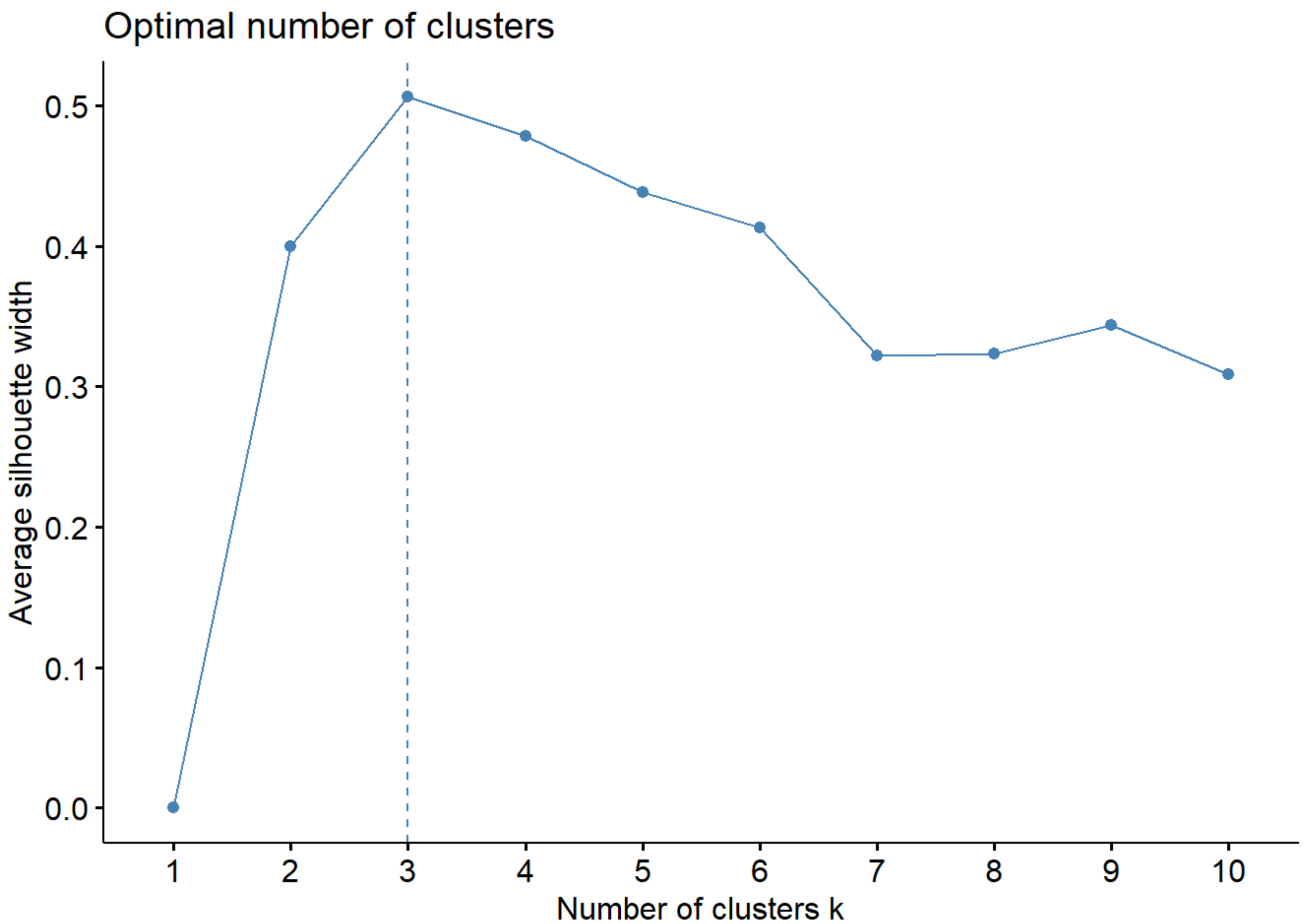
Elbow Method:

```
fviz_nbclust(RFM_Matrix, kmeans, method='wss')
```



Silhouette Method:

```
fviz_nbclust(RFM_Matrix, kmeans, method='silhouette')
```



According to the Elbow and Silhouette methods, the optimal number of clusters is 3.