Classic Retail Customer Segmentation

Prof B

2/18/2021

#Introduction

Recency, frequency and monetary value of purchase are classic marketing variables that are used often in what is called ‘RFM Customer Segmentation.’ In addition we also have data on the duration of the customer’s relationship with the firm. Lets do some data driven segmentation on this dataset using k-means clustering.

#Data Ingestion

We begin by loading the packages required for our analysis, with the code below:

```
#If document rendering becomes time consuming due to long computations you can use knitr caching to improve performanc
e.
library(knitr)
opts_chunk$set(cache = TRUE)

required_packages = c(
  # Add to this list the packages that you will use - if unavailable, it will be
  # automatically installed

  "rmarkdown",
  "dplyr",
  "readr",
  "tidyr",
  "knitr",
  "data.table",
  "lubridate",
  "rlang",
  "ggplot2",
  "cluster",
  "skimr",
  "GGally"
    )

packages_to_install = required_packages[!(required_packages %in%
                                                installed.packages()[, 1])]

if (length(packages_to_install) > 0) {
  install.packages(packages_to_install)
}

suppressPackageStartupMessages({
  sapply(required_packages, require, character.only = TRUE)
})

#Note: library(package) and require(package) both load the namespace of the package with name package and attach it on
 the search list. require is designed for use inside other functions; it returns FALSE and gives a warning (rather than
an error as library() does by default) if the package does not exist.

# Note: The sapply() function works like lapply(), which is listApply, but it tries to simplify the output to the most
 elementary data structure that is possible. And indeed, sapply() is a ‘wrapper’ function for lapply().
```

##Importing data

We now import the data required for our analysis, using the `read_csv()` command from `readr` package. We import the two files as shown below:

```
setwd("C:\\Users\\rbapna\\Dropbox\\cmba5832-bizAnalyticsCompAdv-2021Spring\\class 2 - descriptive")

customerRFM <- read_csv('CA-customerData.csv')
```

##Data Structure & Summarization

We first look at the structure of the table to get an idea about the variables and their data types:

```
str(customerRFM)
```

```
## spec_tbl_df [2,000 × 5] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ Monetary : num [1:2000] 138 240 97 348 239 253 118 326 249 294 ...
##  $ Recency  : num [1:2000] 28 14 6 2 20 10 16 14 12 10 ...
##  $ Frequency: num [1:2000] 3 1 2 7 2 4 1 2 3 12 ...
##  $ tenure   : num [1:2000] 40 14 10 38 28 20 16 22 20 58 ...
##  $ ID#      : num [1:2000] 2 30 59 89 96 120 128 131 139 177 ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   Monetary = col_double(),
##   ..   Recency = col_double(),
##   ..   Frequency = col_double(),
##   ..   tenure = col_double(),
##   ..   `ID#` = col_double()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

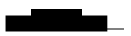
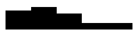
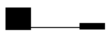
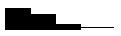
Let us now look at the summary statistics of each of the table, using the `skim` command in R:

```
skim(customerRFM)
```

Data summary

Name	customerRFM
Number of rows	2000
Number of columns	5
Column type frequency:	
numeric	5
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Monetary	0	1	206.79	101.34	15	126.75	207	281.25	477	
Recency	0	1	13.52	8.24	2	8.00	12	16.00	36	
Frequency	0	1	4.01	3.55	1	1.00	2	6.00	12	
tenure	0	1	27.42	18.74	2	14.00	22	38.00	99	
ID#	0	1	24753.23	14425.50	2	12699.25	24201	37299.50	49962	

##Missing value treatment (if necessary)

#Data normalization

##Min-max normalization

One of the main requirements of clustering is that the variables used for clustering must be normalized, ie. the scale of all variables must be the same.

For our dataset, we use a common normalization technique known as min-max normalization. The normalization function is as defined below:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

We define a function in R for the `min-max` normalization

```
# Function to perform min-max normalization
mmnormalize <- function(x){
  n_max <- max(x, na.rm = TRUE)
  n_min <- min(x, na.rm = TRUE)
  mmnormalized <- (x - n_min)/(n_max - n_min)
  return(mmnormalized)
}
```

Now that we have the normalized dataset, we remove from our data the columns that will not be used for clustering, namely `ID#`

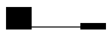
```
customerRFM_for_clustering <- select(customerRFM,-"ID#")
customerRFM_for_clustering  <- mutate_all(customerRFM,.funs = mmnormalize)

skim(customerRFM_for_clustering)
```

Data summary

Name	customerRFM_for_clusterin…
Number of rows	2000
Number of columns	5
Column type frequency:	
numeric	5
Group variables	None

Variable type: numeric

skim_variable	n_missing	complete_rate	mean	sd	p0	p25	p50	p75	p100	hist
Monetary	0	1	0.42	0.22	0	0.24	0.42	0.58	1	
Recency	0	1	0.34	0.24	0	0.18	0.29	0.41	1	
Frequency	0	1	0.27	0.32	0	0.00	0.09	0.45	1	
tenure	0	1	0.26	0.19	0	0.12	0.21	0.37	1	
ID#	0	1	0.50	0.29	0	0.25	0.48	0.75	1	

#Clustering

Cluster analysis or **clustering** is the task of grouping a set of objects in such a way that objects in the same group (called a cluster) are more similar (in some sense or another) to each other than to those in other groups (clusters). It is a main task of exploratory data mining, and a common technique for statistical data analysis.

There are various algorithms to perform clustering such as Hierarchical clustering, k-means, DBSCAN etc. We perform the **k-means** algorithm, as it is a simple and effective solution to most clustering problems:

##k-means algorithm

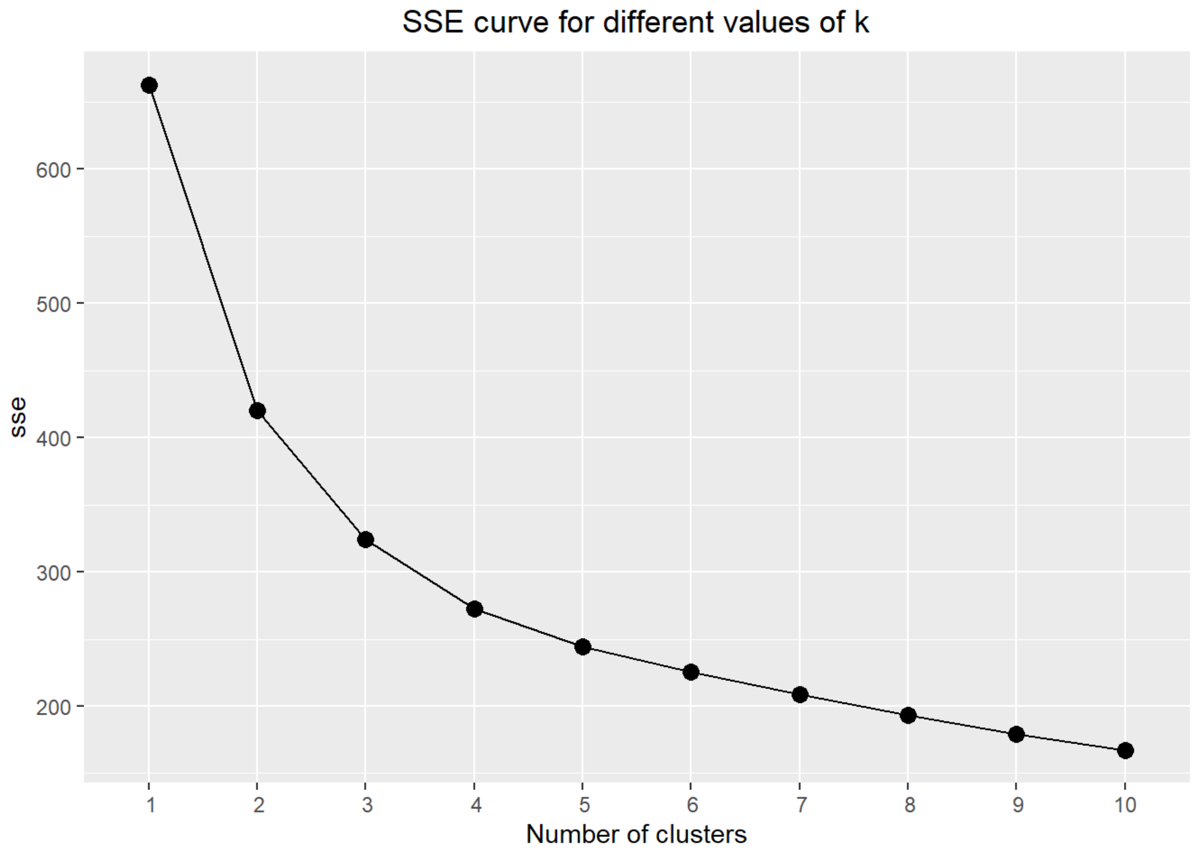
Let us evaluate what the ideal number of clusters is using the SSE curve:

```
SSE_curve <- c()
for (n in 1:10) {
    kcluster <- kmeans(customerRFM_for_clustering, centers = n, nstart = 10)
    sse <- sum(kcluster$withinss)
    SSE_curve[n] <- sse
}

print("SSE curve for the ideal k value")
```

```
## [1] "SSE curve for the ideal k value"
```

```
#plot(1:10, SSE_curve, type="b", xlab="Number of Clusters", ylab="SSE")
ggplot(data = data.frame(k = 1:10, sse = SSE_curve), aes(x = k, y = sse)) +
  geom_line() +
  geom_point(size = 3) +
  ggtitle("SSE curve for different values of k") +
  theme(plot.title = element_text(hjust = 0.5)) +
  scale_x_discrete(limits = 1:10) +
  xlab("Number of clusters")
```



We see that that the decrease in SSE stops being significant with `k` >= 4. Hence, we choose `k` = 4 as our desired number of clusters.

```
# Set seed for reproducible results - this ensures that we get the same results
# every time we run the code
set.seed(42)

#Perform k-means algorithm to identify 6 distinct clusters
km_all <- kmeans(customerRFM_for_clustering,
                 centers = 4,
                 nstart = 10)
```

Now that we have obtained our clusters, we add this information to our original dataset for further analysis.

```
customerRFM <- bind_cols(customerRFM, data.frame(cluster = km_all$cluster))
head(customerRFM)
```

```
## # A tibble: 6 × 6
##   Monetary Recency Frequency tenure `ID#` cluster
##      <dbl>   <dbl>     <dbl>  <dbl> <dbl>   <int>
## 1      138      28         3     40     2       2
## 2      240      14         1     14    30       1
## 3       97       6         2     10    59       1
## 4      348       2         7     38    89       3
## 5      239      20         2     28    96       1
## 6      253      10         4     20   120       1
```

#Segment Analysis

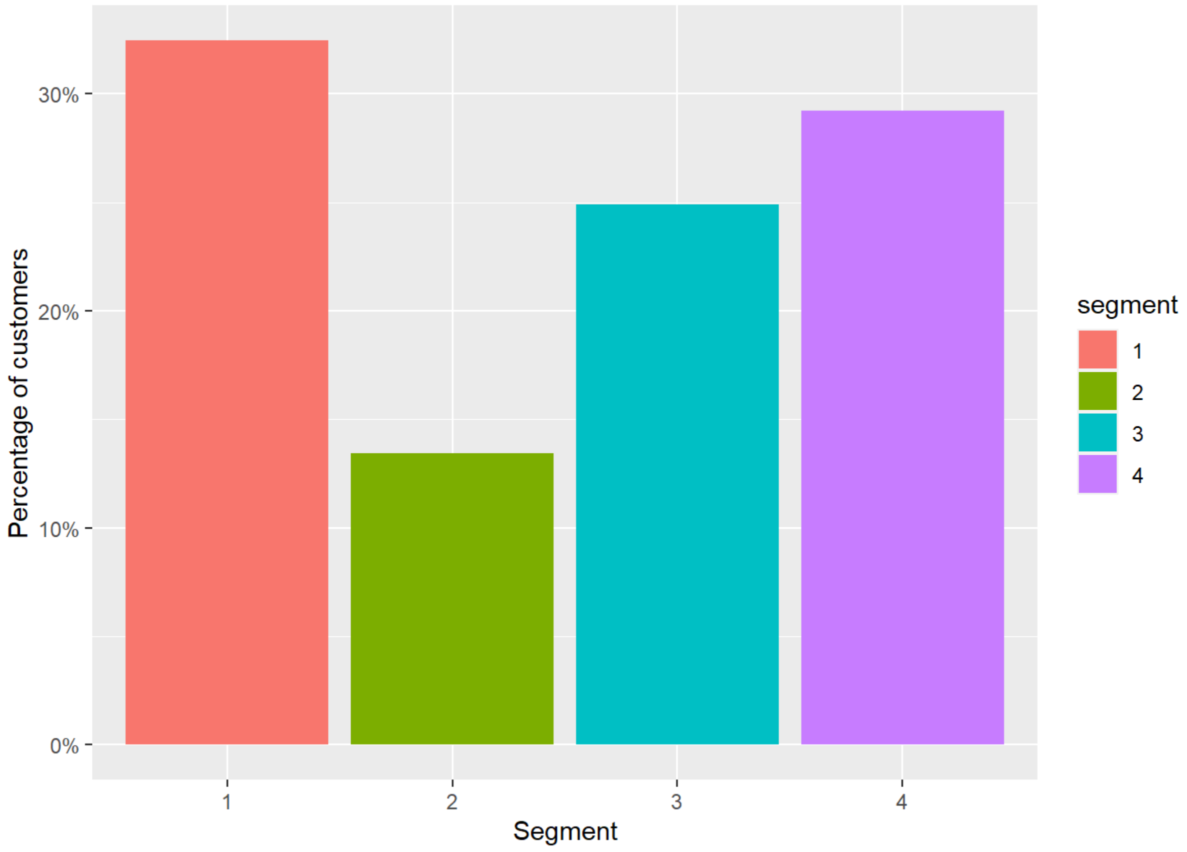
We load the file that has the *cluster* attributed to each customer in the sampled data.

```
segment_df <- customerRFM %>%
  mutate(segment = as.factor(cluster))


segment_df$cluster <- as.factor(segment_df$cluster)
```

Let us visualise the percentage of customers in each segment. We visualize using the `ggplot()` function in R, from the powerful `ggplot2` package.

```
ggplot(segment_df, aes(x = segment)) +
  geom_bar(aes(y = (..count..)/sum(..count..), fill = segment)) +
  scale_y_continuous(labels = scales::percent) +
  ylab("Percentage of customers") +
  xlab("Segment")
```



Let us examine the 'centroid' of each cluster

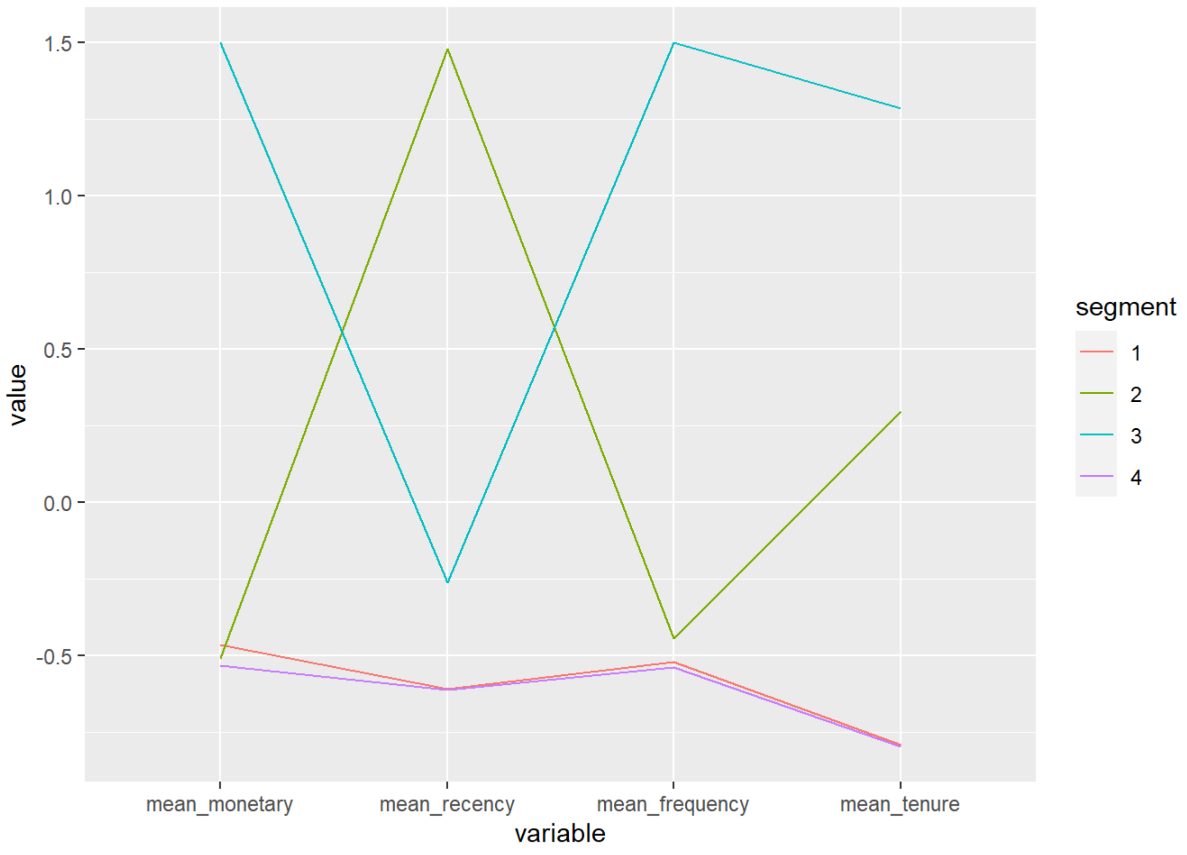
```
segment_summary <- segment_df %>% group_by(segment) %>% summarise(mean_monetary = mean(Monetary),
                                              mean_recency = mean(Recency),
                                              mean_frequency = mean(Frequency),
                                              mean_tenure = mean(tenure),
                                              n = n())

segment_summary
```

```
## # A tibble: 4 × 6
##   segment mean_monetary mean_recency mean_frequency mean_tenure     n
##   <fct>           <dbl>        <dbl>          <dbl>       <dbl> <int>
## 1 1                178.         10.4           2.16        15.9   649
## 2 2                175.         28.1           2.43        34.9   269
## 3 3                299.         13.3           9.52        52.2   498
## 4 4                174.         10.4           2.08        15.7   584
```

Let us visualize this using a parallel coordinates plot

```
ggparcoord(segment_summary,
    columns = 2:5, groupColumn = 1)
```



1.5
1.0
0.5
0.0
-0.5
value
mean_monetary
mean_recency
mean_frequency
mean_tenure
variable
segment
1
2
3
4