Online Retail - Clustering

Markus Stellwag

09.01.2020

Table of Contents

1 Context and Goal

This data set comprises all transactions occurring between 01-12-2010 and 09-12-2011 for a UK-based and registered non-store online retail.

We aim to cluster the customers based on a RFM Analysis so that the company can target its customers efficiently.

```
library(data.table)

library(ggplot2)
library(dplyr)

library(lubridate)

library(rlang)

library(rfm)

library(DMwR)

library(scales)
```

2 Cleaning the data frame

In the first step we read the .csv file 'OnlineRetail.csv' and check the structure as well as head and tail to see if the data set was loaded properly.

```
retail <- read.csv('OnlineRetail.csv', sep = ',', header = TRUE)

str(retail)

## 'data.frame':    541909 obs. of  8 variables:
##  $ InvoiceNo  : Factor w/ 25900 levels "536365","536366",..: 1 1 1 1 1 1 1 2 2 3 ...
##  $ StockCode  : Factor w/ 4070 levels "10002","10080",..: 3538 2795 3045 2986 2985 1663 8
01 1548 1547 3306 ...
##  $ Description: Factor w/ 4224 levels ""," 4 PURPLE FLOCK DINNER CANDLES",..: 4027 40
35 932 1959 2980 3235 1573 1698 1695 259 ...
##  $ Quantity   : int  6 6 8 6 6 2 6 6 6 32 ...
##  $ InvoiceDate: Factor w/ 23260 levels "01-02-2011 08:23",..: 607 607 607 607 607 607 607
608 608 609 ...
##  $ 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    : Factor w/ 38 levels "Australia","Austria",..: 36 36 36 36 36 36 36 36 36 36 ...

head(retail)

##   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 01-12-2010 08:26      2.55      17850 United Kingdom
## 2 01-12-2010 08:26      3.39      17850 United Kingdom
## 3 01-12-2010 08:26      2.75      17850 United Kingdom
## 4 01-12-2010 08:26      3.39      17850 United Kingdom
## 5 01-12-2010 08:26      3.39      17850 United Kingdom
## 6 01-12-2010 08:26      7.65      17850 United Kingdom
```

```
tail(retail)
```

```
##        InvoiceNo StockCode                    Description Quantity
## 541904    581587     23256     CHILDRENS CUTLERY SPACEBOY        4
## 541905    581587     22613     PACK OF 20 SPACEBOY NAPKINS       12
## 541906    581587     22899    CHILDREN'S APRON DOLLY GIRL        6
## 541907    581587     23254   CHILDRENS CUTLERY DOLLY GIRL        4
## 541908    581587     23255 CHILDRENS CUTLERY CIRCUS PARADE        4
## 541909    581587     22138   BAKING SET 9 PIECE RETROSPOT        3
##             InvoiceDate UnitPrice CustomerID Country
## 541904 09-12-2011 12:50      4.15      12680  France
## 541905 09-12-2011 12:50      0.85      12680  France
## 541906 09-12-2011 12:50      2.10      12680  France
## 541907 09-12-2011 12:50      4.15      12680  France
## 541908 09-12-2011 12:50      4.15      12680  France
## 541909 09-12-2011 12:50      4.95      12680  France
```

In the next step we want to prepare the data set. In doing so we check the data frame on duplicates in terms of rows. The logical data type 'FALSE' indicates the number of unique rows. 'TRUE' indicates the number of double rows.

```
retail_duplicates <- retail %>% duplicated() %>% table()
print(retail_duplicates)
```

```
## .
##  FALSE   TRUE
## 536641   5268
```

In conclusion we can remove double rows by using the function distinct. If we check the variable updated_retail depicts we can see that the data set does not contain any 'TRUEs'' anymore. All double rows were removed successfully. Now we want to see how many NAs the data frame contains. In order to proceed with the is.na function, we have to convert the column 'CustomerID' into a character class. With the function na.omit we can omit all rows containing NAs. As a result we have a clean data frame and can move on to the next step.

```
retail <- retail %>% distinct()

updated_retail <- retail %>% duplicated() %>% table()

print(updated_retail)
```

```
## .
##  FALSE
## 536641
```

```
retail$CustomerID <- retail$CustomerID %>% as.character()
```

```
sum(is.na(retail$CustomerID))
```

```
## [1] 135037
```

```
retail <- retail %>% na.omit()
```

```
sum(is.na(retail$CustomerID))
```

```
## [1] 0
```

3 Data Visualization - Total Monthly Transactions

Aim of the visualization was to return a bar chart which shows the distribution of total transactions for each month. in this data set. At this point we have to remark that the month december shows the sum of transactions for both 2010 and 2011.

Firstly, we had to convert the column 'InvoiceDate' into date class with the as.POSIXlt function. The function unclass returns a count for each format "%d-%m-%Y %H:%M". Thus, we could create a visualizaton with ggplot.

In conclusion we can say that during the period from september to december transactions are the highest, especially in november was the peak.

```
test_date <- as.POSIXlt(retail$InvoiceDate, format = "%d-%m-%Y %H:%M")
dates_unclassed <- unclass(test_date)

viz_months <- ggplot(retail, aes(x=dates_unclassed$mon), fill=Quantity) + geom_bar(aes(fill=
Quantity)) + labs(title = 'Sum of Monthly Transactions', subtitle='for both 2010 and 2011', x='Mon
ths January - December', y='Total Transactions') + theme(axis.text.x = element_blank(), axis.tic
ks.x = element_blank()) + scale_y_continuous(labels = format_format(scientific = FALSE, bi
g.mark = ',', decimal.mark = '.'))

print(viz_months)
```

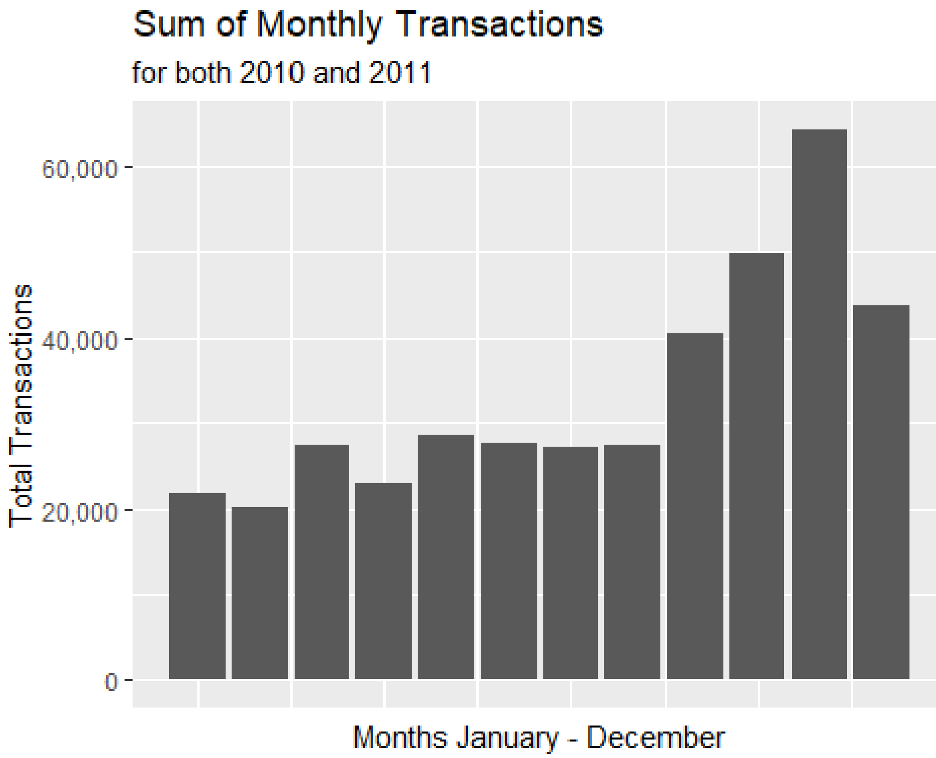

Figure 1: Sum of Monthly Transactions for both 2010 and 2011

4 Preparing Data for RFM Analysis

In this part we prepared the data set for RFM Analysis. Here we need to calculate frequency, recency and monetary.

4.1 Calculating Frequency

To calculate the total transactions the customers' made. In doing so we simply sum the invoice numbers for each customer and group them by the customer id.

```
total_frequency <- retail %>% group_by(CustomerID) %>%
              summarize(count_frequency = n()) %>% arrange(desc(count_frequency))

print(total_frequency)

## # A tibble: 4,372 x 2
##    CustomerID count_frequency
##    <chr>                <int>
##  1 17841                 7812
##  2 14911                 5898
##  3 14096                 5128
##  4 12748                 4459
##  5 14606                 2759
##  6 15311                 2478
```

```
##  7 14646              2085
##  8 13089              1853
##  9 13263              1667
## 10 14298              1640
## # ... with 4,362 more rows
```

4.2 Calculating Monetary

For the monetary value, we will multiply the unitary price by the quantity bought and sum them for each client and eventually group them again by the customer id.

```
total_amount <- retail %>% group_by(CustomerID) %>% summarize(count_total = sum(Quantity*UnitPrice)) %>% arrange(desc(count_total))

print(total_amount)

## # A tibble: 4,372 x 2
##    CustomerID count_total
##    <chr>            <dbl>
##  1 14646          279489.
##  2 18102          256438.
##  3 17450          187322.
##  4 14911          132459.
##  5 12415          123725.
##  6 14156          113215.
##  7 17511           88125.
##  8 16684           65892.
##  9 13694           62691.
## 10 15311           59284.
## # ... with 4,362 more rows
```

4.3 Merging two data frames

In the next step we merged the data frames ‘total frequency’ and ‘total_amount’ by the customer id. We call the new data frame by using head function and see if it was successfully merged.

```
mon_freq_join <- total_frequency %>% inner_join(total_amount)

## Joining, by = "CustomerID"

head(mon_freq_join)

## # A tibble: 6 x 3
##   CustomerID count_frequency count_total
##   <chr>                <int>       <dbl>
## 1 17841                 7812      39869.
## 2 14911                 5898     132459.
## 3 14096                 5128      57121.
## 4 12748                 4459      28406.
## 5 14606                 2759      11633.
## 6 15311                 2478      59284.
```

4.4 Calculating Recency and forming a final rfm-data frame

To calculate recency, we had a look into the invoice dates. Since the date of our last invoice was 09-12-2011, we consider it as the most recent one. Then we subtracted each day from the most recent day to calculate the other recencies for each transaction.

To find out the differences from the latest date to all others we firstly have to convert the column ‘InvoiceDate’ to the class character. The latest date was defined as time_max. The function difftime returns the latest days for all transactions made stored in a new column ‘difference’.

Then we seeked the oldest date from the column difference by using min() function and stored the result into the variable most_recent. To eliminate unnecessary digits we rounded the whole column last_date. Besides, we wanted to remove the word ‘days’ from each row using gsub().

Having a look at the structure of table most_recent, we want to have uniform classes. Hence, we converted the columns of both data frames most_recent and mon_freq_join into the class double.

Finally, we joined the tables mon_freq_join and most_recent by performing an inner join and renamed a columns.

```
retail$InvoiceDate <- as.character(retail$InvoiceDate)

time_max <- as.character('09-12-2011 12:50')

retail$difference <- difftime(strptime(time_max, format = "%d-%m-%Y %H:%M"),
strptime(retail$InvoiceDate, format = "%d-%m-%Y %H:%M"), units='days')


most_recent <- retail %>% group_by(CustomerID) %>% summarize(last_date = min(differenc
e, na.rm = TRUE))

most_recent$last_date <- round(most_recent$last_date, digits = 0)

most_recent <- most_recent %>% mutate(last_date=gsub('\\days','',last_date))

str(most_recent)

## Classes 'tbl_df', 'tbl' and 'data.frame':    4372 obs. of  2 variables:
##  $ CustomerID: chr  "12346" "12347" "12348" "12349" ...
##  $ last_date : chr  "325" "2" "75" "18" ...

most_recent$last_date <- as.double(most_recent$last_date)
most_recent$CustomerID <- as.double(most_recent$CustomerID)

mon_freq_join$CustomerID <- as.double(mon_freq_join$CustomerID)
mon_freq_join$count_frequency <- as.double(mon_freq_join$count_frequency)
```

```r
rfm <- mon_freq_join %>% inner_join(most_recent)

## Joining, by = "CustomerID"

rfm <- rfm %>% rename(transaction_count=count_frequency,total_amount=count_total,recency_days=last_date)
```

5 Clustering with k-means

5.1 Elbow-Method

Initially, we needed to determine the number of cluster. For this we used the elbow-method. It’s called Elbow-Method as the curve looks like an Elbow. Basically, it is an arbitrary pick, but according to the method we should choose the number of clusters where the ‘Elbow’ is shown. In the first step we needed to scale the data so that they have a comparable scale in terms of the clustering process. Therefore, we used the scale function. rfm.scaled is the data we want to cluster. We created the variable xs which implies the range of clusters from where we can choose the number of clusters for the analysis. The function sapply works like a for-loop. This means it picks the numbers from 2:8 and puts them into the function(x). Basically, the numbers picked from 2:8 are iterations and are stored into the variable ys. As a result we can plot xs and ys and choose three clusters for the further analysis.

```r
rfm.scaled <- scale(rfm[, c("transaction_count", "total_amount", 'recency_days')])

xs <- 2:8
ys <- sapply(xs, function(x) {
  model <- kmeans(rfm.scaled, x)
  return(model$tot.withinss)
})

elbow_plot <- qplot(xs, ys, geom = "line") + labs(title = 'Elbow-Method')

print(elbow_plot)
```

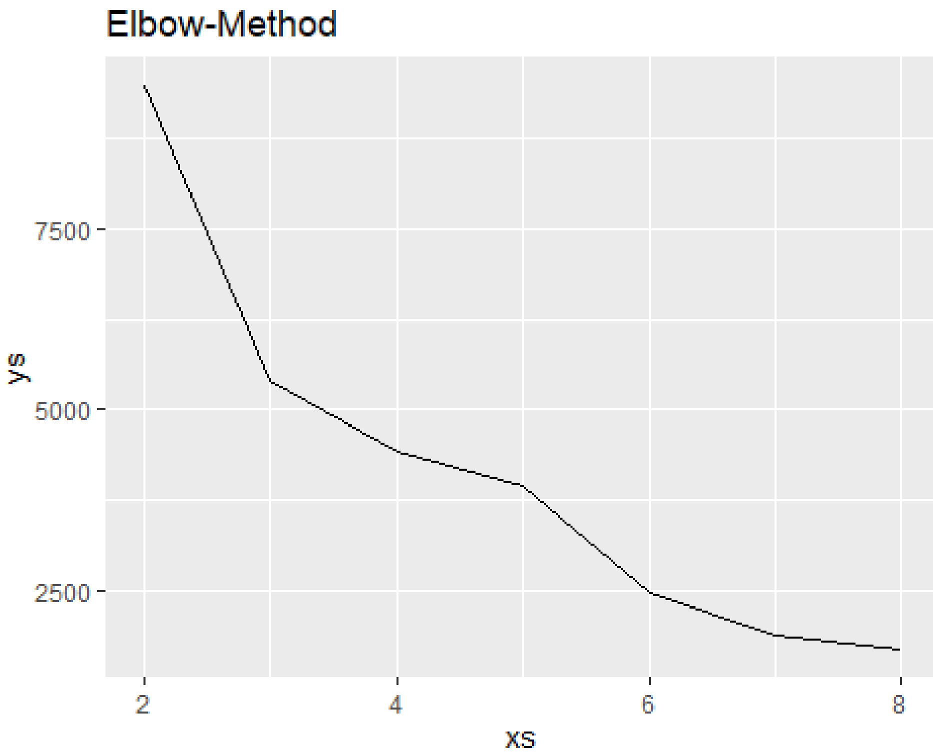

Figure 2: Elbow-Method

5.2 Implementing k-means algorithm

Secondly, we used the k-means algorithm with three clusters as result in the variable cl_model. Then we create the variable centers which we want to include in the next plot to see the centers of the clusters (black dots).

```
cl_model <- kmeans(rfm.scaled, 3, iter.max = 100, nstart = 4, algorithm = "MacQueen")

centers <- data.table(unscale(cl_model$centers, rfm.scaled))


amount_trans <- qplot(rfm$transaction_count, rfm$total_amount,
         color = as.factor(cl_model$cluster),
         xlab = "Total Transactions",
         ylab = "Total Amount") +
  geom_point(aes(x = transaction_count, y = total_amount), color = "black", data = centers) + s
cale_y_continuous(labels = format_format(scientific = FALSE, big.mark = '.', decimal.mark = ',
')) + labs(title='Cluster - Total Amount and Total Transactions', color=' # of Cluster')

print(amount_trans)
```

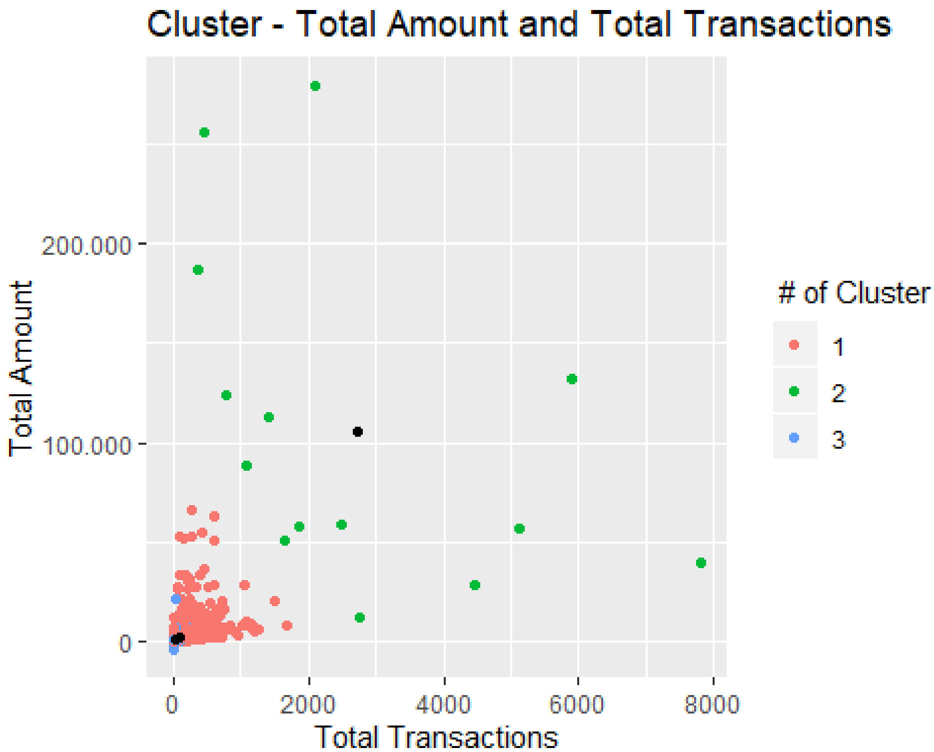

Figure 3: Cluster - Total Amount and Total Transactions

In the plot amount_trans we can see that Cluster #1 shows customers with a high frequency in purchasing and high spendings. Cluster #2 depicts also high spendings and transactions, but only a small group of customers. Cluster #3 indicates less transactions as well as low spendings.

```
amount_recency <- qplot(rfm$recency_days, rfm$total_amount,
         color = as.factor(cl_model$cluster),
         xlab = "Recency Days",
         ylab = "Total Amount") +
  geom_point(aes(x = recency_days, y = total_amount), color = "black", data = centers) + scale
_y_continuous(labels = format_format(scientific = FALSE, big.mark = '.', decimal.mark = ',')) +
labs(title='Cluster - Total Amount and Recency', color='# of Cluster')

print(amount_recency)
```

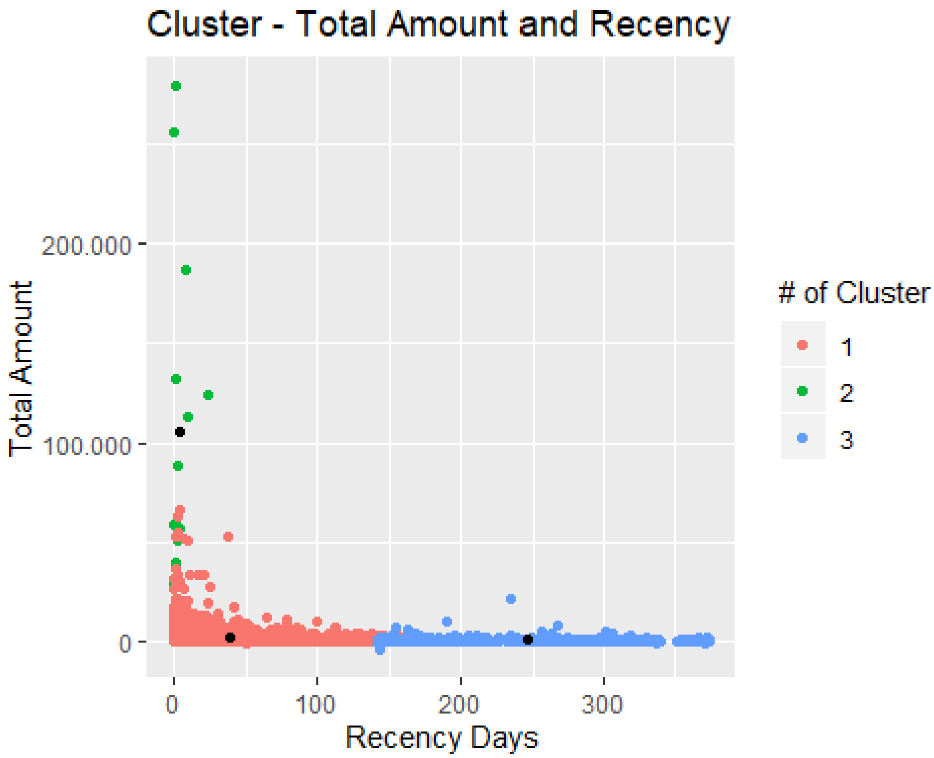

Figure 4: Cluster - Total Amount and Recency

In the plot amount_recency we can see that Cluster #1 shows customers with the latest purchases and high spendings. Cluster #2 depicts higher spendings and low recent purchases, but again only a small group of customers. Cluster #3 indicates that customer haven't purchased recently as well as low spendings.

```
trans_recency <- qplot(rfm$recency_days, rfm$transaction_count,
          color = as.factor(cl_model$cluster),
          xlab = "Recency Days",
          ylab = "Total Transactions") +
  geom_point(aes(x = recency_days, y = transaction_count), color = "black", data = centers) + l
abs(title='Cluster - Total Transactions and Recency', color='# of Cluster')

print(trans_recency)
```

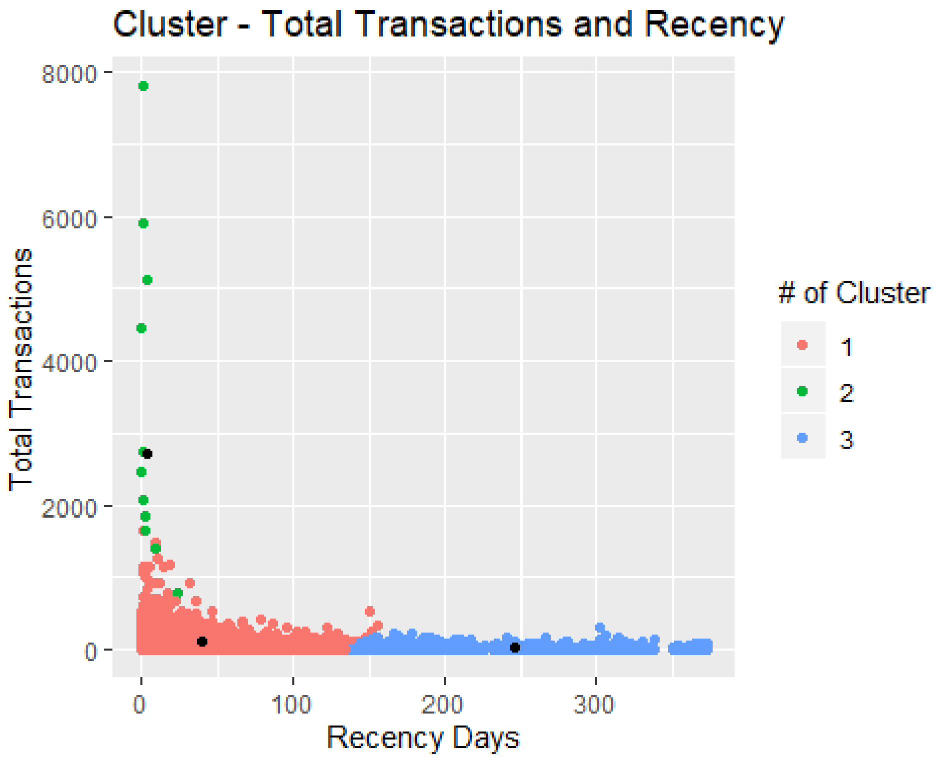

Figure 5: Cluster - Total Transactions and Recency

In the plot amount_trans we can see that Cluster #1 shows customers with a high frequency in purchasing and latest purchases. Cluster #2 depicts high amount of transactions, and a small group of customers that purchased recently. Cluster #3 indicates a less transactions and the purchases where about 140 days ago.

6 Conclusion

Customers in cluster #1 is the segment we need to target in the future as they were a large group of recent buyers that purchased in high quantity and built the highest revenues. Cluster #2 can be considered in marketing activities, however, it is a small group. Besides, we can conclude that we can neglect the customers from cluster #3 as they haven't purchased products recently, purchased in low quantity as well as haven't generated turnover.

7 References


```
citation('data.table')

##   Matt Dowle and Arun Srinivasan (2019). data.table: Extension of
##   `data.frame`. R package version 1.12.2.
##   https://CRAN.R-project.org/package=data.table
citation('ggplot2')
```

```
## H. Wickham. ggplot2: Elegant Graphics for Data Analysis.
## Springer-Verlag New York, 2016.
citation('dplyr')

## Hadley Wickham, Romain François, Lionel Henry and Kirill Müller
## (2019). dplyr: A Grammar of Data Manipulation. R package version
## 0.8.0.1. https://CRAN.R-project.org/package=dplyr
citation('lubridate')

## Garrett Grolemund, Hadley Wickham (2011). Dates and Times Made Easy
## with lubridate. Journal of Statistical Software, 40(3), 1-25. URL
## http://www.jstatsoft.org/v40/i03/.
citation('rlang')

## Lionel Henry and Hadley Wickham (2019). rlang: Functions for Base
## Types and Core R and 'Tidyverse' Features. R package version 0.3.4.
## https://CRAN.R-project.org/package=rlang
citation('data.table')

## Matt Dowle and Arun Srinivasan (2019). data.table: Extension of
## `data.frame`. R package version 1.12.2.
## https://CRAN.R-project.org/package=data.table
citation('rfm')

## Aravind Hebbali (2019). rfm: Recency, Frequency and Monetary Value
## Analysis. R package version 0.2.0.
## https://CRAN.R-project.org/package=rfm
citation('DMwR')

## Torgo, L. (2010). Data Mining with R, learning with case studies
## Chapman and Hall/CRC. URL:
## http://www.dcc.fc.up.pt/~ltorgo/DataMiningWithR
citation('scales')

## Hadley Wickham (2018). scales: Scale Functions for Visualization. R
## package version 1.0.0. https://CRAN.R-project.org/package=scales
```

https://www.kaggle.com/hellbuoy/online-retail-customer-clustering

https://www.udemy.com/course/machine-learning-komplett/ (by Jannis Seemann)