RFM and Clustering on Retail data using R

* By Aditya Jakkam

# Introduction

Data was obtained from online website data repository for the Recency, Frequency and Monetization analysis on the retail industry market.

* This data was used mainly for two purposes
* 1. To rank the customers based on their transactions history
* 2. Cluster them into three categories: a. leaving customers, b. left customers and c. good customers

About Data

Data has more than 560k observations with the following variables:

1. InvoiceNo – Transaction number of the event
2. StockCode – Inventory number for the product
3. Description – Name of the product
4. Quantity – Quantity of the product bought
5. InvoiceDate – Transaction date and time
6. UnitPrice – Price of the product
7. CustomerID – Unique ID of the customer
8. Country – Country where the product was bought

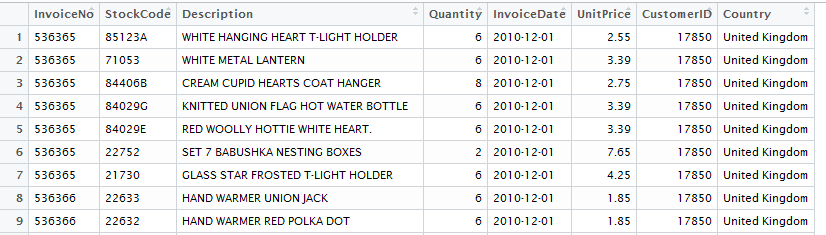


Figure 1. Data which was extracted from online

Data Cleaning

Since the data has lot of missing values in the customerid section, it was removed as it doesn’t make much sense for the analysis. There are more than 160k observations which doesn’t have customerid. It was removed because of our analysis.

For our analysis, 4 variables are taken into consideration for the project. They are customerid, invoicedate, Unitprice and quantity.

These are transitioned with the new name types as following:

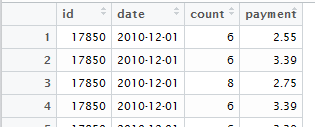


Figure 2. After cleaning data with required variables

Data was combined based on the customerid and date for the best analysis based on the customers. After the combinations data has unique customerid for date.

Code was given in the appendix for the clarification.

RFM Analysis and Clustering

Functions was used to get the recency based transaction date, frequency based on the quantity and monetization based on the money spent by the customer in buying different products.

This data was further divided into three different clusters based on the ranks given to each customer. This will be used by the marketing department in deciding about the different promotional activities.

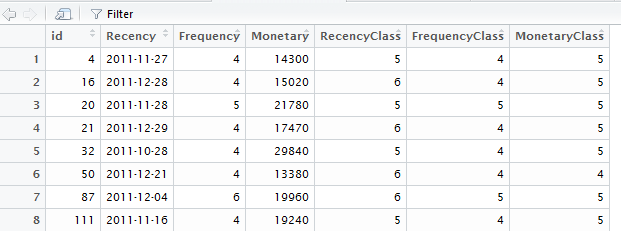


Figure 3. Data set for the good standing customers

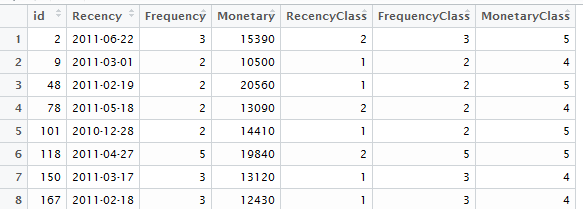


Figure 4. Data set for the left customers

Result

Result was extracted from R so that ID’s of customers with different clusters. This will be used for the email promotions and coupon promotions.

For the good customers, it is always best to promote and give coupons as they are reliable and we can except purchases from them. We can upsell here for more revenue to the company.

For the leaving customers, it is good if we can attract them to purchase more. We can upsell and cross sell here so that customers will make some more purchases and increase the revenue to the company.

For the left customers, it is better to promote less as we expect less purchases from them. This way we can decrease the operational costs for the company.

Appendix

**R Code which was used:**

#libraries used for the analysis

library(readxl)

library(magrittr)

library(dplyr)

library(plyr)

library(Hmisc)

library(histogram)

library(ggplot2)

library(scales)

library(grid)

library(RColorBrewer)

library(foreign)

#reading the file

Online\_Retail <- read\_excel("D:/projects/retail/Online Retail.xlsx")

View(Online\_Retail)

#Converting charater format to Date for "InvoiceDate" variable

Online\_Retail$InvoiceDate = as.Date(Online\_Retail$InvoiceDate, format('%Y-%m-%d'))

View(Online\_Retail)

#Removing the unwanted observations - observations without customerID

str(Online\_Retail)

complete.cases(Online\_Retail)

Online\_Retail <- Online\_Retail[complete.cases(Online\_Retail), ]

str(Online\_Retail)

Online\_Retail <- na.omit(Online\_Retail)

View(Online\_Retail)

#Combining the changes and creating a new data set

Retail <- data.frame((cbind(Online\_Retail$CustomerID, Online\_Retail["InvoiceDate"], Online\_Retail$Quantity, Online\_Retail$UnitPrice)))

View(Retail)

#Adding column names to the data set

names <- c("id","date","count","payment")

names(Retail) <- names

View(Retail)

#Partial output of the data and ther number of observations

head(Retail)

dim(Retail)

#Adding the prices, frequency for the same ID and date observations

ddply(Retail,.(id,date),summarize,fre=sum(count),payment=sum(payment))

#Removing the duplicacy of the data

uid <- Retail[!duplicated(Retail[,"id"]),]

dim(uid)

# RFM amalysis function

#'result <- rfm\_auto(data)

data <- Retail

rfm\_auto <- function(data, id="id", payment="payment", date="date",

breaks=c(r=5, f=5, m=5), date\_format,

to\_text=" to ", exact=FALSE, tz=Sys.timezone()) {

if(is.list(breaks)) {

breaks <- c(r=breaks[["r"]], f=breaks[["f"]], m=breaks[["m"]])

} else if(is.vector(breaks) && is.numeric(breaks)) {

if(length(breaks) == 1) {

breaks <- c(r=breaks, f=breaks, m=breaks)

} else {

breaks <- c(r=unname(breaks["r"]), f=unname(breaks["f"]), m=unname(breaks["m"]))

}

}

if(length(breaks) != 3) stop()

is.Date <- function(x) is(x, "Date")

is.POSIXlt <- function(x) is(x, "POSIXlt")

data <- data.frame(data)

if(is.factor(data[,date]))

data[,date] <- as.character(data[,date])

if(!missing(date\_format) && is.character(data[,date]))

data[,date] <- strptime(data[,date], format = date\_format, tz = tz) %>% as.POSIXct

if(is.Date(data[,date]))

data[,date] <- as.character(data[,date])

if(is.character(data[,date]) || is.POSIXlt(data[,date]))

data[,date] <- as.POSIXct(data[,date], tz = tz)

dots <- list(sprintf("max(%s)", date), ~n(), sprintf("sum(%s)", payment))

rfm <- data %>%

group\_by\_(.dots = id) %>%

summarise\_(.dots = dots %>% setNames(c("Recency", "Frequency", "Monetary")))

r\_breaks <- rfm\_compute\_breaks(rfm$Recency, breaks["r"])

f\_breaks <- rfm\_compute\_breaks(rfm$Frequency, breaks["f"])

m\_breaks <- rfm\_compute\_breaks(rfm$Monetary, breaks["m"])

max\_date <- max(r\_breaks)

if(!exact) {

r\_breaks\_date <- r\_breaks %>% as.Date(tz=tz)

r\_breaks\_date <- c(r\_breaks\_date[1], Map(function(d) d+1, r\_breaks\_date[-1]) %>% unlist)

r\_breaks <- r\_breaks\_date %>% as.character %>% as.POSIXct(tz=tz)

f\_breaks <- rfm\_pretty\_breaks(f\_breaks)

m\_breaks <- rfm\_pretty\_breaks(m\_breaks)

}

lower\_breaks <- function(breaks) {

c(breaks[1], breaks[-c(1,length(breaks))] + 1)

}

upper\_breaks <- function(breaks) {

c(breaks[1], breaks[-c(1,length(breaks))] - 1)

}

r\_class <- paste(lower\_breaks(r\_breaks), r\_breaks[-1], sep=to\_text)

f\_class <- Map(function(upper, count) {

ifelse(count == 1, upper, paste(upper - count + 1, upper, sep=to\_text))

}, f\_breaks[-1], diff(f\_breaks)) %>% unlist

m\_class <- paste(lower\_breaks(m\_breaks), m\_breaks[-1], sep=to\_text)

r <- cut(rfm$Recency, r\_breaks, include.lowest=TRUE) %>% as.numeric

f <- cut(rfm$Frequency, f\_breaks, include.lowest=TRUE) %>% as.numeric

m <- cut(rfm$Monetary, m\_breaks, include.lowest=TRUE) %>% as.numeric

rfm <- rfm %>% mutate(RecencyClass=r, FrequencyClass=f, MonetaryClass=m) %>%

data.frame

r\_breaks\_date <- as.Date(r\_breaks, tz=tz)

r\_breaks\_days <- difftime(max(r\_breaks\_date), r\_breaks\_date, units="days")

r\_class\_days <- paste(upper\_breaks(r\_breaks\_days), r\_breaks\_days[-1], sep=to\_text)

rf\_table <- table(Recency=r, Frequency=f)

fr\_table <- table(Frequency=f, Recency=r)

fm\_table <- table(Frequency=f, Monetary=m)

mf\_table <- table(Monetary=m, Frequency=f)

mr\_table <- table(Monetary=m, Recency=r)

rm\_table <- table(Recency=r, Monetary=m)

get\_table <- function(type=c("RF", "FR", "FM", "MF", "MR", "RM"),

R\_slice, F\_slice, M\_slice) {

type <- match.arg(type)

d <- get\_sliced\_rfm(R\_slice=R\_slice, F\_slice=F\_slice, M\_slice=M\_slice)

r <- cut(d$Recency, r\_breaks, include.lowest=TRUE) %>% as.numeric

f <- cut(d$Frequency, f\_breaks, include.lowest=TRUE) %>% as.numeric

m <- cut(d$Monetary, m\_breaks, include.lowest=TRUE) %>% as.numeric

tbl <- switch(type,

"RF"=table(Recency=r, Frequency=f),

"FR"=table(Frequency=f, Recency=r),

"FM"=table(Frequency=f, Monetary=m),

"MF"=table(Monetary=m, Frequency=f),

"MR"=table(Monetary=m, Recency=r),

"RM"=table(Recency=r, Monetary=m))

row\_names <- switch(type,

"RF"=r\_class\_days, "FR"=f\_class,

"FM"=f\_class, "MF"=m\_class,

"MR"=m\_class, "RM"=r\_class\_days)

col\_names <- switch(type,

"RF"=f\_class, "FR"=r\_class\_days,

"FM"=m\_class, "MF"=f\_class,

"MR"=r\_class\_days, "RM"=m\_class)

dummy\_table <- switch(type,

"RF"=rf\_table, "FR"=fr\_table,

"FM"=fm\_table, "MF"=mf\_table,

"MR"=mr\_table, "RM"=rm\_table)

dummy\_table[] <- 0

for(row\_name in rownames(tbl)) {

for(col\_name in colnames(tbl)) {

dummy\_table[row\_name, col\_name] <- tbl[row\_name, col\_name]

}

}

tbl <- dummy\_table

rownames(tbl) <- row\_names

colnames(tbl) <- col\_names

tbl

}

get\_sliced\_rfm <- function(R\_slice, F\_slice, M\_slice) {

d <- rfm

if(!missing(R\_slice)) {

d <- d %>% filter(RecencyClass %in% R\_slice)

}

if(!missing(F\_slice)) {

d <- d %>% filter(FrequencyClass %in% F\_slice)

}

if(!missing(M\_slice)) {

d <- d %>% filter(MonetaryClass %in% M\_slice)

}

d

}

list(rfm=rfm,

breaks=list(recency\_breaks=r\_breaks, recency\_breaks\_days=r\_breaks\_days,

frequency\_breaks=f\_breaks, monetary\_breaks=m\_breaks),

classes=list(recency\_class=r\_class, recency\_class\_days=r\_class\_days,

frequency\_class=f\_class, monetary\_class=m\_class),

get\_table=get\_table, get\_sliced\_rfm=get\_sliced\_rfm)

}

#'Computing breaks

rfm\_compute\_breaks <- function(values, break\_num=5) {

breaks <- quantile(values, probs = seq(0, 1, length.out = break\_num + 1))

if(break\_num == 2 && is.integer(values)) {

min\_value <- min(values)

max\_value <- max(values)

for(i in min\_value:(max\_value-1)) {

breaks <- c(min\_value-1, i, max\_value)

tbl <- table(cut(values, breaks))

if(tbl[1] > tbl[2]) break

}

return(breaks)

}

breaks <- unname(breaks)

if(length(breaks) == length(unique(breaks))) return(breaks)

min\_value <- min(values)

next\_values <- Filter(function(x) x != min\_value, values)

next\_breaks <- rfm\_compute\_breaks(next\_values, break\_num = break\_num - 1)

min\_value <- max(1, min\_value)

unname(c(min\_value - 1, next\_breaks))

}

#'Pretty breaks

rfm\_pretty\_breaks <- function(breaks) {

digits <- floor(log10(breaks))

head <- ifelse(digits[1] <= 0, floor(breaks[1]), {

base <- 10 ^ (digits[1] - 1)

floor(breaks[1] / base) \* base

})

tail <- Map(function(break\_, digit) {

if(digit <= 0) {

ceiling(break\_)

} else {

base <- 10 ^ (digit - 1)

ceiling(break\_ / base) \* base

}

}, breaks[-1], digits[-1]) %>% unlist

c(head, tail)

}

#'Generate sample data for RFM analysis

rfm\_generate\_data <- function(id\_num=1000,

date\_type=c("char", "Date", "POSIXct", "POSIXlt"),

seed, ...) {

date\_type <- match.arg(date\_type)

if(!missing(seed)) set.seed(seed)

data <- data\_frame(id=seq\_len(id\_num), count=rpois(id\_num, lambda = 2))

ids <- apply(data, 1, function(x) rep(x[["id"]], x[["count"]])) %>% unlist

n <- length(ids)

payment <- round(rgamma(n, shape = 2, scale = 2) \* 100) \* 10

if(date\_type == "char") {

date <- rfm\_rdate(n, ...) %>% as.character

} else if(date\_type == "Date") {

date <- rfm\_rdate(n, ...)

} else if(date\_type == "POSIXlt"){

date <- rfm\_rdatetime(n, ...) %>% as.POSIXlt

} else if(date\_type == "POSIXct") {

date <- rfm\_rdatetime(n, ...)

}

data\_frame(id=ids, payment=payment, date=date) %>% data.frame

}

#Generating random Date sequence

rfm\_rdate <- function(n, begin=as.Date("2010-12-01"), end=as.Date("2011-12-31"),

by="days", tz=Sys.timezone()) {

if(is.character(begin)) begin <- as.Date(begin, tz=tz)

if(is.character(end)) end <- as.Date(end, tz=tz)

date\_seq <- seq(begin, end - 1, by=by)

sample(date\_seq, size = n, replace = TRUE)

}

#Generating random dateteime sequence

rfm\_rdatetime <- function(n, begin=as.POSIXct("2014-12-01", tz=Sys.timezone()),

end=as.POSIXct("2015-01-01", tz=Sys.timezone())) {

datetime\_seq <- seq(begin, end, by="sec")

datetime\_seq <- datetime\_seq[-length(datetime\_seq)]

sample(datetime\_seq, size = n, replace = TRUE)

}

#Output of RFM\_Auto funcion

Retail <- rfm\_auto(data)

head(Retail$rfm)

#Applying Breaks and Classes for further analysis

Retail$breaks

Retail$classes

#Analysing the output for Recency and Frequency

Retail$get\_table("RF", M\_slice=4:5)

Retail$get\_table("RF")

#Assigning the name to the data set

data <- rfm\_generate\_data()

head(data)

Retail <- rfm\_auto(data)

head(data)

data <- rfm\_generate\_data(10000, begin="2010-12-01", end = "2011-12-31", seed = 123)

Retail <- rfm\_auto(data, breaks=list(r=6, f=5, m=5))

Retail$get\_table ("RF", M\_slice=4:5)

#Clustering the customers based on the RFM values into three categories left\_customers, leaving\_customers and good\_customers

left\_customers <- Retail$get\_sliced\_rfm(R\_slice=1:2, F\_slice=2:5, M\_slice=4:5)

leaving\_customers <- Retail$get\_sliced\_rfm(R\_slice = 3:4, F\_slice = 4:5, M\_slice = 4:5)

good\_customers <- Retail$get\_sliced\_rfm(R\_slice = 5:6, F\_slice = 4:5, M\_slice = 4:5)

#Output of the clusters

View(left\_customers)

View(leaving\_customers)

View(good\_customers)

#Extracting the data into text format for marketing and promotions purposes

write.table(left\_customers, "D:/left\_customers.txt", sep = "\t")

write.table(leaving\_customers, "D:/leaving\_customers.txt", sep= "\t")

write.table(good\_customers, "D:/good\_customers.txt", sep= "\t")