Recency, Frequency and Monetary Value

Yufeng Huang

Associate Professor of Marketing, Simon Business School

August 18, 2022

RFM targeting

- You already know the meaning of segmentation from your econ/marketing class
- RFM is a classical way to segment the customers and target the valuable sub-groups
  - internal logic: human behavior is persistent and therefore past choices of a consumer can predict future choices
  - not the ideal targeting approach but this is something easy to compute
- Of course, my real purpose is for you to practice for-loops

Prep-steps

```
# ====== CLEAR EVERYTHING ======
rm(list = ls())

# ====== READ TRIAL DATA =======
url <- 'https://dl.dropboxusercontent.com/s/xxfloksp0968mgu/CDNOW_sample.txt'
if (!file.exists('CDNOW_sample.txt')) download.file(url, 'CDNOW_sample.txt')
df.raw <- read.fwf('CDNOW_sample.txt', width = c(6, 5, 9, 3, 8), stringsAsFactors = F)  # load data

# a) generate year and month (using date functions, but you can use all sorts of methods)
df.raw[[1]] <- NULL     # drop old id
names(df.raw) <- c("id", "date", "qty", "expd")
df.raw$date <- as.Date(as.character(df.raw$date), "%Y%m%d")
df.raw$year <- as.numeric(format(df.raw$date, '%Y'))
df.raw$month <- as.numeric(format(df.raw$date, '%m'))

# b) aggregate into monthly data with number of trips and total expenditure
df.raw$trips <- 1
df.month <- aggregate(cbind(qty, expd, trips) ~ id + year + month, data = df.raw, FUN = sum)
#    note 1: cbind() ~ ... will aggregate using the same function, but on multiple variables

# c) generate a table of year-months, merge, replace no trip to zero.
id.year.month <- expand.grid(list(id = 1:max(df.raw$id),
        year = unique(df.raw$year), month = unique(df.raw$month)))
id.year.month$t <- (id.year.month$year - 1997)*12 + id.year.month$month
df <- merge(id.year.month, df.month, by = c("id", "year", "month"), all = TRUE)
df <- df[df$year == 1997 | df$month <= 6, ]      # subset, get rid of 199807-199812
df <- df[order(df$id, df$year, df$month), ]      # sort
df[is.na(df$qty), c("qty", "expd", "trips")] <- 0        # replace no-trip months to zero
```

Recency (1)

```
# Method 1
# let's keep track of time
t0 <- proc.time()       # log current system time

# initialize recency
df$recency <- NA

# double for-loop, on individual and then year-month
#    note: indentation for readability
for (i in 1:max(df$id)) {
        df$recency[df$id == i & df$t == 1] <- NA
        for (m in 2:18) {
                if (df$qty[df$id == i & df$t == m - 1] > 0) {
                        df$recency[df$id == i & df$t == m] <- 1
                } else {
                        df$recency[df$id == i & df$t == m] <- df$recency[df$id == i & df$t == m - 1] + 1
                }
        }
}

t1 <- proc.time()

# confirm that it works
head(df, 4)

##   id year month t qty  expd trips recency
## 1  1 1997     1 1   4 59.06     2      NA
## 2  1 1997     2 2   0  0.00     0       1
## 3  1 1997     3 3   0  0.00     0       2
## 4  1 1997     4 4   0  0.00     0       3

# elapsed time
t1 - t0

##    user  system elapsed
##    4.86    2.00    6.86
```

Thoughts?

- Some of you did year and then month as two separate loop (so in total 3 layers of loops)
  - at first glance straightforward, because the variables you have are “year” and “month” separate
  - however, does not add value to the *structure*, because the logic of the algorithm should be
    - “continue to the next month, for the same individual”, rather than
    - “continue to the next month, for the same individual in the given year”
  - therefore, it is much easier to construct an additional variable, here $t
- Still, we have two layers of loops, and they take quite some time
  - any ways to improve the code?

Thoughts?

- Note that the only reason we need individual i is that we try to treat the first period separately
  - if it is the first month for a given i, we reset value for recency to NA
  - for second month onwards, we let recency continue from its last value
- Could write the code into a single-layer loop
  - now, iterator is "an observation" rather than "an individual"
  - at each step, check whether we are now looking at a new individual or not
  - if yes, reset recency to NA
  - if no, let recency continue from its last value

```r
# Method 2
t0 <- proc.time()

# sort df (should be sorted, just to make sure)
df <- df[order(df$id, df$t), ]

# initialize recency
df$recency <- NA

# single for-loop, on observation number (note that we start from 2)
for (obs in 2:nrow(df)) {
    if (df$id[obs] != df$id[obs - 1]) { # check whether id switched
            df$recency[obs] <- NA          # if id switched, assign NA to current obs of data
        } else {
            if (df$qty[obs - 1] > 0) {
                df$recency[obs] <- 1    # if bought last month, reset recency to 1
            } else {
                df$recency[obs] <- df$recency[obs - 1] + 1      # if not, advance recency by 1
            }
        }
}

t1 <- proc.time()

# works exactly the same way
head(df, 4)

##   id year month t qty  expd trips recency
## 1  1 1997     1 1   4 59.06     2      NA
## 2  1 1997     2 2   0  0.00     0       1
## 3  1 1997     3 3   0  0.00     0       2
## 4  1 1997     4 4   0  0.00     0       3

# 15-20 times faster (!!!) because we only have one-layer of loop
t1 - t0

##    user  system elapsed
##    0.27    0.09    0.36
```

Frequency (1)

```
# One might think about writing a loop for frequency
#    but note that frequency is just the last quarter's total number of trips
#    now we realize that we do not need loops for this

# construct quarter
df$quarter <- (df$t - 1) %/% 3 + 1  # 6 quarters in total

# aggregate to get frequency
frequency <- aggregate(trips ~ id + quarter, df, sum)   # number of trips in a quarter
frequency$quarter <- frequency$quarter + 1    # so it later merges with next quarter in df
colnames(frequency)[3] <- "frequency"

# merge back to data
df <- merge(df, frequency, by = c("id", "quarter"), all.x = TRUE)

# observe
head(df, 4)

##   id quarter year month t qty  expd trips recency frequency
## 1  1       1 1997     1 1   4 59.06     2      NA        NA
## 2  1       1 1997     2 2   0  0.00     0       1        NA
## 3  1       1 1997     3 3   0  0.00     0       2        NA
## 4  1       2 1997     4 4   0  0.00     0       3         2
```

Frequency (2)

```r
# We can of course get it from a loop

# initialize frequency
df$frequency <- NA

# for-loop to get frequency
for (i in 1:max(df$id)) {
        for (q in 2:6) {
                df$frequency[df$id == i & df$quarter == q] <-
                        sum(df$trips[df$id == i & df$quarter == q - 1])
        }
}

# observe
head(df, 4)

##   id quarter year month t qty  expd trips recency frequency
## 1  1       1 1997     1 1   4 59.06     2      NA        NA
## 2  1       1 1997     2 2   0  0.00     0       1        NA
## 3  1       1 1997     3 3   0  0.00     0       2        NA
## 4  1       2 1997     4 4   0  0.00     0       3         2
```

Monetary value (1)

```
# Idea: average non-zero monthly expenditure in the past
# so we need: 1) average of nonzero months
#             2) do so by different sets of time periods (first to previous month for each id)

# Method 1
t0 <- proc.time()
df$monvalue <- NA
df <- df[order(df$id, df$t), ]

# directly loop over rows
for (obs in 1:nrow(df)) {
        if (df$t[obs] == 1) {   # if this is initial observation
                expd.temp <- numeric(0) # initialize an empty vector if no history yet
        } else {
                expd.temp <- c(expd.temp, df$expd[obs - 1])     # expand vector of expenditure
                expd.temp[expd.temp == 0] <- NA # replace zeros into NAs
                df$monvalue[obs] <- mean(expd.temp, na.rm = T)  # take mean
        }
}

t1 <- proc.time()

# check result
head(df, 4)

##   id quarter year month t qty  expd trips recency frequency monvalue
## 1  1       1 1997     1 1   4 59.06     2      NA        NA       NA
## 2  1       1 1997     2 2   0  0.00     0       1        NA    59.06
## 3  1       1 1997     3 3   0  0.00     0       2        NA    59.06
## 4  1       2 1997     4 4   0  0.00     0       3         2    59.06

t1 - t0

##    user  system elapsed
##    0.31    0.06    0.37
```

Monetary value (2)

```
# Idea: we can just keep track of scalar values
#    realizing that average monthly expenditure is total expd / number of months

# Variant 1: avoid vector expansion
t0 <- proc.time()
df$monvalue <- NA
df <- df[order(df$id, df$t), ]

# this time, keep sum and total number as two scalar variables
for (obs in 1:nrow(df)) {
        if (df$t[obs] == 1) {
                sum.expd <- 0                  # initialize their value at the start of each individual
                total.months <- 0
        } else {
                if (df$trips[obs - 1] > 0) {    # if there are some trips
                        sum.expd <- sum.expd + df$expd[obs - 1]
                        total.months <- total.months + 1
                }
                df$monvalue[obs] <- sum.expd / total.months
        }
}

t1 <- proc.time()

# takes roughly 10-20% less time because we don't have vector expansion
t1 - t0

##    user  system elapsed
##    0.27    0.00    0.26
```

Monetary value (3)

```
# Idea: can use an index to keep track of the initial obs for each individual

# Variant 2: avoid defining a local vector
t0 <- proc.time()
df$monvalue <- NA
df <- df[order(df$id, df$t), ]

# turn zero expenditure into NA
df$expd[df$expd == 0] <- NA

# this time, subset "on the fly" and make use of na.rm = T
for (obs in 1:nrow(df)) {
        if (df$t[obs] == 1) {
                init <- obs     # initial observation for the given individual
        } else {
                df$monvalue[obs] <- mean(df$expd[init:(obs - 1)], na.rm = T)    # subset, but do not copy
        }
}

t1 <- proc.time()

df$expd[is.na(df$expd)] <- 0    # return the zeros

# no time saving, presumably because mean() sums up a range of values
#    and there're wasted calculations across time periods
t1 - t0

##    user  system elapsed
##    0.34    0.03    0.38
```

Thoughts?

- Looping over observation index rather than looping over individual-time (worse is individual-year-month in 3 layers) again saves a lot of time
- In addition, in the baseline method here, some time is wasted when we dynamically expand the vector expd.temp
  - does not help when we subset and take averages because of wasteful calculations in the mean(vector)
  - efficiency can be improved by observing that mean = sum / nr. months, and keeping track of scalar sums and number is more efficient

Finally, RFM index

```
# compute RFM index
df$index <- -0.1*df$recency + 3.5*df$frequency + 0.2*df$monvalue

# split into quantiles
#    note: cut() itself will generate a factor, in this case happened to be nicely ordered
#    note: perfectly fine to use a for-loop for this, back to this in the apply lecture notes
df$quantiles <- as.numeric(cut(df$index, quantile(df$index, seq(0, 1, 0.1), na.rm = T)))
# show parts of data
df[1:18, c(1, 5, 7, 8, 9, 10, 11, 12, 13)]

##    id  t  expd trips recency frequency monvalue  index quantiles
## 1   1  1 59.06     2      NA        NA       NA     NA        NA
## 2   1  2  0.00     0       1        NA    59.06     NA        NA
## 3   1  3  0.00     0       2        NA    59.06     NA        NA
## 4   1  4  0.00     0       3         2    59.06 18.512         9
## 5   1  5  0.00     0       4         2    59.06 18.412         9
## 6   1  6  0.00     0       5         2    59.06 18.312         9
## 7   1  7  0.00     0       6         0    59.06 11.212         8
## 9   1  8 14.96     1       7         0    59.06 11.112         8
## 8   1  9  0.00     0       1         0    37.01  7.302         6
## 10  1 10  0.00     0       2         1    37.01 10.702         8
## 11  1 11  0.00     0       3         1    37.01 10.602         8
## 12  1 12 26.48     1       4         1    37.01 10.502         8
## 14  1 13  0.00     0       1         1    33.50 10.100         8
## 15  1 14  0.00     0       2         1    33.50 10.000         8
## 13  1 15  0.00     0       3         1    33.50  9.900         8
## 17  1 16  0.00     0       4         0    33.50  6.300         6
## 16  1 17  0.00     0       5         0    33.50  6.200         6
## 18  1 18  0.00     0       6         0    33.50  6.100         5
```

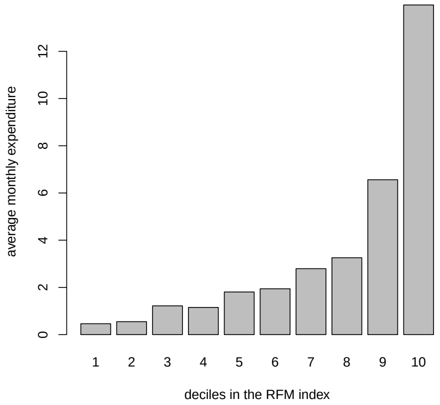
Alternatives to the cut() function?

```
# can write a loop over the 10 possible groups
qtile_cutoffs <- quantile(df$index, seq(0.1, 1, 0.1), na.rm = T)

df$quantiles <- 1
for (i in 2:10) {
        df$quantiles <- ifelse(df$index > qtile_cutoffs[i-1] &
                                df$index <= qtile_cutoffs[i], # test: if between two neighboring cutoffs

                        i,      # yes: take group i
                        df$quantiles)   # no: unchanged
}
```

```r
# per-month expenditure
expd.by.group <- aggregate(expd ~ quantiles, df, mean)
# bar plot
barplot(expd.by.group[, 2], names.arg = 1:10,
        ylab = "average monthly expenditure", xlab = "deciles in the RFM index")
```

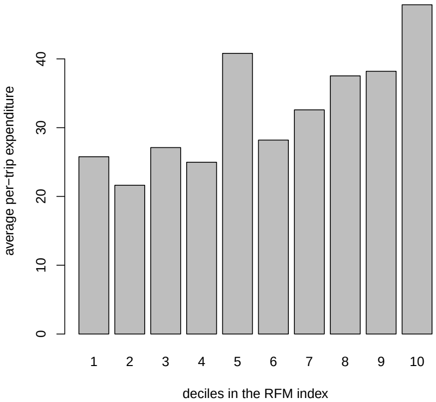


Is RFM useful at targeting consumers?

- Recall that we only use historical data to generate RFM
  - that means, no guarantee that high-RFM consumers will buy today
  - but we see that they do spend dramatically more
- Therefore, yes, RFM is useful at targeting consumers
  - why? Consumer behavior is persistent because different people have different preference/habit/purchase patterns
  - and we should target from top to bottom deciles depending on the marketing cost

Anyone tried to plot per-trip expenditure?

```
# per-TRIP expenditure (for existing trips)
trip.expd.by.group <- aggregate(expd/trips ~ quantiles, df, mean)
# bar plot
barplot(trip.expd.by.group[, 2], names.arg = 1:10,
        ylab = "average per-trip expenditure", xlab = "deciles in the RFM index")
```



Why are there such differences?

- Conditional on coming to the store, expenditure is much higher
  - (mechanically) because many individual-months have zero expenditure
- Do RFM predict expenditure amount, given that they are positive?
  - well, yes, but to a lesser extent compared to predicting *whether* expenditure is non-zero
- I.e., the index we constructed is good at predicting who comes when, but not how much they spend given that they come
  - these are both good marketing insights and should be examined separately