Foundations of Marketing Analytics: Module 1: Statistical Segmentation

Stefan Avey

2017-01-27

Contents

1 Load the Data

```
###############
## Load data ##
###############
purchases <- read.delim("data/purchases.txt", header = FALSE)

###################
## Preprocessing ##
###################
## Add column names
colnames(purchases) <- c("customer_id", "purchase_amount", "date_of_purchase")

## Convert date and add column for purchase year
purchases <- purchases %>%
    mutate(date_of_purchase = ymd(date_of_purchase)) %>%
    mutate(year_of_purchase = year(date_of_purchase))

## Look at the data
head(purchases)
```

customer_id	purchase_amount	date_of_purchase	year_of_purchase
860	50	2012-09-28	2012
1200	100	2005-10-25	2005
1420	50	2009-07-09	2009
1940	70	2013-01-25	2013
1960	40	2013-10-29	2013
2620	30	2006-03-09	2006

```r
summary(purchases)
```

```
##   customer_id     purchase_amount   date_of_purchase     year_of_purchase
##  Min.   :    10   Min.   :   5.00   Min.   :2005-01-02   Min.   :2005
##  1st Qu.: 57722   1st Qu.:  25.00   1st Qu.:2009-01-17   1st Qu.:2009
##  Median :102440   Median :  30.00   Median :2011-11-23   Median :2011
##  Mean   :108937   Mean   :  62.34   Mean   :2011-07-14   Mean   :2011
##  3rd Qu.:160528   3rd Qu.:  60.00   3rd Qu.:2013-12-29   3rd Qu.:2013
##  Max.   :264200   Max.   :4500.00   Max.   :2015-12-31   Max.   :2015
```

2 Compute Recency, Frequency, and Monetary Value

Three common characteristics of customers readily available from a transactional database are recency, frequency, and monetary value.

1. *Recency*: Time since last purchase.
2. *Frequency*: Number of purchases made in the past.
3. *Monetary value*: Amount spent at each purchase occasion.

Here we use these 3 simple characteristics to divide customers into actionable segments.

```r
## Add columns for recency, frequency, and monetary value
startDate <- as.Date("2016-01-01")
customers_2015 <- purchases %>%
    group_by(customer_id) %>%
    summarize(recency = min(startDate - date_of_purchase),
              first_purchase = max(startDate - date_of_purchase),
              frequency = n(),
              monetary_value = mean(purchase_amount))
```

2.1 Managerial Segmentation

In many cases, statistical segmentation explored in the Week 2 Recital is not used because the segments are not stable and it requires constant updating on the whole data set. Managerial segmentation uses fixed rules and only new purchase data needs to be used to update segments rather than the whole data set.

```r
########################################
## Simple segmentation on recency alone ##
########################################
seg1 <- customers_2015 %>%
    mutate(segment = ifelse(recency < 365, "active", "cold"))
## How many customers are in each group?
table(seg1$segment)
```

```
##
## active   cold
##   5398  13019
```

```
## Median of recency, frequency, monetary_value within each group
seg1 %>%
    select(-customer_id) %>%
    group_by(segment) %>%
    mutate(recency = as.numeric(recency)) %>%
    summarize_each(funs(median(.)))
```

segment	recency	first_purchase	frequency	monetary_value
active	59	1120 days	3	40
cold	1554	2354 days	1	30

```
################################################
## Multiple Segmentation using Recency Only  ##
################################################
seg2 <- customers_2015 %>%
    mutate(recency = as.numeric(recency)) %>%
    mutate(segment = cut(recency, breaks = c(Inf, 365 * 3:1, -1),
                         labels = c("active", "warm", "cold", "inactive")))

seg2 %>%
    select(-customer_id) %>%
    group_by(segment) %>%
    mutate(recency = as.numeric(recency)) %>%
    summarize_each(funs(median(.)))
```

segment	recency	first_purchase	frequency	monetary_value
active	59	1120 days	3	40.00000
warm	447	779 days	2	31.66667
cold	807	982 days	1	30.00000
inactive	2137	2633 days	1	30.00000

```
#######################################################
## Multiple Segmentation using Multiple Variables ##
#######################################################
segLevels <- c("inactive", "cold", "warm high value", "warm low value",
               "new warm", "active high value", "active low value", "new active")
## In original analysis "new warm" and "new active" were not defined with the same
## criteria. Warm was within 2 years but active within 1 year. I think calling someone
## new should be consistent no matter when they last purchased so I chose 1 year.
seg2015 <- customers_2015 %>%
    mutate(recency = as.numeric(recency),
           first_purchase = as.numeric(first_purchase)) %>%
    mutate(high_value = monetary_value >= 100) %>%
    mutate(new = first_purchase <= 365) %>%
    mutate(segment = cut(recency, breaks = c(Inf, 365 * 3:1, -1),
                         labels = c("active", "warm", "cold", "inactive"))) %>%
```

```
    mutate(segment = as.character(segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & new,
                             paste("new", segment), segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & !high_value,
                             paste(segment, "low value"), segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & high_value,
                             paste(segment, "high value"), segment)) %>%
    mutate(segment = factor(segment, levels = segLevels))

## Show each segment
table(seg2015$segment)
```

```
##
##          inactive              cold   warm high value    warm low value
##              9158              1903               235              1723
##          new warm active high value  active low value        new active
##                 0               573              3313              1512
```

```
seg2015Res <- seg2015 %>%
    select(-high_value, -new) %>%
    group_by(segment) %>%
    summarize_each(funs(median(.))) %>%
    as.data.frame()
seg2015Res
```

segment	customer_id	recency	first_purchase	frequency	monetary_value
inactive	105115	2137	2633	1	30.00000
cold	196200	807	982	1	30.00000
warm high value	205130	419	734	2	130.00000
warm low value	209650	455	784	2	30.00000
active high value	143220	38	1967	5	133.33333
active low value	143820	62	1969	5	35.83333
new active	255165	59	62	1	30.00000

2.2 Retrospective segmentation

If we go back in time, we can repeat the same analysis but using a different start date.

```
## Create new data frame of what we would have had up to the end of 2014
startDate <- as.Date("2014-12-31")
customers_2014 <- purchases %>%
    filter(date_of_purchase <= startDate) %>%
    group_by(customer_id) %>%
    summarize(recency = min(startDate - date_of_purchase),
              first_purchase = max(startDate - date_of_purchase),
              frequency = n(),
              monetary_value = mean(purchase_amount))
```

```
#####################################################
## Multiple Segmentation using Multiple Variables ##
#####################################################
segLevels <- c("inactive", "cold", "warm high value", "warm low value",
               "new warm", "active high value", "active low value", "new active")
seg2014 <- customers_2014 %>%
    mutate(recency = as.numeric(recency),
           first_purchase = as.numeric(first_purchase)) %>%
    mutate(high_value = monetary_value >= 100) %>%
    mutate(new = first_purchase <= 365) %>%
    mutate(segment = cut(recency, breaks = c(Inf, 365 * 3:1, -1),
                         labels = c("active", "warm", "cold", "inactive"))) %>%
    mutate(segment = as.character(segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & new,
                             paste("new", segment), segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & !high_value,
                             paste(segment, "low value"), segment)) %>%
    mutate(segment = ifelse( (segment == "warm" | segment == "active") & high_value,
                             paste(segment, "high value"), segment)) %>%
    mutate(segment = factor(segment, levels = segLevels))

## Show each segment
table(seg2014$segment)
```

```
##
##          inactive              cold   warm high value    warm low value
##              7512              2002               211              1923
##          new warm active high value  active low value        new active
##                 0               498              3187              1572
```

```
seg2014Res <- seg2014 %>%
    select(-high_value, -new) %>%
    group_by(segment) %>%
    summarize_each(funs(median(.))) %>%
    as.data.frame()
seg2014Res
```

segment	customer_id	recency	first_purchase	frequency	monetary_value
inactive	98130	2016.0	2392.0	1	30.0000
cold	170855	866.0	1013.5	1	30.0000
warm high value	193200	462.0	689.0	1	112.5000
warm low value	194050	481.0	678.0	1	30.0000
active high value	131125	42.0	1826.5	5	136.0417
active low value	131870	68.0	1845.0	5	35.0000
new active	224575	90.5	154.0	1	30.0000

2.3 Revenue Generation by Segment

Now that we've defined segments, we can explore which segments are generating revenue.

```
## Create data frame to hold revenue and join in full segmentation from 2015
revenue <- purchases %>%
    group_by(customer_id) %>%
    summarize(revenue = sum(purchase_amount[year_of_purchase == 2015])) %>%
    full_join(seg2015)
```

```
## Joining, by = "customer_id"
```

```
revenue %>%
    group_by(segment) %>%
    select(segment, revenue) %>%
    summarize(mean(revenue))
```

segment	mean(revenue)
inactive	0.00000
cold	0.00000
warm high value	0.00000
warm low value	0.00000
active high value	323.56894
active low value	52.30604
new active	79.16614

```
## How much money did we get in 2015 based on the segments from 2014?
revenue <- purchases %>%
    group_by(customer_id) %>%
    summarize(revenue = sum(purchase_amount[year_of_purchase == 2015])) %>%
    full_join(seg2014)
```

```
## Joining, by = "customer_id"
```

```
plotDat <- revenue %>%
    group_by(segment) %>%
    select(segment, revenue) %>%
    summarize(revenue_avg = mean(revenue)) %>%
    arrange(-revenue_avg)


ggplot(data = plotDat) +
    geom_bar(stat = "identity", aes(x = factor(segment, levels = plotDat$segment),
                                    y = revenue_avg)) +
    xlab("Segment") +
    ylab("Revenue") +
```

```
    getBaseTheme() +
    theme(axis.text.x = element_text(angle = 45, hjust = 1))
```