Paper112-2011

A Side of Hash for You To Dig Into

Shan Ali Rasul, Toronto, Ontario, Canada.

ABSTRACT

Within the realm of Customer Relationship Management (CRM) there is always a need for segmenting customers in order to gain insights into customer behaviour and being able to use that information to better align customers with current and ongoing marketing strategies. This paper discusses a macro created using SAS ® version 9.2, which makes use of the DATA step hash object for the purposes of customer segmentation.

INTRODUCTION

There are many techniques with regards to segmenting customer level data. This is a representation of one such method and how to utilize the power of hashing. There are a number of introductory articles available from past SAS conferences that outline the basic definition and syntax of the DATA step hash object. This paper highlights a macro developed for the purpose of segmenting customers based on a particular variable which in this case is the Recency, Frequency and Monetary score or RFM score. The paper does not cover the process of building an RFM score.

SEGMENTATION METHODOLOGY

The objective is to segment active customers into categories of behaviour critical to the success of your company, based on some kind of metric such as RFM (Curry et.al. 2000). Almost every company has come up with their own definition of what constitutes an active customer. In this paper active customers are defined as persons who have purchased goods or services from your company within the past 12 months (Curry et.al. 2000). For the purposes of this paper the customer segmentation is based on the active customer's most recent RFM score. RFM score is defined as Recency, Frequency and Monetary score. For each of the active customers a separate Recency score, Frequency score and Monetary score is created (each one of the three scores is a decile score, which ranges from 1 to 10). The three scores are then summed up to come up with a single RFM score per customer, with RFM score ranging from 3 to 30. The scope of this paper does not cover how to create those scores. The focus of the paper is on using the RFM scores to segment customers into a "standard customer pyramid" (Curry et.al. 2000) using the DATA step hash object.

According to Curry et.al. (2000) a standard customer pyramid is formed by clustering customers according to four categories based on their RFM score "Top", "Big", "Medium" and "Small". The four categories are listed as follows:

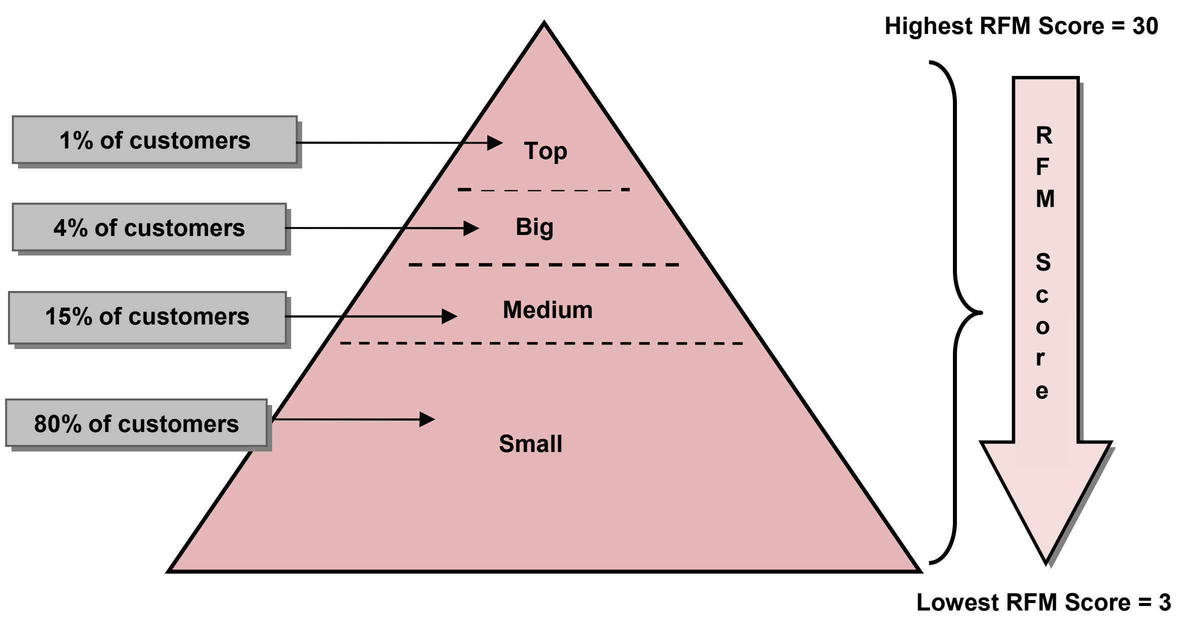
- "Top" Customers – the top 1% of active customers in terms of RFM score
- "Big" Customers – the next 4% of active customers in terms of RFM score
- "Medium" Customers – the next 15% of active customers in terms of RFM score
- "Small" Customers – the remaining 80% of active customers in terms of RFM score

With the help of DATA step hash tables, the RFM score is sorted in descending order from the highest score of 30 to the lowest score of 3, and the score is then used to categorize active customers into the "Top" 1%, "Big" 4%, "Medium" 15% and "Small" 80%.

Figure 1 below helps illustrate this idea from a visual perspective.

Figure 1

Customer Segmentation Pyramid (based on RFM Scores)



DATA STEP HASH OBJECT AND 'HASHS' MACRO

The DATA step hash object is implemented using a DATA step and object oriented programming syntax i.e. ***ObjectName.Method(parameters)*** *(*Parman 2006) . The hash macro, pet name 'hashs' was created with the point of view of being able to take any customer related data set with a pre-defined metrics like RFM scores or net sales etc and split it into the four customer pyramid segments mentioned above. Note the macro is dynamic enough that the percentage of those segments can be easily manipulated. There are three parts to the entire process. Part A uses the 'hashs' macro to split the data into the four segments. Part B puts the segments together into one data set and creates a field which helps identify each of the four segments. Part C looks at some general procedures like PROC FREQ, PROC MEANS in order to validate the results of the segmentation exercise.

PART A

Part A is made up of steps 1 -15. By explaining each of the steps it will help make the process of understanding the hash syntax and the workings of the 'hashs' macro clear.

Step 1: All variables from the data source that are to be used to define the hash table are defined in the length statement.

Step 2: Initiate the hash table 'g' and automatically load the table in descending key order. The result is a table with customers sorted in descending order by RFM Score.

Step 3: DefineKey is a method used by the hash object to define the hash table's key *(*Parman 2006).

Step 4: DefineData is a method used by the hash object to define the values returned when the hash table is searched *(*Parman 2006).

Step 5: DefineDone is a method used by the hash object to conclude the declaration of the hash table. When DEFINEDONE is called, DEFINEKEY and DEFINEDATA can no longer be called (Secosky et. al. 2006).

Step 6: Initialize the variables used in the hash object to missing values *(*Parman 2006).

Step 7: Initialize the hash iterator object.

Step 8: Specifies the name of the variable that contains the number of items in the hash object (SAS (R) 9.2 Language Reference).

Step 9: Creates the variables n1, n2, n3 & n4. Which represent the top 1 percent, next 4 percent, next 15 percent & bottom 80 percent of the customer segmentation pyramid. Note that to create the next 4 percent segment the percentage has to be set to 0.05 since the top 1 percent of the customers have already been accounted for. A similar approach is to be used for calculating the next 15 percent i.e. 0.20. The extra 5 percent accounts for the top 1 percent and next 4 percent of the total customer population.

Step 10: The FIRST method is used to point to the first item in the hash object (Secosky et. al. 2006) i.e. the first point with the largest key.

Step 11: The first do loop iterates through the first observation until the top 1 percent of observations in the hash table have been identified. The top 1 percent of customers are read into the data set called onepct. The NEXT method moves to the next item in the hash object.

Step 12: The second do loop iterates through the observation succeeding the last observation within the top 1 percent of observations in the hash table and until a full 5 percent of the observations have been processed. The result is that the next 4 percent of customers are read into the data set called fourpct. The NEXT method moves to the next item in the hash object.

Step 13: The third do loop iterates through the observation succeeding the last observation within 5 percent of observations in the hash table and until a full 20 percent of the observations have been processed. The result is that the next 15 percent of customers are read into the data set called fifteenpct. The NEXT method moves to the next item in the hash object.

Step 14: The LAST method in this case fetches the lowest-value item since the table was ordered in descending order by RFM Score i.e. Step 2.

Step 15: The last do loop iterates through the observation succeeding the last observation within the top 20 percent of observations in the hash table and until the remaining 80 percent of the observations have been processed. The result is that the next 80 percent of customers are read into the data set called eightypct. The PREV method returns the previous value of the item in the hash object. This is an important step in order to avoid non-zero values from being returned.

PART B

The four datasets created in Part A are concatenated and a field called segments is created that labels each one of the segments accordingly into 'Top', 'Big', Medium' & 'Small'.

PART C

The results of Part A and Part B are verified by performing some general validation procedures on the final data set. The section titled results provides a visual of the procedures applied in Part C.

THE HASH MACRO & OTHER DATA STEPS

PART A : - create the hash macro, which uses hash tables to create the four RFM score segments;

```
%macro hashs (dataset, order, seg_var, id_var, seg1, seg2, seg3);

data onepct fourpct fifteenpct eightypct; /*define the data sets that will store
                                            each of the four RFM score segments*/

1     length &id_var 8 &seg_var 8;          /*define variables used in the hash
                                            object*/

      if _n_=1 then do;

2     declare hash g (dataset: "&dataset", ordered: "&order");     /* hash table
                                                                 initiated  with the
                                            ORDERED parameter set to descending*/

3     g.DefineKey("&seg_var","&id_var");    /*identify fields to use as keys*/

4     g.DefineData("&seg_var","&id_var");   /*identify fields to use as data*/

5     g.DefineDone();                       /*complete hash table definition*/

6     call missing (&seg_var, &id_var);     /*initialize the variables to fill*/

7     declare hiter hi('g');                /*hash iterator object initiated*/
      end;

8     count=g.num_items;        /*counts number of items in the hash object*/

      n1=int(count * &seg1);    /*1 percent of customers*/
9     n2=int(count * &seg2);    /*5 percent of customers*/
      n3=int(count* &seg3);     /*20 percent of customers*/
      n4=int(count);            /*100 percent of customers*/

10    hi.first();               /*points iterator to first item in the hash table*/

      do i=1 to n;              /*iterates through 1 to n items in hash table*/
11    output onepct;            /*output sent to onepct dataset*/
      hi.next();                /*points iterator to next item in the hash table*/
      end;                      /*end of do statement*/

      do i=n+1  to n2;          /*iterates through n+1 to n2 items in the hash
                                  table*/
12    output fourpct;           /*output sent to fourpct dataset*/
      hi.next();                /*points iterator to next item in the hash table*/
      end;                      /*end of do statement*/

      do i=n2+1  to n3;         /*iterates through n2+1 to n3 items in the
                                  hash table*/
13    output fifteenpct;        /*output sent to fifteenpct dataset*/
      hi.next();                /*points iterator to next item in the hash table*/
      end;                       /*end of do statement*/
```

```
14 hi.last();                /*points iterator to last item in the hash table*/


   do i=n3+1  to n4;         /*iterates through n3+1 to n4 items in the hash
                             table*/
15 output eightypct;          /*output sent to eightypct dataset*/
   hi.prev();                /*points iterator to previous item in the hash
                             table*/
   end;                      /*end of do statement*/
   run;
   %mend hashs;
%hashs(rfm_score_dataset,descending,rfm_score,customer_no,.01,.05,.20);
```

PART B :- merge segmented datasets into one dataset and label each segment.

```
data regroup;
set onepct(in=a) fourpct(in=b) fifteenpct(in=c) eightypct(in=d);
length segments $10.;
if a      then segments='1. Top';
else if b then segments='2. Big';
else if c then segments='3. Medium';
else if d then segments='4. Small';
drop n n2 n3 n4 i count;
run;
```

PART C :- summarize the above created customer segments by RFM score.
Reference the next section titled results for the output of the code below.

```
proc freq data=regroup  noprint;
tables segments/nocol norow nocum  out=reg ;
run;

proc print data =reg noobs;
title 'RFM Score Segments';
format count comma12. percent 8.;
run;

proc means data=regroup   max min mean median maxdec=2 missing chartype ;
class segments;
var rfm_score;
title 'RFM Score Segments Statistics'; run;
```

RESULTS

```
                 RFM Score Segments

         segments      COUNT   PERCENT

         1.   Top          8         1

         2.   Big         35         4

         3.   Medium     131        15

         4.   Small      699        80
```

```
                 RFM Score Segments Statistics

                      The MEANS Procedure
               Analysis Variable : rfm_score

segments      N Obs   Maximum   Minimum    Mean   Median

1.   Top          8     30.00     28.00   29.13    29.00

2.   Big         35     28.00     26.00   27.17    27.00

3.   Medium     131     26.00     23.00   24.92    25.00

4.   Small      699     23.00      3.00   12.04    10.00
```

CONCLUSIONS

In summary what this paper has outlined is a method of using a hash macro to segment customers by RFM score into customer pyramid segments of 'top', 'big', 'medium', and 'small'. This exercise can be recreated to develop customer pyramid segments for two time periods e.g. comparing customers and the segments they belong to for the years 2009 & 2008. The comparison of the two time periods can reveal a migration matrix giving us a dynamic view of customer behaviour in a period of time (Curry et. al. 2000). The migration matrix can help answer questions like "Compared to last year how many customers are newly acquired, upgraded, dropped to inactive status, stayed within the same pyramid level or revived their status from inactive to active?"

The migration matrix can be used to develop a contact plan for how to deal with on-going customer behaviour. For example forecasting customers per pyramid segment and tying it into contact capacity and marketing budgets for next year could provide an optimal solution with regards to targeting strategies.

REFERENCES

Curry, J. and Curry, A. (2000). *Customer marketing method*. New York, NY: Free Press.

Parman, B. (2006). How to implement the SAS® DATA step hash object. Proceedings of the 2006 South East SAS® users group conference. Available

http://analytics.ncsu.edu/sesug/2006/SC19_06.PDF

SAS(R) 9.2 Language Reference: Dictionary, Second Edition, NUM_ITEMS Attribute. Available

http://support.sas.com/documentation/cdl/en/lrdict/62618/HTML/default/a002588144.htm

Secosky, J. and Bloom, J. (2006). Getting started with the DATA step hash iterator. Proceedings of the 2006 South Central SAS® users group conference. Available

http://www.scsug.org/SCSUGProceedings/2006/SecoskyJason.pdf

ACKNOWLEDGMENTS

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at shan_ali_rasul@yahoo.ca