Paper AA-07-2015

Slice and Dice your customers easily by using SAS® Clustering Procedures

Yanping Shen, Alliance Data Systems, Columbus, OH

1. ABSTRACT

With the brighter economy, customer base is expanding at a fast pace for the retail industry in recent years. This growth in customer base coupled with the increase availability of many data attributes at customer level and the move towards personalized marketing efforts possess a new challenge to customer relationship management. In this paper, we illustrate how to innovatively utilize FASTCLUS and CLUSTER procedures in SAS® to create customer segments using large data sets with variables of different units of measurement. The paper concludes by suggesting several business applications that can be developed for each customer segment as well as opportunities to further slice and dice different customer segments to enhance personalized marketing efforts.

2. INTRODUCTION

Since introduced in 1950s, customer segmentation has always been one of the most powerful methods used in marketing and customer relationship management. Businesses of marketers understand that customers are different and not every offer will be right for every customer at exact time. Clustering is one way of organizing the customers into groups of similar traits or shopping behavior. This helps marketers to understand their customers' difference and similarities and thus design marketing strategies specifically targeting each group of customers.

Clusters could be constructed based on a lot of customer information: demographics, psychographics, desired benefit or product preference, and purchase behavior, etc. One can collect this information either directly from the firm internal database or from survey results of a representative customer's sample. Nowadays, many methods have been developed to perform clusters analysis, and they can be grouped into two categories based on clusters structure: hard clustering and soft clustering. The difference between these two is that one observation can only belong to one cluster in hard clustering while one can belong to multiple clusters in soft clustering. Common hard clustering algorisms including k-means and agglomerate hierarchy clustering have been implemented widely in customer segmentation.

In this work, we are interested in segmenting a very large population into appropriate clusters, which are hundreds of millions of loyalty members. To achieve that, we illustrated how to perform disjoint cluster analysis utilizing the PROC FASTCLUS AND PROC CLUSTER procedures provided in SAS, and the combination of those two procedures turns out to be a very efficient way of clustering data on a large scale.

3. DATA

In this study, we leverage a huge amount of data from the rich sources of a loyalty marketing retailer which included demographic characteristics, geographical characteristics, transactional information; loyalty cards use behavior, etc. Checking data quality and screening out 'bad' data in an efficient way is a big challenge for this kind of data we have. To circumvent this problem, we have developed several macros to efficiently examine data quality, impute missing values, and handle outliers. In the next section, we illustrate our data cleaning process.

3.1 MISSING AND ABERRANT VALUES

Considering one person might hold several loyalty cards and hence have multiple records of their personal information in our internal database, we first started by checking whether these records from different sources are consistent for the same person. To achieve that, we developed a macro named %CHECK_VALUE to check whether the information associated with different loyalty cards are consistent for the same cardholder. In addition, this macro also supports checking missing values as one of its byproducts.

The output from the macro is a table containing detailed information about the percentage of missing values, aberrant values, etc. PROC FASTCLUS is used to handle missing values by calculating adjusted distance using nonmissing values. The percentage of observations with inconsistent values are less than 2% in most cases, so we randomly choose one record although detailed investigation of why the aberrant values occurred would help make a better choice. In addition, variables with a large proportion of missing or inconsistent values are excluded from the following cluster analysis.

Name	pct_Multiple_value	pct_Single_values	pct_missing_values
ADV_HH_EDUCATION	0.01731	0.96375	0.01894
ADV_HH_EDUCATION_IND	0.01608	0.96619	0.01773
ADV_HH_MARITAL_STATUS	0.02839	0.95388	0.01773
ADV_HH_MARITAL_STATUS_IND	0.02925	0.95302	0.01773
ADV_LENGTH_OF_RESIDENCE_IND	0.00809	0.97418	0.01773
ADV_NUMBER_OF_ADULTS	0.03691	0.94536	0.01773
ADV_NUMBER_OF_ADULTS_IND	0.01848	0.96379	0.01773
ADV_TARGET_INCOME	0.05512	0.92715	0.01773
ADV_TARGET_INCOME_IND	0.02440	0.95787	0.01773
BIRTH_DATE	0.13492	0.86507	0.00001
GENDER	0.01258	0.97826	0.00916
NUMBER_OF_CHILDREN	0.00496	0.25663	0.73841
p_INCOME_AMT	0.50768	0.48289	0.00944

Table 1 : Example of output from *%CHECK_VALUE* macro

3.2 VARIABLES

To segment loyalty members properly, it's desirable to collect a large set of variables reflecting their attributes as thoroughly as possible. We started with a wide array of variables containing loyalty members' information ranging from demographics, shopping behavior, brand engagement, etc. After excluding variables with a lot of missing values or aberrant values, and variables which are highly correlated with other variables, we came up with a final list of variables used in this cluster analysis which fall in four major categories: Demographic, Loyalty member engagement, Shopping pattern and Shopping Category.

3.3 OUTLIERS

As cluster analysis is very sensitive to outliers, we developed another macro called %CAP_VAR to eliminate the influence of outliers on the clustering results. In basic terms, this macro could examine the outliers determined by your own specific criteria, and then replace the extreme values with either the high or low quantiles you specified in the macro arguments. The output form this macro is a new dataset with outliers of all numeric variables being capped except for those you specify to keep original values and thus exclude from capping.

3.4 VARIABLE STANDARDIZATION

Variables of larger variance would have more influence on the clustering results since they tend to contribute more to the distance calculations. To eliminate this effect caused by variables measurements on different units, the STANDARD procedure is used to standardize the variables to a mean of 0 and standard deviation of 1. The sample code below created the dataset containing transformed variables.

```
/*standardize variables*/
PROC STANDARD DATA=mydata MEAN=0 STD=1 OUT= mydata_2;
     VAR &var_list;
RUN;
```

4. METHODOLOGY

4.1 SAMPLING

We are interested in cluster hundreds of millions of loyalty members in this work. In this case, the huge amount of data we gathered would make it infeasible to perform the cluster analysis on the whole population. To circumvent this technical issue, we started by taking a random sample of 1 million loyalty members from the whole population using the SURVEYSELECT procedure.

```
/*select a random sample constituting 1 million members from the total population*/
PROC SURVEYSELECT DATA=ttl_pop  METHOD=srs N=1000000 OUT=cust_sample;
RUN;
```

This sample data would be used in our initial cluster analysis.

4.2 CLUSTERING

FASTCLUS and CLUSTER are two SAS procedures commonly used for clustering analysis in many fields. FASTCLUS uses a method called nearest centroid sorting and group observations into disjoint clusters where one sample would be classified into one and only one cluster. In basic terms, the algorithms consist of two steps. The algorithm starts by choosing initial cluster centroids or seeds from the dataset as the mean of each cluster. In the first step, each observation is assigned to the cluster with the nearest seed. In the second step, the seeds are updated by taking the average of observations within the corresponding cluster. The distance between old seed and new seed is also calculated and the procedure would repeat these two steps until the calculated distance is less than a defined threshold. In other words, it would repeat until the relative change of seeds is not significant. This procedure scale well to large number of samples but require the number of clusters to be specified.

The CLUSTER procedure is based on the usual agglomerative hierarchy clustering using a bottom up approach: each observation begins with its own cluster and the closest two clusters are merged together. The successive merging of two close clusters repeat until only one cluster containing all observations is left. This whole process could be visualized as a tree or dendrogram, with the root being the final cluster and the leaves representing each individual observation. The distance of clusters could be measured by different metrics and the CLUSTER procedure provides eleven methods to determine the merge strategy.

One advantage of using the CLUSTER procedure for cluster analysis is that one can estimate the number of clusters based on its output statistics. However, CLUSTER is not practical in in dealing with large data sets since the CPU time is of the order $O(N^2)$ or $O(N^3)$ where N is the number of total observations in your dataset. Instead, the time of the FASTCLUS procedure is only of O(N) and therefore it would be more suitable for large dataset. In this paper, we combined the two procedures together to develop a two stage clustering approach. In the first stage, we used PROC FASTCLUS to form a very large number of preliminary clusters. In the second stage, we use the PROC CLUSTER to produce the hierarchal clusters with those initial clusters. This strategy is also suggested by SAS document to cluster a large dataset hierarchically.

4.3 SCORING

Since it is computationally infeasible to perform the cluster analysis on the whole population in our system due to the large amount of data, we utilized the SEED statement in the FASTCLUS procedure to score all the observations and assign them to the preliminary clusters directly in the first stage. A set of seeds are derived from running FASTCLUS on the selected sample constituting one million members as mentioned before. Then each observation in the whole population is directly assigned to the nearest seed. In this case, no clustering iterations are performed to determine the cluster membership and it greatly reduces the need of computer resource and computation time. After obtaining the initial clusters for the whole population, the following hierarchy clustering would be performed using the PROC CLUSTER procedure as usual.

5 RESULTS

5.1 USING THE FASTCLUS PROCEDURE TO FIND INITIAL CLUSTERS

The FASTCLUS procedure takes the dataset which contains selected variables with their values capped and standardized as the input and produces the PRELIM dataset in which each observation is assigned to one specific

cluster. The converge option is set to 0 thus FASTCLUS would run to full convergence. The Maxiter option is set to 500 to decrease the execution time. The procedure would automatically terminate after 500 iterations even if it doesn't converge, and this wouldn't be an issue since FASTCLUS is designed so that it doesn't need to converge to find good clusters. And we choose 300 clusters by specifying MAXC=300. We defined a large number of clusters here to maintain low number of observations in each cluster and thus reduce the effect of this additional FASTCLUS on following hierarchy clustering analysis. Replace option is set at its default value enabling unlimited seed replacements in the initializing step. Since the default method of Replace tends to obtain seeds widely separated, we also set Radius option at default value which places no restrictions on the minimum distance each seed must be separated from others. Finally, in the VAR statement, we specified the variables used in the cluster analysis.

```
/*Run fastclus to cluster the data sample into 300 groups*/
PROC FASTCLUS DATA=mydata3 SUMMARY MAXC=300 MAXITER=500 CONVERGE=0
      MEAN=mean OUT=prelim CLUSTER=preclus OUTSEED= clusterSeeds;
      VAR age -- on_us_group_11_spend;
RUN;
```

The output dataset PRELIM produced by the out statement contains two new variables in addition to the original values, PRECLUS and DISTANCE. The PRECLUS variable indicates the cluster membership and the DISTANCE variable is the distance from the observation to the cluster seed. The PRELIM dataset would be merged with the output dataset created by the hierarchical clustering later. Another dataset produced by MEAN statement contains the cluster means and two other variables, _FREQ_ and _RMSSTD_, and it is used as the input dataset by the CLUSTER procedure later. Finally, the CLUSTERSEEDS dataset output by the OUTSEED option will be used for scoring the whole population.

5.2 USING THE CLUSTER PROCEDURE TO CONSTRUCT FINAL CLUSTERS

PROC CLUSTER supports eleven choices of linkage strategies, including Ward, average, complete, two stage density, etc. A lot of studies have been performed to compare these various methods. Generally, average linkage or ward's minimum variance method have been proved to perform better compared with other methods in terms of finding compact clusters. Other methods, like single linkage or density linkage are more capable of detecting clusters with elongated or irregular shapes on the other hand. In this paper, we utilized the %CLUS macro to explore different methodology and also examine their output clusters by looking at different metrics.

In the following macro, PROC CLUSTER is first invoked to analyze the preliminary clusters created by the previous FASTCLUS procedure. You can specify the method you want to try by assigning values to the MTD macro variable. The TREE procedure is also run to create the tree diagram which visualizes the successive merge of clusters. In addition, you can specify the number of clusters to plot by assigning an integer value to the N_CLUS macro variable. Finally, PROC CANDISC and PROC SGPLOT are utilized to check cluster distribution graphically. The CANDISC procedure finds canonical variables which provide best separation between clusters. The SGPLOT procedure then plots the first two canonical variables can1, can2 to show the cluster membership.

```
%MACRO CLUS(MTD, N_CLUS)
PROC CLUSTER DATA=mean METHOD=&mtd PRINT=15 CCC PSEUDO OUTTREE=fortree;
      VAR age -- on_us_group_11_spend;
   COPY preclus;
RUN;
/*creating the hierarchical tree*/
AXIS1 VALUE=(height=0.5);
PROC TREE DATA=fortree NCL=&n_clus OUT=out HAXIS=axis1;
      HEIGHT _rsq_;
    COPY  age -- on_us_group_11_spend preclus;
RUN;
PROC CANDISC DATA=clus noprint OUT=can;
    CLASS cluster;
      VAR age -- on_us_group_11_spend;
RUN;
PROC SGPLOT DATA=can;
    SCATTER y=can2 x=can1 / GROUP=cluster;
```

```
RUN;
%MEND;
%CLUS(WARD, 10)
%CLUS(SINGLE, 12)
%CLUS(COMPLETE, 12)
…
```

One example of the outputs created by %CLUS(WARD, 10) is shown below:
Figure 1 displays the last 15 generations of the cluster history produced by the CLUSTER procedure using ward methods. The statistics listed in the last three columns are useful in determining the number of clusters for the dataset. For instance, large values of CCC generally indicate good clusters while its negative values indicate outliers. As shown in the figure, the CCC values steadily increases with more number of clusters and level off when the cluster number is 12. pseudo t2 statistic is another method of determining the number of clusters and large values indicate that two clusters shouldn't be merged together. The appropriate way to interpret the t2 value is to read down the column until you find the value significantly larger than the previous one and then move back one column. Moving down the PST2 column, you can see possible clustering levels at 15, 10 and 7. The combination of the two statistics indicates that 10 or 12 clusters would be a good choice for our dataset.

NCL	Clusters	Joined	Freq	SpRSq	RSq	ERSq	CCC	Ps_F	PsT2
15	CL62	CL29	98191	0.0083	0.728	0.514	2544	190000	38000
14	CL44	CL39	186145	0.0087	0.72	0.506	2525	190000	120000
13	CL38	CL14	203727	0.01	0.71	0.497	2500	200000	72000
12	CL15	CL22	183480	0.012	0.698	0.488	2459	200000	41000
11	CL27	CL16	72430	0.0161	0.681	0.477	2222	210000	26000
10	CL25	CL18	126412	0.0185	0.663	0.465	2136	210000	39000
9	CL23	CL13	327504	0.0198	0.643	0.451	2062	220000	120000
8	CL9	CL19	437366	0.0333	0.61	0.435	1849	220000	160000
7	CL11	CL51	85104	0.0333	0.577	0.417	1563	220000	41000
6	CL8	CL24	556147	0.0437	0.533	0.391	1270	220000	170000
5	CL7	CL10	211516	0.0738	0.459	0.356	839	210000	86000
4	CL12	CL5	394996	0.0852	0.374	0.307	504	190000	100000
3	CL17	CL4	408217	0.1186	0.255	0.244	84.4	170000	110000
2	CL3	CL34	419754	0.1223	0.133	0.158	-184	150000	92000
1	CL2	CL6	975901	0.1331	0	0	0	.	150000

Figure 1. Cluster History of the final 15 iterations of running PROC CLUSTER using WARD method

Figure 2 shows the tree diagram created by the tree procedure by default. This figure provides a graphic view of the information shown previously in figure 1. The R^2 increases as the branches go down from the root. And the first 12 clusters account for over half of the variation (70%).

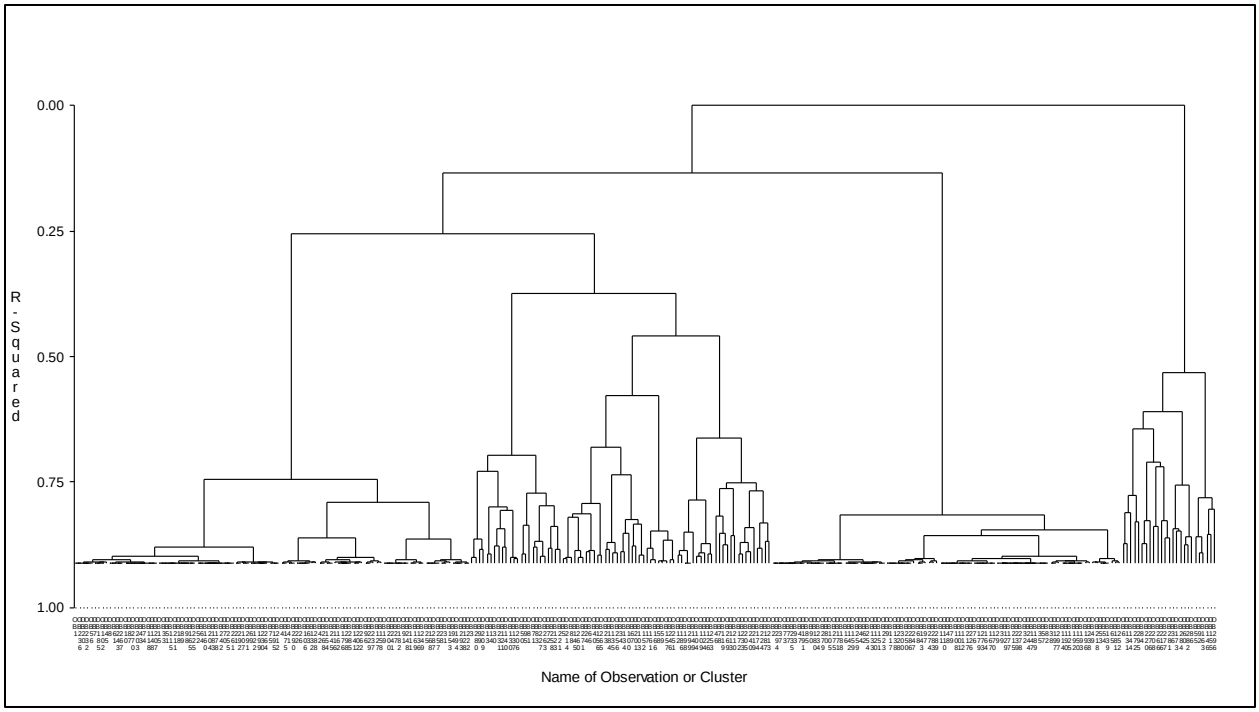

Figure 2. An example of the tree diagram generated by PROC CLUSTER using WARD method

Figure 3 illustrates the separation of twelve clusters produced by the CLUSTER procedure using the WARD method. Combining statistics in figure 1, the tree diagram in figure 2 and the plot of cluster distribution in figure 3, it suggests that 12 clusters seem to be a reasonable choice for our dataset.

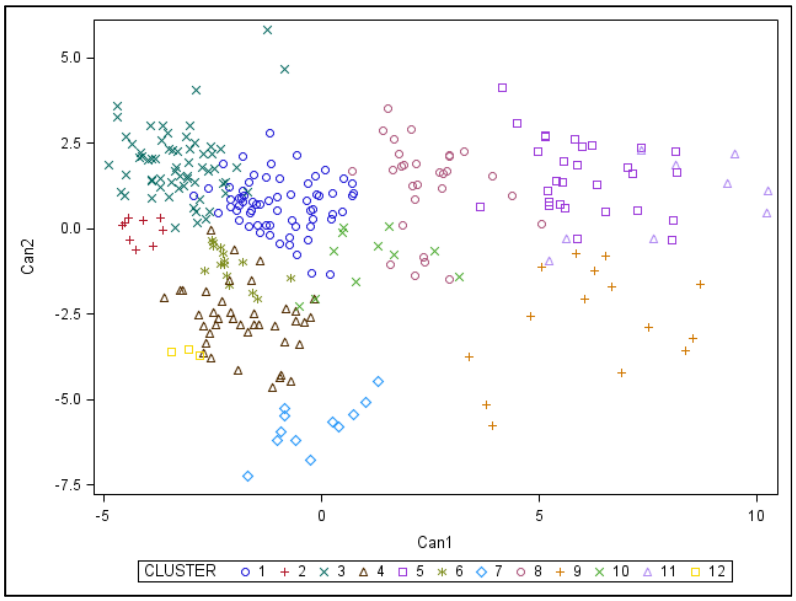

Figure 3. An example of the resulting plot of 12 clusters from PROC CLUSTER with METHOD=WARD

5.3 CREATING CLUSTERS FOR WHOLE POPULATION BY SCORING

As mentioned before, it would be infeasible for us to conduct the cluster analysis on the whole population directly. Thus, we utilized the seed option in FASTCLUS procedure to assign the whole population consisting of hundreds of millions of loyalty members to the 300 preliminary cluster seeds created from running FASTCLUS on the selected

sample. Prior to applying the scoring method on the whole population, we validated the consistency and stability of this scoring method by examining the clustering results from scoring different random samples. Detailed examination reveal that the clusters derived from the new samples exhibit very similar characteristics compared with the clusters form the original sample. The following code illustrates how to score new observations without changing cluster rules.

```
/*scoring all the observations using the seed dataset */
PROC FASTCLUS DATA=ttl_mydata3 SUMMARY MAXC=300 MAXITER=0 SEED = clusterSeeds
      OUT=ttl_prelim MEAN=ttl_mean;
      VAR age -- on_us_group_11_spend;
RUN;
```

The output TTL_MEAN dataset will be used by the CLUSTER procedure to perform the hierarchical clustering as discussed before.

6. CONCLUSION AND BUSINESS APPLICATIONS

In this paper, we illustrated how we successfully cluster hundreds of millions of loyalty cardholder into different groups by elaborating on data cleaning, data mining and exploration of different methods for clustering analysis provided by SAS. After the clusters are constructed, we examined multiple attributes and profiles of loyalty members and find that each cluster revealed very interesting characteristics distinct from others. Business leaders can now develop unique business strategies that are suitable for each specific cluster to achieve their goal. For instance, one of our clusters constitutes of members who show unique interest in digital engagement. Specifically, they have much higher propensity to enroll with their mobile phone, shop online, and sign up for paperless statements. Diving into their demographics profile, these members are relatively young and more likely to be males compared with the rest of the population. This provides unique insight for business leaders to take a targeted action to achieve a strategic goal.

With the help of FASTCLAS and CLUSTER procedures in SAS®, you can slice and dice your customers, allowing you to develop unique solutions for each cluster to achieve your business goals.

REFERENCES

- SAS® 9.2 USER'S GUIDE, HTTP://SUPPORT.SAS.COM/DOCUMENTATION/
- Lefait, G. and Kechadi, T, (2010) *"Customer Segmentation Architecture Based on Clustering Techniques"* Digital Society, ICDS'10, Fourth International Conference, 10-02-2010

ACKNOWLEDGMENTS

The author will like to thank Delali Agbenyegah, Yin Chen, Tim Sweeney, Candice Zhang, Dong Yan, Daniel Basco and the entire Predictive Analytics team at Alliance Data Systems for their support.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Yanping Shen
Alliance Data Systems
3100 Easton Square Place
Columbus, OH, 43219
Yanping.shen@alliancedata.com

APPENDIX

```
*-------------------------------------------------------------------------------------------------------------------------*
*  The %CHECK_VALUE MACRO is used to examine your data quality. Specifically,                                             *
*  it would calculate the percentage of missing values and inconsistent values                                           *
*  in your dataset. The macro takes three arguments and they are defined below:                                          *
*                                                                                                                        *
*      1) data =your own dataset                                                                                         *
*      2) exclude = variables excluded from examination                                                                  *
*      3) Id = The variable used as a key for your dataset                                                               *
*-------------------------------------------------------------------------------------------------------------------------*;
    %macro check_value(data, exclude,Id);
    /*create macro variable var_list including names of all interested variables */
    proc contents data=&data out=temp noprint;run;
    proc sql STIMER EXEC NOPRINT;
        select name into: var_list separated by ' ' from temp where name not in
        &exclude;
    quit;
    /*check the missing / abberant values for every variable by looping through the variable list*/
    %let n=%sysfunc(countw(&var_list));
    %do i=1 %to &n;
        %let val = %scan(&var_list,&i);
        proc sql noprint;
            create table var_&i as
                select  &Id, count(distinct(&val)) as &val from &data
                 group by &Id
            order by &Id;
        quit;
        proc freq data=var_&i noprint; table &val./out=count_&i;run;
        data count_&i._2(rename=(&val=freq));
            set count_&i;
            length name $50.;
            name="&val";
        run;
        proc append base=sum_var_1  data=count_&i._2 force;run;
        proc datasets nolist;delete var_&i count_&i count_&i._2;run;
    %end;
    /*group variable values into three types*/
    proc sql noprint;
        create table sum_var_2 as
        select case freq when 0 then 'missing values' when 1 then 'Single values' else 'Multiple values' end as type,
    sum(count) as count,sum(PERCENT) as percent, name
        from sum_var_1
        group by name, type;
    quit;
    proc transpose data=sum_var_2 out=sum_var_3 (drop=_NAME_ _LABEL_) prefix=count_;
        var count ;
        by name;
        id type;
    run;
    proc transpose data=sum_var_2 out=sum_var_4 (drop=_NAME_ _LABEL_) prefix=pct_;
        var percent ;
        by name;
        id type;
    run;
    /*combine count and frequency columns and impute missing values with 0 */
    data sum_var_type;
        merge sum_var_3 sum_var_4;
        by name;
```

```
        array numeric{*} _numeric_;
        do i=1 to dim(numeric);
                if numeric{i}=. then numeric{i}=0;
        end;
        drop i;
    run;
    proc datasets nolist;delete sum_var_1 sum_var_2 sum_var_3 sum_var_4 temp;run;
    %mend;

*-------------------------------------------------------------------------------------------------------------------------*
* The %CAR_VAR MACRO is used to cap variables to exclude influence of outliers                                           *
* on clustering analysis. The macro takes four arguments and they are defined below:                                     *
*                                                                                                                         *
*     1) data =your own dataset                                                                                           *
*     2) exclude = variables excluded from capping                                                                        *
*     3) low = The lowest quantile used to define outlier: values smaller than                                            *
*       low would be replaced by the value at the low quantile                                                            *
*     4) high = The highest quantile used to define outlier: values larger than                                           *
*       high would be replaced by the value at the high quantile                                                          *
*-------------------------------------------------------------------------------------------------------------------------*;
    %macro cap_var(data,exclude,low,high);
    /*create macro variable numvar including names of all interested variables */
    proc contents data = &data out = temp; run;
    proc sql STIMER EXEC NOPRINT;
        select name into :numvar separated by ' ' from temp where type = 1 and
        name not in &exclude.;
    quit;
    /*calculate low and high quantile as specified in the low and high option for each variable */
    proc means data = &data p&low. p&high. noprint;
        var &numvar.;
        output out = p&low. p&low.() = ;
        output out = p&high. p&high.() = ;
    run;
    data var_pctl (drop = _type_ _freq_); set p&low. p&high.; run;
    /*transpose dataset to align low and high quantile as new columns */
    proc transpose data = var_pctl name = Name out = var_pctl; run;
    data var_pctl; set var_pctl;
        rename  col1= p&low. col2 = p&high.;
        label   col1= p&low. col2 = p&high.;
    run;
    /*create three sets of macro variables to ease the later looping*/
    data null;
    set var_pctl;
        l=length(name);
        call symput('vname_'||left(_n_), substr(name,1,l));
        l=length(p&low.);
        call symput('p_low_'||left(_n_), substr(p&low.,1,l));
        l=length(p&high.);
        call symput('p_high_'||left(_n_), substr(p&high.,1,l));
        call symput('varnum', _n_);
    run;
    /*create output dataset with all interested variables values capped*/
    data  &data._2;
    set &data;
        %do i=1 %to &varnum;
                if  &&vname_&i.. <=  &&p_low_&i.. then  &&vname_&i..  = &&p_low_&i.. ;
                if  &&vname_&i.. >=  &&p_high_&i..  then  &&vname_&i..  = &&p_high_&i..;
        %end;
    run;
    proc datasets lib=work; delete p&low. p&high. Null Temp Var_pctl;run;
    %mend;
```