Paper AA13-2015

Two-Stage Attribute Match and Its Applications in Loyalty Marketing

Jie Liao, Alliance Data Card Service, Columbus, OH
Yi Cao, Alliance Data Card Service, Columbus, OH

ABSTRACT

In loyalty marketing, it is particularly challenging to predict customers' shopping behavior with limited information. So we propose a Two-Stage Attribute Match (TSAM) algorithm with the purpose of finding a look-alike community for these customers and making inference on their future shopping behavior. In the presentation, we explore a variety of applications for this algorithm. For example, by using this algorithm, we are able to infer purchasing amount levels for customers' nine merchant category groups simultaneously. Another example on a client's reward program incrementality analysis is also slightly touched. To make the algorithm accessible in SAS, a macro %TwoStageAM is developed to facilitate its future usage.

INTRODUCTION

As "know more, sell more" becomes a catchword in marketing world, more and more managers are aware that the recipe to win the heart of customers is to anticipate what customer wants based on solid understanding of them. Business world is swarmed with successful cases. Amazon makes a fortune by knowing up front which book might cater to its customer with a recommendation engine. Walmart also proves the power of knowing more with its well-known diaper and beer story. However, our information of customers is not always adequate enough for us to make sound prediction and know them more. In fact, lack of information is almost a norm in many cases. Just imagine how many new customers come to retail stores every day, with only couples of purchases in history and few months of tenure on the brand. Therefore, how to play with limited information and exploit the data that is available to its maximum capacity is an inevitable topic.

Attribute match has a notch above other methods by requiring no rigorious statistical assumptions on the data and adapting well to varying situations, which just fits to the issue of limited information. The basic idea behind attribute match is simple: If two customers look alike each other in almost all aspects, they should be considered "identical" and able to provide inference of unobserved aspects to each other. Of course we're not going to match customers on all aspects. With the sole curiosity of future shopping behavior, we should only need to match on relevant attributes. That being said, as long as we select the right attributes to match on, there should be a great chance that matched customers' future shopping behaviors are also similar. The first case study in the paper provides that evidence.

Bearing the idea in mind, we propose the two-stage attribute match algorithm which conducts a one-to-many match. "One" stands for customer of interest, whose future shopping behavior you're curious of. "Many" denotes customers in the reference pool, whose shopping behaviors have already been observed so as to serve as inference to the customer of interest. Two-stage attribute match is far more than just making inference with limited information. It has a variety of other applications like inferring multiple shopping behaviors all at once. Examples will be showcased later in the paper.

ALGORITHM

Two-stage attribute match is intended to find a cluster of customers from reference pool resembling each customer of interest. Before clarifying why two stages rather than one stage for the attribute match, we would like to first discuss the different nature of match methods that are usually adopted for categorical variables and numerical variables. When matching customers on categorical variables, simple exact match will suffice. For instance, if both customers are females, they are matched on gender, otherwise, they are not.

However, when matching customers on numerical variables, the match process becomes tricky. To put it simply, it's hard to consider annual spend on apparel will distinguish one customer from another by only $0.10. Some nuance should be allowed in numerical match. And your tolerance of the variance, which calls for expert knowledge on the data, will directly affect match result. Very often a distance is employed to measure the closeness of numerical variables, without fixed cutoff for numerical match. In another word, numerical match is more like fuzzy match.

Some researches have been made on combining categorical match and numerical match in one stage by transforming either of the two distinct match processes. Zengyou He, Shengchun Deng and Xiaofei Xu [3] brought up

an idea to measure distance for categorical variables and establish connection between categorical (k-modes) and numerical (k-median) clustering, which achieve analogous results to attribute match. Such studies usually require a second thought on choosing weight for the transformed process when summing up results, which is not intuitive enough for most users to make confident decision.

Therefore, instead of converting categorical variables to numerical variables or the other way around with the purpose of matching simultaneously, we decide to handle them separately with two stages and start with the straightforward categorical match.

STAGE ONE – CATEGORICAL MATCH

For each pair of customers, all categorical variables will be examined in stage one. In a single categorical variable scenario, only if the customer pair share the same level, they can be deemed as matched. When there're multiple categorical variables, the inspection is carried over one by one followed by aggregation of their match results, which gives us a match rate. Match rate is defined as the ratio of the number of matched categorical variables to the number of total categorical variables, as shown below.

$$\theta_{AB} = \sum_{i=1}^{K_c} I_{ABi} / K_c \ , \tag{1}$$

$$I_{ABi} = \begin{cases} 1, & if\ matched\ on\ variable\ i \\ 0, & otherwise \end{cases} \tag{2}$$

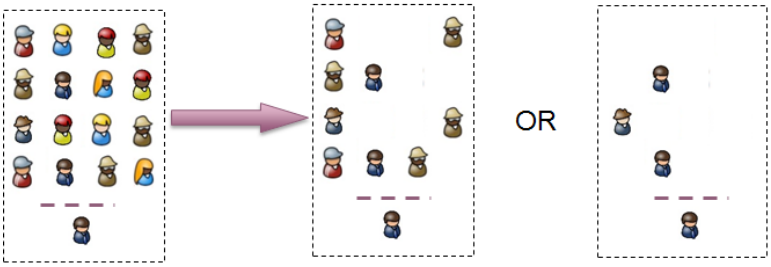
Where θ_{AB} is match rate between customer A and customer B, K_c is number of categorical variables, I_{ABi} is an indicator of whether customer A and customer B are matched on the ith variable with matched coded as 1 and 0 otherwise. Figure 1 gives an example of how to calculate match rate for customer A and customer B with four categorical variables. Only gender is matched in this case, thus match rate is 25%.

Figure 1. Example for Match Rate

	Customer A	Customer B	
Education	University	High School	Match Rate θ_{AB} = 25%
Gender	Female	Female	
State	TX	FL	
Occupation	Professor	Student	

A higher match rate amounts to a closer proximity of the customer pair. Consequently, to determine which customers in the reference pool resemble the customer of interest concerning only categorical variables, we refer to their match rates with the customer of interest and filter out the lower ones. Here comes a classical conversation of variance versus bias. If we set the bar high, selected customers from the pool would be closer to the customer of interest, which helps reduce the bias of inference from the matched group. Meanwhile, fewer customers are chosen due to the high standard, which increases the variance of the inference.

Figure 2. Two Typical Categorical Match Scenarios



So a careful consideration needs to be given to the balance between match rate cutoff and matched sample size baseline. Besides, considering numerical match succeeds categorical match, a reasonable amount of shortlisted customers from stage one need to be set aside for stage two, lest categorical match outweighs numerical match. In view of the above-mentioned concerns, we suggest loosening the standard properly in stage one. And feel free to reselect matched customers again after implementing two-stage attribute match. We need users to specify minimum matched sample size for stage one in our macro %TwoStageAM, with a default value 100. Based on the sample size constraint, the macro %TwoStageAM will automatically match those with highest match rate to each customer of interest.

STAGE TWO – NUMERICAL MATCH

Moving on to stage two, we will evaluate numerical variables for each pair of customers that are passed on from stage one. Instead of summing up each variable's match result like categorical match, numerical match will utilize Euclidian distance to measure proximity of all numerical variables simultaneously. To bring variables with different scales into alignment, normalizing the values for each variable before putting into Euclidian distance calculator is indispensable. Euclidean distance is calculated with formula below.

$$d_{AB} = \sqrt{\sum_{i=1}^{K_n} (x'_{Ai} - x'_{Bi})^2} \ , \tag{3}$$

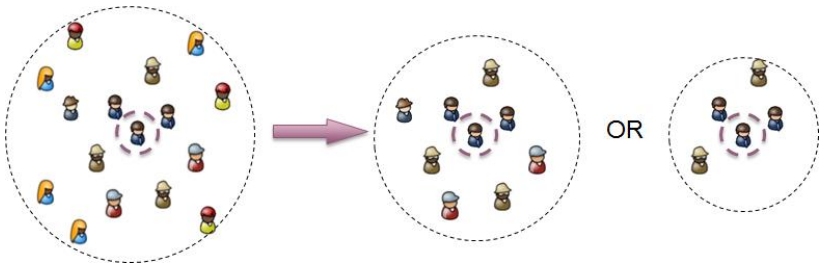
$$x'_{Ai} = \frac{x_{Ai} - \mu_i}{\sigma_i} \ , \tag{4}$$

Where d_{AB} is Euclidean distance between customer A and customer B, K_n is number of numerical variables, x_{Ai} is value of the ith variable for customer A, x'_{Ai} is corresponding normalized variable value, μ_i is the ith variable's sample mean and σ_i is its sample standard deviation. Figure 3 shows a simple example, where values in the parenthesis are normalized values.

Figure 3. Example for Euclidean Distance

	Customer A	Customer B	
Spend	\$50 (0.5)	\$100 (1.0)	Euclidean Distance d_{AB} = 0.648
Risk Score	650 (0.3)	700 (0.4)	
Credit Limit	\$1,000 (-0.1)	\$5,000 (0.3)	

Figure 4. Two Typical Numerical Match Scenarios



As with categorical match, trade-off between variance and bias when determining Euclidian distance cutoff claims our attention (see Figure 4). Our macro %TwoStageAM provides a range of cutoffs for your reference based on the data, and its default cutoff is the median of all customer pairs' Euclidian distance, which varies with data.

After two stages of attribute match, our macro %TwoStageAM will provide a name list from reference pool that are matched to each customer of interest, along with their match rate and Euclidean distance. As mentioned before, you are free to make second selection on the output.

In the end, we'd like to measure how well the customers are matched with the two-stage attribute match. Stefano M. Iacus, Gary King, and Giuseppe Porro (2011) [1,2] have introduced L-distance imbalance metrics to gauge similarity of two continuous distribution, which could supplement the function of our macro %TwoStageAM by measuring the closeness of two distributions. The closer the L distance is to 0, the more similar two distributions are. A good match tend to bear a very small L distance.

In case study session, we will borrow this metric to validate the closeness of matching with or without application of match algorithm.

MACRO FUNCTIONARLITY

The macro %TwoStageAM is shown below:

```
%TwoStageAM(        test= ,
                    control = ,
                    ID = ,
                    distance_cutpoint = ,
                    numvarlist = ,
                    baseline_match_size=
                    ) ;
```

Table 1 explains the meanings of each argument in the macro.

Table 1. Description of Macro Parameters

Parameter	Explanation
Test	Customers of interest
Control	Customers reference pool
ID	ID index
Distance_cutpoint	User specified cut-off point for Euclidian distance, default = P50
Numvarlist	Numeric variable names (if not specified here, will be treated as categorical)
Baseline_match_size	User specified minimum sample size to maintain for stage one categorical match, default = 100

APPLICATIONS

In this session, two case studies are given to illustrate how two-stage attribute match is applied in loyalty marketing. The first case study is to make inference on customers' shopping behavior from their look-alike group and the second case study is to evaluate the effectiveness of the store credit card reward program.

CASE STUDY I

In credit card loyalty marketing, we are able to collect information as to which merchant category customers shopped and how much they spent. By understanding customer's spending habits in merchant category groups, marketers would have insights into which products or services their customers might be interested in. However, most marketers only reply on basic descriptive statistics or even gut-feelings to make their best guesses because of the troubles in building hundreds of predictive models for so many merchant category groups and going through all kinds of hypothesis test every time. As such, two-stage attribute match becomes hugely advantageous as it makes concurrent multiple inferences possible and doesn't demand any rigorous statistical assumptions.

The objective of this case study is to prove the concept that spend levels on merchant category groups can be inferred for a customer by his/her look-alike community. We took the study with the following steps:

(1) Select nine merchant category groups for performance measure

We select nine merchant category groups of interest ranging from gas station, and grocery to clothing store, as listed below. In general, those nine merchant category groups would represent 70% of credit card spending.

Gas Station	Grocery Store and Supermarket
Amusement and Entertainment	Clothing Store
Restaurants	Personal Service
Drug Store and Pharmacies	Business/Contracted/Repair/Electronic Miscellaneous
Lodging	

(2) Create control and test groups for attribute match

From our customer database, we randomly select credit cardholders who actively shopped between July-2014 and June-2015 and split them into test (customers of interest) and control (reference pool) groups for attribute match. We also obtain the following two months' (July-2015 to August-2015) transaction data for validation. If matched customers exhibit similar spend level in the following two months, it means it's reasonable to make inference for test group customers from their matched control groups.

(3) Fomulate attributes

There are several types of attributes: historical spend on all nine merchant category groups, customers' credit card APR, tenure, payment, balance, purchase attributes etc. All historical spend are aggregated at two-month level and encoded as 0 for high level, 1 for median level and 2 for low level. See Table 2.

Table 2. Historical Spend Amount Level for Nine Merchant Category Groups

ACCT_ID	GAS_STATIONS_w1	GAS_STATIONS_w2	GAS_STATIONS_w3	GAS_STATIONS_w4	GAS_STATIONS_w5	GAS_STATIONS_w6	GROCERY_STORE___SUPERMARKET_w1	GROCERY_STORE___SUPERMARKET_w2
710000004	2	2	2	1	1	2	2	2
710000005	2	2	2	1	1	2	0	2
710000006	0	0	0	0	1	0	0	0
710000007	0	0	0	0	0	0	0	0
710000008	0	0	0	0	0	0	1	0
710000009	0	2	0	0	0	2	0	2
710000010	2	2	2	1	1	2	0	0
710000012	2	1	2	0	0	2	0	2

We end up with 120 attributes in final data set: 78 categorical attributes and 42 numeric attributes.

(4) Match customers in test group with customers in control group using %TwoStageAM.

The matching procedure is executed in %TwoStageAM macro as follows.

```
%TwoStageAM(test= test, control = control, id = ACCT_ID, distance_cutpoint = P50,
numvarlist=  Tenure APR Delinquent Age, baseline_match_size= 100);
```

Table 3 shows the matching output of the macro.

Table 3. Macro Output

	T_ACCT_ID	CHAR_MATCH_RATE	DISTANCE	C_ACCT_ID	NO_MATCHED
1	710000081	0.7692307692	2.0225136822	710000078	153
2	710000081	0.7692307692	1.3828500664	710000126	153
3	710000081	0.7692307692	2.7459365723	710000208	153
4	710000081	0.7692307692	1.5457635703	710000491	153
5	710000081	0.7692307692	3.5336180399	710001327	153
6	710000081	0.7692307692	2.2675359326	710001810	153
7	710000081	0.7692307692	3.1417978992	710001962	153
8	710000081	0.7692307692	5.7880654721	710002386	153
9	710000081	0.7692307692	1.8143954642	710002479	153
10	710000081	0.7692307692	7.0025592785	710002636	153

The macro outputs five columns. T_ACCT_ID is the customer account ID of test group and C_ACCT_ID is customer account ID of control group. Customers from C_ACCT_ID are those passing matching criteria of both stages and thus paired with T_ACCT_ID. CHAR_MATCH_RATE is the match rate of categorical attributes and DISTANCE is the Euclidean distance of numerical attributes for each matched pair. NO_MATCHED represents the number of customers in control group that are matched to the customer of interest in test group.

Here is how to read the output: For cardholder “710000081”, the algorithm finds out 153 cardholders from control group that look alike him/her. That is, 153 matched pairs. For pair “710000081”and “710000078”, 77% of their categorical attributes are exactly the same and Euclidean distance between them is 2.02. Thus, we can see that “7100000126” looks more alike “710000081” than “710000078” does because even though their match rates are the same, the distance between “710000126” and “710000081” is shorter.

(5) Make inference

Since each customer of interest is matched to a group rather than an individual, to make inference for the customer, we need to figure out a way to generate a single statistics which is representative of the matched group. And it should be studied case by case. Here, instead of simply taking the mode of spend level 0, 1, 2 as inference, we choose weighted voting method in the sense that even if a spend level is not majority in the group, as long as its proportion is way greater than expected, it should be taken as our inference. Table 4 is spend level inference that we made for customers in test group (the table was truncated due to limited space).

Table 4. Spend Level Inference for Merchant Category Groups

ACCT_ID	INFERRED_GAS_STATIONS_w7	INFERRED_GROCERY_STORE___SUPERMARKET_w7	INFERRED_RESTAURANTS_w7	INFERRED_DRUG_STORES_AND_PHARMACIES_w7
710000004	0	2	2	1
710000005	2	2	2	1
710000006	0	0	0	1
710000007	0	0	0	0
710000008	0	0	0	0
710000009	0	0	0	0
710000010	2	0	0	0
710000012	0	0	1	0

To validate the results, we use confusion matrix to evaluate the performance of the inference.

Table 5. Confusion Matrix

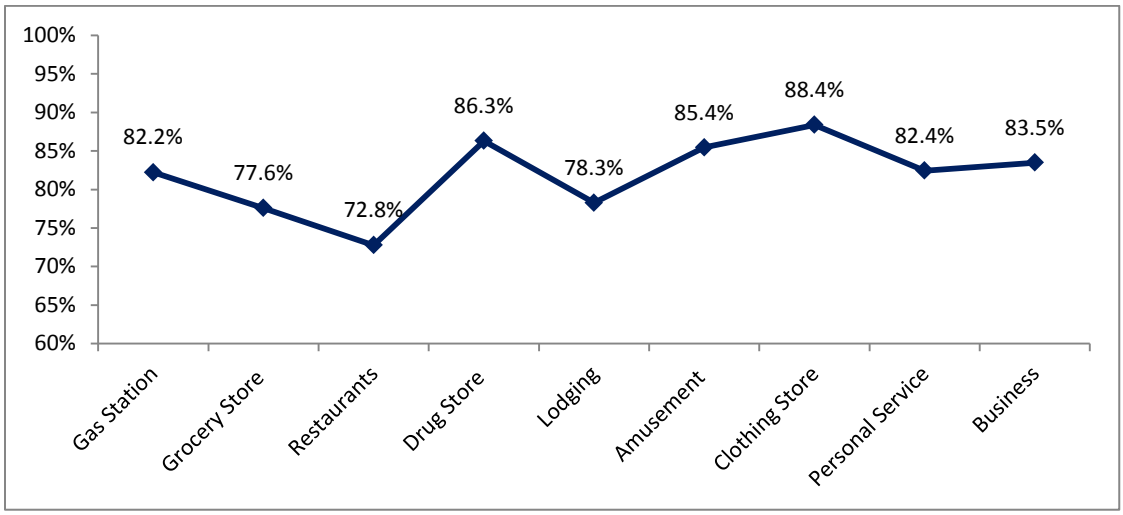
		Actual Spend Level		
		High (0)	Medium (1)	Low (2)
Inferred Level	High (0)	a		
	Medium (1)		b	
	Low (2)			c

The accuracy is calculated by

$$\text{Accurary} = \frac{(a + b + c)}{\text{Total}}$$

Figure 5 demonstrates the accuracy for all nine merchant category groups.

Figure 5. Inference Accuracy



The accuracy for nine merchant category groups all fall between 73% and 88% which is fairly good in marketing analytical domain. Instead of constructing nine independent predictive models, we leverage attribute match algorithm to make inferences for nine merchant category groups together with satisfactory accuracy.

CASE STUDY II

Another application of two-stage attribute match is to control for the confounding influence of pretreatment variables in observational data. In control/test scenario, the key goal of matching is to prune observations from the data so that the remaining data have better balance between test and control groups, which means the empirical distributions of the covariates in the groups are more similar. This is particularly desirable in causal inference in practice as marketers would like the comparison of treatment and control to be apple-to-apple such that the effect of treatment is justified in a solid statistical manner. Next, we are going to discuss how the two-stage attribute match can help evaluate effectiveness for the store credit card reward program.

In our case study, the reward program is designed particularly for store credit card owner to incent their in-store purchases. We start the analysis by randomly selecting a few thousand store loyalty members before reward program was launched. Once the program began, some loyalty members turned into reward credit cardholders whereas others did not. To measure the lift of sales the reward program has driven, we need to compare the sales between cardholders who have enjoyed the benefit of reward program and non-cardholders who are free from any reward incentives. However, if compared directly without matching, the observed incremental lift is -$8.39 for average in-store sales and -0.50 for average number of trips. The numbers indicate that the reward program brings neither incremental in-store sales nor trips. But if we pay closer attention to attribute distributions for membership tenure, web shopper indicator, historical sales etc., we will notice that the imbalance metrics for those attributes are between 10% and 40% even before program launch(See Table 6). In another word, there's a discrepancy in distributions of attributes for cardholders and non-cardholders which makes them incomparable in the first place. The negative lift is highly likely due to imbalances.

Table 6. Imbalance Measure with/without Matching

Attribute	Imbalance metric (random select without match)	Imbalance metric (random select with match)
Cycle XX	25%	5%
Dept_CNT	27%	3%
Recency	40%	9%
Sales_01	8%	0%
Sales_02	16%	1%
Sales_03	6%	0%
Tenure	26%	5%
Trip_01	10%	0%
Web_Ind	2%	0%
Cat_Ind	10%	0%

To adjust for the imbalances in attributes, we apply two-stage attribute match to make sure the distribution of each attribute between cardholders and non-cardholders are balanced prior to comparison. As shown in Table 6, the imbalance measures become pretty small for all attributes after attribute match, which enables marketer to re-evaluate the incremental in-store sales and trip driven by reward program. The captured lift due to reward program is $19.68 in sales and 0.33 in number of trips in the end.

CONCLUSION

Two-stage attribute match allows users to automate one-to-many matching process to a great extent. Unlike conventional matching methodologies, which either can't handle categorical and numerical attributes together, or have to convert one to the other at the risk of distorting the nature of the attributes, the proposed two-stage attribute match can tackle categorical variable and numerical variable at the same time without sacrificing their own characteristics. Meanwhile, the algorithm has a great flexibility in that by adding more sensible attributes, we are able to achieve better accuracy without resorting to rigorous statistical assumptions which are indispensable in parametric model building.

The macro we code for the algorithm is very user-friendly because it only asks for input of test and control data set along with user-specified distance cutoff and matched sample size baseline. Even if the later two parameters are not specified, the default values can still provide fairly good results and be applied to many scenarios. It will save users a lot of time in exploration if he/she is new to the macro.

As for applications, the algorithm demonstrates great capability in both causal inference which tests effect of certain treatment and concurrent multiple inferences which are made on customer shopping habits. As a next step, we will further refine the algorithm by assigning more weight on relatively more important attributes.

REFERENCES

[1] Stefano M. Iacus, Gary King, and Giuseppe Porro. Multivariate Matching Methods That Are Monotonic Imbalance Bounding. Journal of American Statistical Association. Vol. 106, No. 493. 2011.

[2] Stefano M. Iacus, Gary King, and Giuseppe Porro. Causal Inference without Balance Checking: Coarsened Exact Matching. The Society for Political Methodology. 2011. 8.

[3] Zengyou He, Shengchun Deng and Xiaofei Xu. Approximation Algorithms for *K*-Modes Clustering. Computational Intelligence. Volume 4114 of the series Lecture Notes in Computer Science pp 296-302. 2006.

ACKNOWLEDGMENTS

We would like to profusely thank Qiang Zhu, Ning Ma, Thom Young and Tim Sweeney for their guide and mentorship during the course of this work. We also thank Marketing Insights Department at Alliance Data for providing us the opportunity to attend the conference.

CONTACT INFORMATION

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

Name: Jie Liao
Enterprise: Alliance Data Card Services
Address: 3100 Easton Square Place
City, State ZIP: Columbus, Ohio, 43210
Work Phone:
E-mail: jie.liao@alliancedata.com

Name: Yi Cao
Enterprise: Alliance Data Card Services
Address: 3100 Easton Square Place
City, State ZIP: Columbus, Ohio, 43210
Work Phone: 614-944-3716
E-mail: yi.cao@alliancedata.com

APPENDIX

Due to confidentiality, we only provide part of the code for macro %TwoStageAM as below.

```
/*================= (A). Normalize numerical variables ===============*/;
*----- extract mean and std. from control group ----*;
proc summary data = temp_cont;
       var c_numvar1 - c_numvar&p_num.;
       output out = mean(drop=_TYPE_ _FREQ_) mean()= ;
       output out = std(drop=_TYPE_ _FREQ_) std()= ;
run;
data _NULL_;
       set mean;
       %do i = 1 %to &p_num.;
       call symput("mean"||"&i.",&&c_numvar&i..);
       %end;
run;
data _NULL_;
       set std;
       %do i = 1 %to &p_num.;
```

```
	call symput("std"||"&i.",&&c_numvar&i..);
	%end;
run;

*----- normalization for control group: reference pool ----*;
data temp_cont;
	set temp_cont;
	%do i = 1 %to &p_num.;
	if &&std&i.. ^=0 then &&c_numvar&i.. = (&&c_numvar&i.. - &&mean&i..)/&&std&i..;
	else do; &&c_numvar&i.. = (&&c_numvar&i.. - &&mean&i..);end;
	%end;
run;
*----- normalization for test group: customer of interest ----*;
data temp_test;
	set temp_test;
	%do i = 1 %to &p_num.;
	if &&std&i.. ^=0 then &&t_numvar&i.. = (&&t_numvar&i.. - &&mean&i..)/&&std&i..;
	else do; &&t_numvar&i.. = (&&t_numvar&i.. - &&mean&i..);end;
	%end;
run;

%put p_num = &p_num.;
%put p_char = &p_char.;



/*========== (B). Pair test case with every control case    ==============*/;
proc sql;
	create table match_out(drop = distance1 - distance&p_num.)
	as select a.&acct_id., b.&acct_id. as cont_&acct_id.,
		%do i = 1 %to &p_char.;
			case when a.&&t_var&i.. = b.&&c_var&i.. then 1 else 0 end as m&i.,
		%end;
		sum(&m_ls.)/&p_char. as match_rate,
		%do j = 1 %to &p_num.;
			(a.&&t_numvar&j.. - b.&&c_numvar&j..)**2 as distance&j.,
		%end;
		sqrt(sum(&distance_ls.)) as distance
	from temp_test a
	cross join
	temp_cont b
	;
quit;



/*===============(C). Stage One: Categorical Match ==================*/;
*----- calculate matched count above different match rate level ----*;
%do i = 1 %to &p_char.;
	proc sql;
		create table match_cnt_char_&i.
		as select distinct &acct_id., min(match_rate) as char_mr_cutpoint format
= percentn10., COUNT(*) as match_count
		from match_out(where = (match_rate >= &i./&p_char.))
		group by &acct_id.
		;
	quit;
%end;
data _match_cnt_char_;
	set match_cnt_char_1 - match_cnt_char_&p_char.;
run;

*----- apply baseline_match_size constraint ----*;
proc sql;
	create table char_mr_cutpoint
	as select distinct &acct_id., max(char_mr_cutpoint) as char_mr_cutpoint,
```

```
min(match_count) as match_count
       from _match_cnt_char_
       where match_count >= &baseline_match_size.
       group by &acct_id.;
quit;

proc sql;
       create table match_out(drop=m1 - m&p_char.)
       as select a.*, b.char_mr_cutpoint
       from match_out a
       inner join char_mr_cutpoint b
       on a.&acct_id. = b.&acct_id.
       where a.match_rate >= b.char_mr_cutpoint;
quit;


/*================(D). Stage Two: Numerical Match ==================*/;
*----- apply distance_cutpoint constraint ----*;
proc sql;
       create table match_out
       as select &acct_id., cont_&acct_id., match_rate, char_mr_cutpoint, distance,
&distance_cutpoint. as distance_cutpoint
       from match_out(where = (distance <= &distance_cutpoint.))
       ;
quit;
```