Paper 812-2017

A General SAS® Macro to Implement Optimal N:1 Propensity Score Matching Within a Maximum Radius

Brian Murphy and Kathy H. Fraeman, Evidera, Waltham, MA

ABSTRACT

A propensity score is the probability that an individual will be assigned to a condition or group, given a set of baseline covariates when the assignment is made. For example, the type of drug treatment given to a patient in a real-world setting might be non-randomly based on the patient's age, gender, geographic location, and socioeconomic status when the drug is prescribed. Propensity scores are used in many different types of observational studies to reduce selection bias. Subjects assigned to different groups are matched based on these propensity score probabilities, rather than matched based on the values of individual covariates. Although the underlying statistical theory behind the use of propensity scores is complex, implementing propensity score matching with SAS® is relatively straightforward. An output data set of each subject's propensity score can be generated with SAS using PROC LOGISTIC. And, a generalized SAS macro can generate optimized N:1 propensity score matching of subjects assigned to different groups using the radius method. Matching can be optimized either for the number of matches within the maximum allowable radius or by the closeness of the matches within the radius. This presentation provides the general PROC LOGISTIC syntax to generate propensity scores, provides an overview of different propensity score matching techniques, and discusses how to use the SAS macro for optimized propensity score matching using the radius method.

INTRODUCTION

In experimental studies and controlled clinical trials, subjects or patients are randomly assigned to a condition or group. However, in real-world observational studies, the “assignment” of a person to a group or condition is usually not randomly based. For example, the type of drug treatment given to a patient in a real-world setting may be based on conditions that exist when the drug is prescribed, such as the patient’s age, gender, geographic location, and/or socioeconomic status. In epidemiologic terms, this non-random assignment of subjects or patients to different treatment groups can produce something known as “selection bias.” A study with selection bias – where patients are not randomly assigned into groups – can cause the resulting statistical analyses of the study’s data to be distorted, unless this selection bias is accounted for in the study’s analyses.

Propensity scores are used in observational studies to reduce selection bias, by matching subjects or patients on the probability that they would be assigned to a specific group. A propensity score is simply a probability that a subject would be assigned to a specific group, and matching subjects on propensity scores produces comparison groups of subjects who would be equally likely to have been assigned to the study’s group or condition.

The underlying statistical theory behind the use of propensity scores and propensity score matching is beyond the scope of this paper. This statistical theory is well explained in some of the listed reference articles. Further, the use of propensity score matching as a means of controlling selection bias in observational studies is not the only method that can be used to control for selection bias, nor is the propensity score method consistently endorsed or used by all epidemiologists and statisticians who analyze observational data. This paper neither encourages nor discourages the use of propensity scores in the analysis of observational data to control for selection bias.

The purpose of this paper is to demonstrate how to implement propensity score matching using the radius method with SAS, which is one of several methods used to match on propensity score. (Baser, 2006) One potential drawback of propensity score matching using the radius method is the difficulty in knowing *a priori* what radius is reasonable.

COMPUTING PROPENSITY SCORES

A propensity score is simply a probability, a number ranging from 0 to 1. A propensity score is the probability that a subject will be assigned to a condition or group, based on conditions that exist at the time of the group assignment.

The basic SAS syntax to generate propensity scores using PROC LOGISTIC is given below:

```
PROC LOGISTIC data=patient_variables descending;
     model drug_treat_flag = <BASELINE VARIABLES> ;
     output out=propensity_score pred = prob_treat;
     run;
```

The components of the PROC LOGISTIC are broken down as follows:

- PATIENT_VARIABLES is the data set with one observation per subject or patient, that includes the binary group assignment variable and the baseline variables that play a role in the group assignment
- DESCENDING is the PROC LOGISTIC option that gives the probability the outcome will be true
- DRUG_TREAT_FLAG is the binary 1/0 treatment group variable that has a value of 1 if the subject was treated and 0 if the subject was not treated
- <BASELINE VARIABLES> are the variables used in the propensity score model. These baseline variables must reflect the conditions that existed before and up to the time the subject was assigned to the treatment group. Any variables that reflect conditions that occurred after the assignment of the treatment group are "outcome" variables and cannot be included in the model. Types of baseline variables that can be included in a propensity score model include age, gender, geographic location, and variables that reflect health status at the time of group assignment.
- PROPENSITY_SCORE is the name of the output data set that contains all of the variables in the original data set PATIENT_VARIABLES, plus the new probability variable PROB_TREAT
- PROB_TREAT is the name of the variable with the predicted probability, with values ranging from 0 to 1

Deciding which specific baseline variables to use in a propensity score model is the most complex part of this process and is dependent on variable availability and the needs of each study. The only generalization that can be made is that the values of these baseline variables must reflect the conditions before and up to the time of the group assignment.

MATCHING PROPENSITY SCORES

A variety of methods and algorithms can be used to match patients assigned to different groups based on propensity scores. These methods include modifications of matching patients on the actual propensity score, or on matching patients based on the percentage group of the score.

The method of matching patients in different groups based on propensity scores demonstrated here is based on matching on an allowable absolute difference between exact propensity scores, or a "radius" around the score. This matching is done using a generalized SAS macro for propensity score matching that can match a "control group" to a "patient group" at an N:1 ratio.

The generalized macro can either do completely random matching of patients to controls within the maximum radius, or it can use one of two different algorithms to either maximize the number of propensity score matches, or to give matching priority to the closest possible matches. The algorithm that optimizes the number of matches is based on obtaining the matches for patients with the fewest possible number of matches first, and the algorithm that optimizes the number of close matches is based on assigning the closest matches first.

The input parameters to the generalized propensity score matching program are:

- `pat_dsn` = The name of the SAS data set with the treated patients
- `pat_idvar` = The name of the patient ID variable in PAT_DSN, can be character or numeric
- `pat_psvar` = The name of the propensity score probability variable in PAT_DSN
- `cntl_dsn` = The name of the SAS data set with the untreated patients
- `cntl_idvar` = The name of the patient ID variable in CNTL_DSN, can be character or numeric
- `cntl_psvar` = The name of the propensity score probability variable in CNTL_DSN
- `match_dsn` = The name of the output SAS data set with the patient IDs for the matched pairs
- `match_ratio` = The matching ratio, must be a number from 1 to N for N:1 control to patient matching
- `score_diff` = A number between 0 and 1 that gives the allowable absolute difference (radius) between the treated and control patients' matched propensity scores
- `opt` = The type of optimization used to match patient. The default is "none", where the matches will be totally randomized. The value "num" will optimize the number of matches by matching the patients with the fewest number of matches first. The value "close" will optimize the closeness of the matches within the allowable radius by assigning the closest matches first.
- `seed` = An optional input parameter, which is the seed for the random number generator

The entire code for this matching macro is given in the Appendix to this paper, and the SAS code to call the macro is shown below:

```
/******************************************************************/
/* Separate patients treated with the drug from untreated patients
/******************************************************************/
DATA prop_score_treated
     prop_score_untreated;
     set propensity_scores;
     if drug_treat_flag = 1 then output prop_score_treated;
     else if drug_treat_flag = 0 then output prop_score_untreated;
     run;


/**********************************************************************/
/* 1:1 Matching with an absolute difference between propensity scores
/* of 0.01, random matching (no value given for parameter OPT, defaults to
/* none
/**********************************************************************/
%psmatch_multi(pat_dsn     = prop_score_treated,
               pat_idvar   = pat_id,
               pat_psvar   = prob_treat,
               cntl_dsn    = prop_score_untreated,
               cntl_idvar  = pat_id,
               cntl_psvar  = prob_treat,
               match_dsn   = matched_pairs1,
               match_ratio = 1,
               score_diff  = 0.01
               );
```

```
/*********************************************************************/
/* 2:1 Matching with an absolute difference between propensity scores
/* of 0.05, matching optimized for number of matches
/*********************************************************************/
%psmatch_multi(pat_dsn     = prop_score_treated,
               pat_idvar   = pat_id,
               pat_psvar   = prob_treat,
               cntl_dsn    = prop_score_untreated,
               cntl_idvar  = pat_id,
               cntl_psvar  = prob_treat,
               match_dsn   = matched_pairs2,
               match_ratio = 2,
               opt         = num,
               score_diff  = 0.05);

/*********************************************************************/
/* 1:1 Matching with an absolute difference between propensity scores
/* of 0.001, matching optimized for the closeness of the matches
/*********************************************************************/
%psmatch_multi(pat_dsn     = prop_score_treated,
               pat_idvar   = pat_id,
               pat_psvar   = prob_treat,
               cntl_dsn    = prop_score_untreated,
               cntl_idvar  = pat_id,
               cntl_psvar  = prob_treat,
               match_dsn   = matched_pairs3,
               match_ratio = 1,
               opt         = close,
               score_diff  = 0.001
               );
```

If the macro takes a very long time to run without completion, the chosen value of the allowable radius (SCORE_DIFF) is probably too large for the input data, and a smaller radius should be used. As previously stated, the disadvantage of the radius method of propensity score matching is the inability to know *a priori* what the optimum matching radius should be, and often finding the optimum allowable matching radius must be done by trial and error.

AN EXAMPLE OF COMPARING UNMATCHED AND PROPENSITY SCORE MATCHED PATIENTS

Propensity scores are used for determining probabilities other than the probability of a subject being treated with a specific drug. Propensity scores can also be used to predict if a patient will be assigned to a condition.

A study was conducted using inpatient hospitalization data to look at the incremental costs and resource utilization for patients who developed a surgical site infection (SSI) following coronary artery bypass graft (CABG) surgery. The study compared these outcomes between patients who did develop a post-operative infection to the patients who did not develop a post-operative infection. However, the probability that these patients developed a post-operative infection following CABG surgery is not random. Surgery patients who are older and sicker at the time of their surgery have a higher probability of developing an SSI, and the costs of treating these older and sicker patients would be higher anyway, even if they did not develop an infection following surgery.

Propensity score matching was used to match patients on the probability that they would develop an SSI following CABG surgery. In other words, we wanted to compare the costs and resource utilization of two groups of patients who underwent CABG surgery who were equally likely to develop an SSI following surgery. One group of "equally likely to develop an infection" patients did develop the post-operative infection, and the other group of equally likely patients did not.

The risk factors for developing an SSI following CABG surgery have been widely published, and some of the baseline factors used on the propensity score model for this study included:

- Patient age and gender
- Patient comorbidities at the time of surgery, such as diabetes, obesity, COPD, and renal disease
- Characteristics of the hospital where the surgery was performed, such as urban/rural, teaching, hospital size (number of beds), annual volume of CABG surgeries performed at the hospital, geographic location of the hospital
- Characteristics of the CABG surgery, such as number of vessels involved and surgery time

The resulting propensity scores from this logistic regression model were the probability that a patient would develop an infection following CABG surgery. The patients who developed an SSI were matched to patients who didn't develop an SSI with a 1:1 matching, where the absolute difference between propensity scores was +/- 0.01, and using the matching algorithm to maximize the number of resulting matches.

Table 1 shows some of the patient characteristics before and after propensity score modeling:

Patient Characteristics	**Post-CABG SSI** **n = 3,126**	**No post-CABG SSI Before propensity score matching** **n = 55,877**	**No post-CABG SSI After propensity score matching** **n = 3,126**
Age, years (Mean, SD)	66.56 (10.94)	64.6 (10.73)	66.03 (10.85)
Gender (n, %)			
Male	2,000 (64.0%)	41,433 (74.2%)	2,045 (65.4%)
Female	1,126 (36.0%)	14,444 (25.8%)	1,081 (34.6%)
Baseline Comorbidities (n, %)			
Diabetes	1,623 (52.2%)	22,221 (39.7%)	1,616 (51.7%)
Obesity	658 (21.0%)	9,179 (16.4%)	645 (20.6%)
COPD	1,099 (35.2%)	12,704 (22.7.8%)	1,120 (35.8%)
Renal Disease	946 (30.3%)	5,549 (9.9%)	914 (29.2%)
Congestive Heart Failure	1,070 (34.2%)	8,398 (15.0%)	1,069 (34.2%)

Table 1. Patient Characteristics Before and After Propensity Score Matching

The propensity score matched patients who did not develop an SSI have baseline characteristics that are very similar to the patients who did develop an SSI – slightly older, larger percentages of females, and are sicker at baseline as reflected by their increased numbers of baseline comorbidities.

Table 2 shows some of the patient outcomes before and after propensity score matching:

Patient Outcomes	**Post-CABG SSI** **n = 3,126**	**No post-CABG SSI Before propensity score matching** **n = 55,877**	**No post-CABG SSI After propensity score matching** **n = 3,126**
Total Hospitalization Days			
Mean (SD)	16.0 (10.4)	7.8 (3.7)	9.3 (4.8)
Median (range)	14 (3-117)	7 (1-74)	8 (1-48)
Died During Hospitalization			
Yes	128 (4.1%)	770 (1.4%)	116 (3.7%)
Total Cost of CABG Hospitalization			
Mean	$47,874	$28,061	$32,164
Median	$40,060	$25,527	$28,478

Table 2. Patient Outcomes Before and After Propensity Score Matching

The propensity score matched patients who did not develop an SSI have more hospitalization days, death, and total cost of hospitalization than the patients without an SSI before propensity score matching, but they do not have as many adverse outcomes as the patients who developed a Post-CABG SSI.

CONCLUSION

Analysis of observational data collected to compare the effects of a primary classification or treatment variable on outcomes will need to be adjusted for the non-random classification of subjects with this primary variable. This non-random classification of subjects is called "selection bias", and propensity score matching provides a way to adjust for selection bias in observational studies. The implementation of propensity score matching with SAS is straightforward, involving a logistic regression model with PROC LOGISTIC and a method for matching subjects' propensity score probabilities generated with PROC LOGISTIC.

REFERENCES

Baser, O. Too Much Ado about Propensity Score Models? Comparing Methods of Propensity Score Matching. Value in Health 9(6) 377-384. 2006.

Foster EM. Propensity Score Matching: An Illustrated Analysis of Dose Response. *Medical Care* 41(10) 1183-1192. 2003.

Leslie RS, Ghomrawi H. The Use of Propensity Scores and Instrument Variable Methods to Adjust for Treatment Selection Bias. Proceedings of the SAS Global Forum 2008, San Antonio, TX. 2008.

Parsons LS. Reducing Bias in a Propensity Score Matched-Pair Sample Using Greedy Matching Techniques. *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*, Long Beach, CA. 2001.

Parsons LS. Performing a 1:N Case-Control Match on Propensity Score. *Proceedings of the Twenty-Sixth Annual SAS Users Group International Conference*, Montreal, Canada. 2004.

Patkar A and Fraeman KH. Economic Impact of Surgical Site Infection Post-CABG Surgery Using Multi-Institutional Hospitalization Data. *Presented at the 3rd. Joint Meeting of the Surgical Infection Society-North America and the Surgical Infection Society of Europe, Chicago, IL.* 2009.

Rosenbaum PR and Rubin DB. The Central Role of the Propensity Score in Observational Studies for Causal Effects. Biometrica 70, 41-55. 1983.

SAS Institute, Inc. 2008 “Usage Note 30971: How can I compute and match observations on propensity scores?” http://support.sas.com/kb/30/971.html

CONTACT INFORMATION

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

Kathy H. Fraeman
Director, Senior Data Analyst
Evidera
7101 Wisconsin Ave #1400
Bethesda, MD 20814
Work Phone: +1 240 235 2525
Fax: +1 301 654 9864
E-mail: kathy.fraeman@evidera.com
Web: www.evidera.com

APPENDIX: SOURCE CODE OF GENERAL PROPENSITY SCORE MATCHING MACRO

The entire source code for the generalized propensity score matching macro is given below:

```
/***********************************************************************/
/* Program:       PSMatch_Multi.sas
/*
/* Platform:      SAS 9.4
/*
/* Author:        Kathy H. Fraeman
/*
/* Drug/Protocol: Generalized SAS Macro
/*
/* Description:   Does N:1 propensity score matching within specified
/*                absolute differences (radius) of propensity score
/*
/*                Can optimize on either number of matches, or closeness
/*                of matches, or no optimization
/***********************************************************************/

%macro psmatch_multi(subj_dsn =,    /* Data set with subject data  */
                     subj_idvar=,   /* Subject ID variable in
                                       &SUBJ_DSN  */
                     subj_psvar=,   /* Propensity Score variable in
                                       &SUBJ_DSN  */
                     cntl_dsn=,     /* Data set with control data  */
                     cntl_idvar=,   /* Control ID variable in data set
```

```
                          &CNTL_DSN  */
                cntl_psvar=,   /* Propensity Score variable in
                          &CNTL_DSN  */
                match_dsn=,    /* Output data set */
                match_ratio=,  /* Number of matches per subject  */
                score_diff=,   /* Maximum allowable absolute
                          differences between propensity
                          scores*/
                opt=none,      /* Type of matching optimization --
                          by number of matches( = num),
                          by closeness ( = close),
                          or default none ( = none) */
                seed=1234567890)  /*  Optional seed for random number
                              generator  */
                ;

/*********************************************************************/
/*  Delete final matched pairs dataset, if exists from a prior run
/*********************************************************************/
PROC DATASETS nolist;
     delete __final_matched_pairs;
     run;
     quit;

/**********************************************/
/*  Make all internal macro variables local
/**********************************************/
%local __dsid __varnum __cntl_type __rc __num;

/************************************************/
/* Control ID variable (numeric or character)
/************************************************/
%let __dsid      = %sysfunc(open(&cntl_dsn,i));
%let __varnum    = %sysfunc(varnum(&__dsid, &cntl_idvar));
%let __cntl_type = %sysfunc(vartype(&__dsid, &__varnum));
%let __rc        = %sysfunc(close(&__dsid));
%put &__cntl_type;

/***************************/
/*  Subject Matching Data
/***************************/
DATA __subjmatch (keep = &subj_idvar &subj_psvar);
     set &subj_dsn;
     run;

/***************************/
/*  Control Matching Data
/***************************/
DATA __contmatch (keep = &cntl_idvar &cntl_psvar);
     set &cntl_dsn;
     run;

/**************************************************************/
/*  Find all possible matches between subjects and controls
/*  Propensity scores must match within +/- &match (radius)
/**************************************************************/
PROC SQL;
```

```
     create table __matches0 as
        select
        s.&subj_idvar as subj_idvar,
        c.&cntl_idvar as cntl_idvar,
        s.&subj_psvar as subj_score,
        c.&cntl_psvar as cntl_score,
         abs(s.&subj_psvar - c.&cntl_psvar) as diff_score
       from __subjmatch s left join __contmatch c
       on abs(s.&subj_psvar - c.&cntl_psvar) <= &score_diff
       order by subj_idvar;
       quit;

/************************************/
/*  Data set of all possible matches
/************************************/
DATA __possible_matches;
     set __matches0;
     /*----------------------------------------*/
     /*  Create a random number for each match
     /*----------------------------------------*/
     call streaminit(&seed);
     rand_num = rand('uniform');

     /*----------------------------------------------*/
     /*  Remove subjects who had no possible matches
     /*----------------------------------------------*/
     %if &__cntl_type = C %then %do;
        if cntl_idvar ^= '';
     %end;
     %else %if &__cntl_type = N %then %do;
        if cntl_idvar ^= .;
     %end;

     /*-------------------------------------------*/
     /*  Round DIFF_SCORE to an order of magnitude
     /*-------------------------------------------*/
     %if &opt = close %then %do;
        if . < diff_score < .000000001 then
           sort_diff_score = .000000001;
        else if .000000001 <= diff_score < .00000001 then
           sort_diff_score = round(diff_score, .000000001);
        else if .00000001 <= diff_score < .0000001 then
           sort_diff_score = round(diff_score, .00000001);
        else if .0000001 <= diff_score < .000001 then
           sort_diff_score = round(diff_score, .0000001);
        else if .000001 <= diff_score < .00001 then
           sort_diff_score = round(diff_score, .000001);
        else if .00001 <= diff_score < .0001 then
           sort_diff_score = round(diff_score, .00001);
        else if .0001 <= diff_score < .001 then
           sort_diff_score = round(diff_score, .0001);
        else if .001 <= diff_score < .01 then
           sort_diff_score = round(diff_score, .001);
        else if .01 <= diff_score < .1 then
           sort_diff_score = round(diff_score, .01);
        else if diff_score >= .1 then
        sort_diff_score = round(diff_score, .1);
     %end;

     /*---------------------------*/
     /*  Create a dummy variable
     /*---------------------------*/
     n = 1;

     run;


/******************************************************************/
/*  Find the number of potential control matches for each subject
/******************************************************************/
PROC FREQ data=__possible_matches noprint;
     tables subj_idvar / out=__matchfreq (keep = subj_idvar count);
     run;

DATA __matches_freq;
     merge __possible_matches
           __matchfreq;
     by subj_idvar;
     /*------------------------------------------------------------*/
     /*  Only keep subjects with minimum number of possible matches
     /*------------------------------------------------------------*/
     if count >= &match_ratio;
     run;

PROC DATASETS nolist;
     delete __matches0;
     run;
     quit;

/*****************************************************************/
/*  Count the number of entries in the file of possible matches
/*****************************************************************/
%let __dsid = %sysfunc(open(__matches_freq,i));
%let __num = %sysfunc(attrn(&__dsid,nobs));
%let __rc = %sysfunc(close(&__dsid));

%do %while (&__num >= 1);

   PROC SORT data=__matches_freq;
        by %if &opt = num %then %do;
              count
           %end;
           %else %if &opt = close %then %do;
              sort_diff_score
           %end;
           rand_num subj_idvar;
        run;

   /************************************************************/
   /*  Get first randomly selected subject
   /*  For options, with either the least number of matches or
   /*   the closest match
   /************************************************************/
   DATA __first_subj_idvar (keep = subj_idvar);
```

```
     set __matches_freq;
     by n;
     if first.n;
     run;

/***************************************/
/*  Get all matches for that subject
/****************************************/
PROC SORT data=__matches_freq;
     by subj_idvar %if &opt = num %then %do;
                     count
                   %end;
                   %else %if &opt = close %then %do;
                      sort_diff_score
                   %end;
                   rand_num;
     run;

DATA __all_first_id;

     merge __matches_freq
           __first_subj_idvar (in=i);
     by subj_idvar;
     if i;
     num + 1;
     run;


DATA __new_matched_pairs (keep = subj_idvar cntl_idvar
                                 subj_score cntl_score);
     set __all_first_id;

     label subj_idvar =
          "Subject ID, original variable name &subj_idvar"
           cntl_idvar =
          "Matched Control ID, original variable name &cntl_idvar"
           subj_score =
         "Subject Propensity Score, original var name &subj_psvar"
           cntl_score =
          "Matched Control Propensity Score, orig var &cntl_psvar"
           ;
     if num <= &match_ratio;
     run;


/*******************************************/
/*  Remove subjects with matched controls
/*******************************************/
PROC SORT data=__new_matched_pairs (keep = subj_idvar)
          out=__new_matched_subj nodupkey;
     by subj_idvar;
     run;

DATA __match_remove_subj;
     merge __possible_matches
           __new_matched_subj (in=id);
     by subj_idvar;
     if ^id;
     run;

/*************************************************************/
/*  Remove all matched pairs that include selected controls
/*************************************************************/
PROC SORT data=__new_matched_pairs (keep = cntl_idvar)
          out=__remove_cont;
     by cntl_idvar;
     run;

PROC SORT data=__match_remove_subj;
     by cntl_idvar;
     run;

DATA __match_remove_cont;
     merge __match_remove_subj
           __remove_cont (in=id);
     by cntl_idvar;
     if ^id;
     run;

PROC SORT data=__match_remove_cont out=__possible_matches;
     by subj_idvar;
     run;

/*********************************************************/
/*  Add new matched pairs to set of final matched pairs
/*********************************************************/
PROC APPEND base=__final_matched_pairs data=__new_matched_pairs;
     run;

/****************************************************************/
/*  Find the number of potential control matches for each subject
/****************************************************************/
PROC FREQ data=__possible_matches noprint;
     tables subj_idvar / out=__matchfreq (keep = subj_idvar
                                                 count);
     run;


DATA __matches_freq;
     merge __possible_matches
           __matchfreq;
     by subj_idvar;
     /*---------------------------------------------------------*/
     /*  Only keep subjects with the minimum number of matches
     /*---------------------------------------------------------*/
     if count >= &match_ratio;
     run;

/*********************************************************/
/*  Determine number of remaining possible matched pairs
/*********************************************************/
%let __dsid = %sysfunc(open(__matches_freq,i));
%let __num = %sysfunc(attrn(&__dsid,nobs));
%let __rc = %sysfunc(close(&__dsid));
```

```
    %end;  /*  of " %do %while (&__num >= 1);  */


    /*****************************************************************/
    /*  Create final output data set with one observation for each
    /*  original subject.
    /*
    /*  Variable names in output data set are:
    /*    SUBJ_IDVAR, SUBJ_SCORE, CNTL_IDVAR, CNTL_SCORE
    /*
    /*  If no match for subject ID (SUBJ_IDVAR), then CNTL variables
    /*    (CNTL_IDVAR, CNTL_SCORE) are missing.
    /*****************************************************************/
    PROC SORT data=__final_matched_pairs;
         by subj_idvar subj_score;
         run;

    DATA __subjmatch_orig;
         set __subjmatch (rename= (&subj_idvar = subj_idvar
                                   &subj_psvar = subj_score));
         run;

    PROC SORT data=__subjmatch_orig;
         by subj_idvar;
         run;

    DATA &match_dsn (label="Final Matched Pairs for PS Matching");
         merge __final_matched_pairs
               __subjmatch_orig;
         by subj_idvar subj_score;
         run;

    /************************************************/
    /*  Delete all temporary datasets created by macro
    /************************************************/
    PROC DATASETS nolist;
         delete __contmatch __final_matched_pairs __matches_freq0
                __matches_freq __match_pair0 __matchfreq
                __match_remove_cont __match_remove_subj
                __new_matched_pairs __subjmatch __subjmatch_orig
                __possible_matches __remove_cont
                __first_subj_idvar __all_first_id
                __new_matched_subj;
      run;
      quit;

%mend psmatch_multi;
```