Paper 185-2007

Local and Global Optimal Propensity Score Matching

Marcelo Coca-Perraillon
Health Care Policy Department, Harvard Medical School, Boston, MA

ABSTRACT

Propensity score-matching methods are often used to control for bias in observational studies when randomization is not possible. This paper describes how to match samples using both local and global optimal matching algorithms. The paper includes macros to perform the nearest available neighbor, caliper, and radius matching methods with or without replacement and matching treated observations to one or many controls. The similarity between observations is evaluated using both the absolute value and the Mahalanobis distance that includes the propensity score along with other covariates. This paper also explains how to find a global optimal match with a variable number of controls using network flows. SAS® 9.1, SAS/STAT®, and SAS/OR® are required.

INTRODUCTION

The objective of randomization in statistics is to obtain groups that are comparable in terms of both observed and unobserved characteristics. When randomization is not possible, causal inference is complicated by the fact that a group that received a treatment or experienced an event maybe very different from another group that did not experience the event or receive the treatment. Thus, it is not clear whether a difference in certain outcome of interest is due to the treatment or is the product of prior differences among groups. One way of overcoming this problem is to adjust the estimates of treatment effect using the measured characteristics of each group as covariates in a model. Another way of establishing causality that has gained popularity in recent years is to select groups that are similar in terms of observed characteristics before making a comparison, which may still involve some type of model adjustment.

Propensity score methods were developed to facilitate the creation of comparison groups that are similar. “Similar” in this sense refers to the distribution of observed characteristics. Propensity score methods were created to be used in the design stage of an observational study, sometimes before the outcome information has been collected or not using the outcome information when it is available (Rubin 2007). The first step involves estimating the likelihood (the propensity score) that a person would have received the treatment given certain characteristics. More formally, the propensity score is the conditional probability of assignment to a particular treatment given a vector of observed covariates (D’Agostino 1998). If a treated person and a potential control unit have the same propensity score, then both units have the same distribution of the covariates that entered into the estimation of the propensity scores. Intuitively, comparing matched treated and untreated individuals with the same observable characteristics is like comparing the individuals in a randomized experiment. In prospective randomization, however, both observed and unobserved characteristics are guaranteed to be balanced.

Two key assumptions of propensity scores are that both the outcome of interest and the treatment assignment do not depend on unobservable characteristics. For example, if the treatment effect depended on the presence of a gene, matching individuals on a multitude of demographic characteristics could fail to create treatment and control groups with a similar proportion of the relevant gene. In the same way, if the treatment was only given to persons who volunteered because of the severity of their symptoms, then matching without explicitly taking into account the severity of the symptoms may fail to produce similar groups (Coca-Perraillon 2006).

After estimating the propensity scores, the scores are used to group observations that are close to each other. One way of accomplishing this is to classify treated and untreated observations into subgroups and then separately compare the outcome for each subgroup. This method is usually referred as subclassification on the propensity scores (Rosenbaum and Rubin 1984). The other way is to match one treated unit to one or more untreated controls, which is usually referred as matching on the propensity score (Rosenbaum and Rubin 1983).

This paper focuses on matching on the propensity score. Key in the implementation of matching using propensity scores is to decide what metric to use when evaluating the distance between scores (usually the absolute value or the Mahalanobis metric) and what type of algorithm to implement (local or global optimal).[1] The first part of this paper describes the most commonly used methods and introduces a SAS macro to implement them. The second part explains how to adapt the code to implement the Mahalanobis metric matching. Finally, the last part describes global optimum matching and provides code to implement it using SAS. This paper assumes that the propensity scores have been already estimated. See D’Agostino and Rubin (2000) and Rubin (2004) for an introduction on how to estimate the propensity scores.

1 See Gu and Rosenbaum (1993) and Smith and Todd (2005) for detailed descriptions of most of the matching methods currently available.

LOCAL OPTIMAL (GREEDY) ALGORITHMS

Local optimal algorithms, often called "greedy" algorithms, are methods that make optimal decisions at each step without attempting to make the best overall (global) decision. The following is an example of matching using a greedy algorithm: both treatment and control units are first randomly sorted. Then the first treated unit is selected to find its closest control match based on the absolute value of the difference between their propensity scores (or the logit of the scores).The closest control unit is selected as a match. This procedure is repeated for all the treated units.

This method guarantees that a match is always found for all the treated units even if the propensity scores are not close and provided there are enough controls available. In each iteration, the best (optimal) control is chosen, but this process does not guarantee that the total distance between propensity scores is minimized. For example, consider the following hypothetical datasets:

```
data Treatment;
  input pscoreT idT;
datalines;
0.10 1
0.20 2
;
data Control;
  input pscoreC idC;
datalines;
0.07 3
0.08 4
0.11 5
0.45 6
0.47 7
;
```

The variables pscoreT and pscoreC are the propensity scores for the treated and control units respectively, and idT and idC are their respective ids. The first iteration step considers idT= 1 and finds the propensity score in the dataset Control with the smallest (pscoreT – pscoreC) in absolute terms. In this case, idC= 5. The second iteration considers idT= 2 and finds that the best match is idC= 4. That is, the optimal matching is the set of pairs (idC, idT) = {(5,1), (4,2)}.

This method of matching is often referred as the nearest available neighbor matching and can be modified in several ways. For example, in "caliper matching," both treatment and controls units are randomly sorted and then the first treated unit is selected to find its closest control match in terms of the propensity score but only if the control's propensity score is within a certain distance (caliper). Thus, in this method, it is possible that a treated unit cannot be matched to a control. The objective is to avoid bad matches.

In addition, both the nearest available neighbor and caliper matching can be modified to include replacement. A control unit could be selected more than once. Matching with replacement minimizes the propensity score distance between the matched units since each treated unit is matched to the closest control, even if the control has been selected before. Another modification includes matching each treated unit to more than one control (1 to N) match. This can be done by creating N replicas of each treated unit and proceeding as described above. Finally, another variation is "radius matching" (Dehejia and Wahba 1999), in which all the control units within a certain distance are chosen.

The macro %PSMatching in Appendix A can be used to perform the nearest available neighbor, caliper, and radius matching with or without replacement and matching 1 to N controls. The details of the SAS implementation are described in Coca-Perraillon (2006). The main idea is to use one data step to iterate over the Treatment dataset and a SAS hash to search through the Control dataset. SAS hashes are implemented as part of the data step Component Object Interface. For practical purposes one can simply think of hashes as datasets that exist for the duration of the data step. They can be accessed and modified one or more times for each observation read in a data step.

The following code calls the macro:

```
%PSMatching(datatreatment=, datacontrol=, method=, numberofcontrols=, caliper=,
            replacement=);
```

The parameters datatreatment and datacontrol refer to the Treatment and Control datasets and they do not need to be sorted. The method parameter can be NN (nearest available neighbor), caliper or radius. Caliper can be any number indicating the size of the caliper and the parameter replacement takes the values of yes or no. All the parameters are case insensitive. If the method is NN, the caliper parameter is ignored.

For example, the following code performs a 1 to 2 nearest available neighbor matching without replacement:

```
%PSMatching(datatreatment= Treatment, datacontrol= Control, method= NN,
            numberofcontrols= 2, caliper=, replacement= no);
```

The output is a dataset called Matched which has two variables, IdSelectedControl and MatchedToTreatID. In this case, the optimal matching is the set {(5,2), (4,1), (3,1), (6,2)}. Note that each treated unit has been matched twice.

MAHALANOBIS METRIC MATCHING

In this type of matching, the definition of distance is changed. The similarity between the propensity score of treated and untreated units is evaluated using the multidimensional Mahalanobis metric matching:

$$D_{ij} = \sqrt{(x_i - y_j)^T S^{-1}(x_i - y_j)},$$

where S^{-1} is the pooled variance-covariance matrix and x and y are multivariate vectors. Note that if the variance-covariance matrix is an identity matrix the Mahalanobis metric is reduced to the familiar Euclidean metric. Usually the Mahalanobis metric matching includes the propensity score and other covariates that are considered to be important and are hoped to be balanced (Rosenbaum and Rubin 1985).

Implementing the Mahalanobis matching using SAS is complicated by the fact that at each step of the algorithm the distance involves matrix multiplication. Fortunately, one can instead transform the data using PROC PRINCOMP to make the variance-covariance matrix an identity matrix and then use the Euclidean metric in each iteration.

Consider for example the following datasets:

```
data Treatment;
  input pscoreT idT ageT group;
datalines;
0.10 1 25 1
0.20 2 22 1
;
data Control;
  input pscoreC idC ageC group;
datalines;
0.07 3 45 0
0.08 4 37 0
0.11 5 20 0
0.45 6 27 0
0.47 7 47 0
;
```

Note that there are two additional variables in these datasets. Besides the propensity scores and the ids, the datasets include a continuous variable called age (ageT and ageC for treatment and control respectively) and an indicator variable that takes the value of 1 if the observation is in the treated pool.

The first step to do a Mahalanobis matching is to create a combined dataset:

```
data _TreatControl;
   set Treatment(rename=(idT= id pscoreT= pscore ageT= age))
       Control(rename=(idC= id pscoreC= pscore ageC= age));
run;
```

After all the data are combined, PROC PRINCOMP is used to decompose the data:

```
proc princomp data= _TreatControl std out= _TreatControlDC;
   var pscore age;
run;
```

The output dataset, called _TreatControlDC, has two new variables, prin1 and prin2, which contain the transformed values of pscore and age, respectively. The datasets can be separated again by submitting the following code:

```
      data _Treatment0(rename=(prin1= pscoreT prin2= ageT id= idT))
           _Control0(rename=(prin1= pscoreC prin2= ageC id= idC));
         set _TreatControlDC;
         RandomNumber= ranuni(123456);
         if group= 1 then output _Treatment0;
         else if group= 0 then output _Control0;
      run;
```

The following code can be used to perform a 1 to 1 matching with replacement:

```
proc sort data= _Treatment0 out= _Treatment;
   by RandomNumber;
run;
proc sort data= _Control0 out= _Control;
   by RandomNumber;
run;

data MatchedMH(keep = IdSelectedControl MatchedToTreatID);
   length idC pscoreC ageC 8;
   if _N_= 1 then do;
   declare hash h(dataset: "_Control", ordered: 'no');
      declare hiter iter('h');
      h.defineKey('idC');
      h.defineData('pscoreC', 'idC', 'ageC');
      h.defineDone();
      call missing(pscoreC, idC, ageC);
   end;
   set _Treatment;
   retain BestDistance 99;
   rc=iter.first();
   if (rc=0) then BestDistance= 99;
   do while (rc = 0);
      * Euclidean distance;
      ScoreDistance = sqrt((pscoreT-pscoreC)**2 + (ageC-ageT)**2);
      if ScoreDistance < BestDistance then do;
         BestDistance = ScoreDistance;
        IdSelectedControl = idC;
        MatchedToTreatID = idT;
      end;
      rc = iter.next();
      if (rc ~= 0) then do;
         output;
         rc1 = h.remove(key: IdSelectedControl);
      end;
    end;
run;
```

Note that one of the challenges to create a macro for the Mahalanobis metric matching is that the number of variables to include in the distance may change. In the previous example, there is only one additional variable (age) that enters into the calculation of the Mahalanobis distance, but in practice there may be many more. The methods described in Carpenter (2004, Chapter 9) could be used to create dynamic macro code that accommodates a varying number of variables. Also note that the code is similar to that of the macro %PSMatching and can be modified to include 1 to N matching, matching with replacement, and calipers in the same way as in the %PSMatching macro.

GLOBAL OPTIMAL ALGORITHMS

The matching methods described above match each treated unit to their best control(s), but they do not minimize the overall distance between propensity scores. For example, consider again the following datasets:

```
      data Treatment;
        input pscoreT idT;
      datalines;
      0.10 1
      0.20 2
      ;
```

```
data Control;
  input pscoreC idC;
datalines;
0.07 3
0.08 4
0.11 5
0.45 6
0.47 7
;
```

A nearest neighbor, one-to-one match creates the matches (idC, idT) = {(5,1), (4,2)}. The first match distance is |0.10-0.11| = 0.01 and the second is |0.20-0.08|= 0.12. Therefore, the total distance is 0.13. However, this is not the shortest distance. If the matches are reversed; that is, if the optimal matched set is (idC*, idT*) = {(4,1), (5,2)}, the total distance is 0.11. The reason is that the greedy algorithms do not reconsider a match once it is done. When idT= 2 was evaluated, the algorithm did not consider the possibility of undoing the match (5,1) in favor of (5,2) since that would have freed idC= 4 to be matched with idT= 2 with a net gain of 0.02.

Rosenbaum (1989) introduced the idea of optimal matching by borrowing from the vast literature on network flows, a set of problems from operations research. Basically, the matching problem is recast in terms of graph theory. Treated and potential controls are represented as nodes of a graph. The distance between treated units and controls is represented as the cost of going from one node to the other and the matching problem is reduced to finding the path that minimizes the total distance. In another paper, Ming and Rosenbaum (2001) showed that the matching problem is a special case of the assignment problem, which can be easily solved using PROC ASSIGN.

PROC ASSIGN requires as input a matrix of distances between all treated units and potential controls. Once again, SAS hashes can be used to quickly create a matrix of distances with the following code:

```
proc sort data= Control out= _Control0;
  by idC;
run;
data _Control;
   set _Control0;
   DistCol = compress('d' || _N_);
run;

data DistMatrix (keep= idT d1-d5);
   length pscoreC 8;
   length idC 8;
   if _N_= 1 then do;
   declare hash h(dataset: "_Control", ordered: 'ascending');
      declare hiter iter('h');
      h.defineKey('idC');
      h.defineData('pscoreC', 'idC');
      h.defineDone();
      call missing(idC, pscoreC);
   end;

   set Treatment;
   array dis(*) d1-d5;
   rc=iter.first();
   if (rc=0) then i=1;
   do while (rc = 0);
      dis(i)= abs(pscoreT - pscoreC);
      i+1;
      rc = iter.next();
      if (rc~=0) then output;
   end;
run;
```

Then PROC ASSIGN is used to find the best matches:

```
proc assign data=DistMatrix out= Result;
  cost d1-d5;
  id idT;
run;
```

```
      proc print data= result;
         sum _fcost_;
      run;
```

Note that the variable DistCol in the dataset _Control is not part of the distance matrix but is used to identify the variables d1, d2, etc with the control observations they represent. The following code can be used to create a dataset called MatchedOpt that has the result of the matches:

```
      proc sql;
         create table MatchedOpt as
         select b.idC as IDSelectedControl, a.idT as IDTreatment
         from result a left join _Control b
         on a._ASSIGN_ = b.DistCol;
      quit;
```

As expected, PROC ASSIGN finds the match set (idC*, idT*) = {(4,1), (5,2)}, which minimizes the total distance. See Ming and Rosenbaum (2001) for a clear guide on how to set up the assignment problem in order to obtain an optimal match with a variable number of controls.

CONCLUSIONS

Propensity score methods are increasing in popularity and the introduction of SAS hashes has made matching simpler to implement. Local optimal methods may not provide the best possible matches in a global sense, but they are fast and feasible even when using large datasets. Global methods may be difficult to implement when there are large numbers of potential controls to choose from since it requires the creation of a large distance matrix.

REFERENCES

Carpenter, A. (2004), Carpenter's Complete Guide to the SAS Macro Language, SAS Press.

Coca-Perraillon, M (2006), Matching with Propensity Scores to Reduce Bias in Observational Studies, *Proceedings of the NorthEast SAS Users Group Conference (NESUG)*, Philadelphia, PA.

D'Agostino, R.H. (1998), Propensity Score Methods for Bias Reduction in the Comparison of a Treatment to a Non-randomized Control Group, *Statistics in Medicine*, 17, 2265-2281.

D'Agostino, R.B., and Rubin, D.B (2000), Estimating and Using Propensity Scores with Partially Missing Data, *Journal of the American Statistical Association*, 95, No. 451, 749-759

Dehejia, R. H. and S. Wahba (1999), Causal Effects in Nonexperimental Studies: Reevaluating the Evaluation of Training Programs, *Journal of the American Statistical Association*, 94(448), 1053-1062.

Gu, X.S. and Rosenbaum, P.R. (1993), Comparison of Multivariate Matching Methods: Structures, Distances, and Algorithms, *Journal of Computational and Graphical Statistics*, Vol. 2, No. 4, 405-420.

Ming, K. and Rosenbaum P.R. (2001), A Note on Optimal Matching with Variable Controls Using the Assignment Algorithm, Journal of Computational and Graphical Statistics, 10, No. 3, 455-463.

Rubin, D.B (2004), On Principles for Modeling Propensity Scores in Medical Research, *Pharmacoepidemiology and Drug Safety*, 13, 855-857.

Rubin, D.B. (2007), The Design versus the Analysis of Observational Studies for Causal Effects: Parallels with the Design of Randomized Trials, Statistics in Medicine, 26, 20-36.

Rosenbaum, P.R., and Rubin, D.R. (1983), The Central Role of the Propensity Score in Observational Studies for Causal Effects, *Biometrika*, 70, No. 1, 41-55.

Rosenbaum, P.R., and Rubin, D.R. (1984), Reducing Bias in Observational Studies Using Subclassification on the Propensity Score, *Journal of the American Statistical Association*, 79, No. 387, 516-524.

Rosenbaum, P.R., and Rubin, D.R. (1985), Constructing a Control Group Using Multivariate Matched Sampling Methods That Incorporate the Propensity Score, *The American Statistician*, Vol. 39, No. 1, 3338.

Rosenbaum, P.R. (1989), Optimal Matching for Observational Studies, *Journal of the American Statistical Association*, 84, No. 408, 1024-1032.

Smith, J., and P. Todd (2005). Does Matching Overcome LaLonde's Critique of Nonexperimental Estimators?, *Journal of Econometrics*, 125(1-2), 305-353.

CONTACT INFORMATION

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

Marcelo Coca Perraillon
Health Care Policy Department
Harvard Medical School
180 Longwood Ave.
Boston, MA 02115
Marcelo_Coca@hms.harvard.edu

APPENDIX A. %PSMatching macro

```
%macro PSMatching(datatreatment=, datacontrol=, method=, numberofcontrols=, caliper=,
replacement=);
/* Create copies of the treated units if N > 1 */;
data _Treatment0(drop= i);
   set Treatment;
   do i= 1 to &numberofcontrols;
      RandomNumber= ranuni(12345);
      output;
   end;
run;

/* Randomly sort both datasets */
proc sort data= _Treatment0 out= _Treatment(drop= RandomNumber);
   by RandomNumber;
run;
data _Control0;
   set Control;
   RandomNumber= ranuni(45678);
run;
proc sort data= _Control0 out= _Control(drop= RandomNumber);
   by RandomNumber;
run;

data Matched(keep = IdSelectedControl MatchedToTreatID);
   length pscoreC 8;
   length idC 8;
   /* Load Control dataset into the hash object */
   if _N_= 1 then do;
      declare hash h(dataset: "_Control", ordered: 'no');
      declare hiter iter('h');
      h.defineKey('idC');
      h.defineData('pscoreC', 'idC');
      h.defineDone();
      call missing(idC, pscoreC);
   end;
   /* Open the treatment */
   set _Treatment;
   %if %upcase(&method) ~= RADIUS %then %do;
   retain BestDistance 99;
   %end;
   /* Iterate over the hash */
   rc= iter.first();
   if (rc=0) then BestDistance= 99;
   do while (rc = 0);
       /* Caliper */
   %if %upcase(&method) = CALIPER %then %do;
      if (pscoreT - &caliper) <= pscoreC <= (pscoreT + &caliper) then do;
         ScoreDistance = abs(pscoreT - pscoreC);
         if ScoreDistance < BestDistance then do;
            BestDistance = ScoreDistance;
            IdSelectedControl = idC;
            MatchedToTreatID = idT;
         end;
      end;
   %end;
   /* NN  */
   %if %upcase(&method) = NN %then %do;
      ScoreDistance = abs(pscoreT - pscoreC);
      if ScoreDistance < BestDistance then do;
         BestDistance = ScoreDistance;
        IdSelectedControl = idC;
        MatchedToTreatID = idT;
      end;
   %end;
   %if %upcase(&method) = NN or %upcase(&method) = CALIPER %then %do;
      rc = iter.next();
      /* Output the best control and remove it */
      if (rc ~= 0) and BestDistance ~=99 then do;
       output;
   %if %upcase(&replacement) = NO %then %do;
      rc1 = h.remove(key: IdSelectedControl);
   %end;
      end;
   %end;
   /* Radius */
%if %upcase(&method) = RADIUS %then %do;
   if (pscoreT - &caliper) <= pscoreC <= (pscoreT + &caliper) then do;
      IdSelectedControl = idC;
      MatchedToTreatID = idT;
      output;
   end;
   rc = iter.next();
%end;
end;
run;
/* Delete temporary tables. Quote for debugging */
proc datasets;
   delete _:(gennum=all);
run;
%mend PSMatching;
```