Matching with Propensity Scores to Reduce Bias in Observational Studies

Marcelo Coca-Perraillon, Adheris Inc., Burlington, MA

ABSTRACT

In observational studies subjects are not randomly assigned to treatment and control groups. Therefore, both groups can be significantly different from each other and their differences after applying a treatment cannot be attributed to the treatment effect. Propensity scores methods offer a way to balance groups by matching treatment and control units based on a set of covariates. This paper explains how to use SAS® to match samples employing the most commonly used matching methods, such as nearest available neighbor, calipers and radius with- and without replacement. This paper also serves as an introduction to propensity score matching methods and illustrates both the relevance and the inherent problems of any method that attempts to select a valid comparison group.

INTRODUCTION

The objective of randomization in experiments is to obtain two groups of individuals that are, at baseline and on average, identical in terms of observed and unobserved characteristics. Therefore, any average difference between the groups after applying a treatment can only be attributed to the treatment effect. In observational studies, by definition, there is no prior randomization and the group that received the treatment can be substantially different from the group that did not receive the treatment.

An obvious way to overcome this problem is to compare the group that received the treatment to a similar group. If there were only one characteristic that determined the participation in the study or influenced the outcome, like age or gender, selecting a suitable comparison group would be straightforward. However, when there are several variables to consider simultaneously, the problem of choosing controls becomes more complex.

Propensity scores were developed to overcome this difficulty. The idea is to estimate the likelihood (the propensity score) that a person would have participated in the experiment 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 the difference between the treatment and control outcome means is an unbiased estimator of the treatment effect. Intuitively, comparing matched treated and untreated individuals with the same observable characteristics is like comparing the individuals in a randomized experiment.

The key assumptions of propensity scores are that both the outcome of interest and the treatment assignment do not depend on unobservable characteristics. If, for example, the effect of the treatment depended on the presence of a particular gene, matching individuals on gender, age or a multitude of other observable covariates could fail to create treatment and control groups with a similar proportion of the relevant gene. In a similar 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 could also fail to produce comparable groups.

Implementing propensity scores matching can be a difficult task given the multitude of methods proposed over the years. The objective of this paper is to provide an introduction to some of the most commonly used methods and to explain how to implement them using SAS. The first part of this paper provides an overview of the estimation of propensity scores and matching algorithms. The second part describes how to use SAS to match a hypothetical group of treatment and controls units, while the final part applies some of the algorithms to a dataset that has been extensively used to show both the usefulness and the problems of matching.

ESTIMATING THE PROPENSITY SCORE

The first step to perform propensity score matching consists of estimating the propensity score for a particular individual. Since the propensity score is defined as the conditional probability of assignment to a particular treatment, in most applications it is estimated using a logistic (logit) regression in which the dependent variable is a dummy variable indicating treatment participation. In principle, however, other estimation techniques can be used, such a random effects models, models that include instrumental variables, or discriminant analysis. The independent variables are all the observable factors that are considered to be related to the treatment inclusion and outcome.

The strategy of estimating the likelihood of participating in a treatment appears to be counterintuitive since in practical application it is already known which individuals received the treatment and thus there is no need to model the participation. However, the objective is to estimate how likely was a person to receive the treatment given certain characteristics. Another way to understand this strategy is to think of propensity scores as a way of profiling the individuals who participated so similar individuals can be selected as controls. If the independent variables include all the factors related to participation and outcome, then the treatment effect after matching would be the same as if the observational study had been a random experiment. The major drawback of matching with propensity scores is that in practical situations it is not always possible to include all the variables related to participation and outcome. Randomization avoids this problem since it does not rely on knowing this information in

order to achieve balanced groups.

MATCHING METHODS

This section explains in detail two of the most commonly used matching methods along with their variants: the nearest available neighbor and caliper matching. One of the difficulties in implementing propensity score matching is that there are many methods described in the literature[1] and no definitive way to choose among them (most of the recommendations are based on theory or rely on simulations). All the methods, however, share some common elements. They must use an operational definition of similarity (distance) between propensity scores and covariates and must decide on the number of controls to be matched to each treated unit. A related question is whether the control units can be chosen more than once (with- or without replacement). All the methods implicitly assume that there are more control units than treated units and hence the problem is one of choosing one or more controls from a set of competing alternatives.

NEAREST AVAILABLE NEIGHBOR

In this method, both treatment and control units are first randomly sorted. Then the first treatment unit is selected to find its closest control match based on the absolute value of the difference between the propensity score (or the logit of the propensity score) of the selected treatment and that of the control under consideration. 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.

CALIPER MATCHING

This method is similar to the nearest available neighbor matching method but it adds an additional restriction. 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 (or the logit of the propensity score) but only if the control's propensity score is within a certain radius (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.

A variation of caliper matching is the nearest available Mahalanobis metric matching within calipers defined by the propensity score. The difference with caliper matching is that the definition of distance between the propensity score of treated and untreated units incorporates the propensity score and a set of covariates (using the multidimensional Mahalanobis distance) and that the calipers are defined by the propensity score (see D'Agostino 1998 and Rubin 1980 for further details).

Both the nearest available neighbor and caliper matching can be modified in at least three ways. First, the control unit could be replaced once it is matched to a treatment unit. Thus, 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. Second, each treated unit could be matched to an arbitrary number of controls. Third, before performing the matching, a common support region can be defined. The common support region excludes treated units whose propensity score is higher than the highest propensity score of the control units and control units whose propensity scores are lower than the lowest propensity score of the treated units (see Smith and Todd 2005 for additional definitions of the common support region). Note that if a common support region is established, the nearest available neighbor matching method could fail to find a match for some of the treated units.

RADIUS MATCHING

Another variant of caliper matching is radius matching, used by Dehejia and Wahba (2002). The radius method is similar to caliper matching but removes a restriction. Instead of matching a treated unit with its closest control within a caliper, all the units that fall within the caliper or radius are selected.

LOCAL VERSUS GLOBAL OPTIMUM

As described above, all the matching methods are examples of algorithms that choose a local optimum (often called "greedy" algorithms). That is, for each treated unit, the best possible control is selected and once a decision is made it is not considered again. This way of matching, however, does not guarantee that the global distance between matches is minimized. Furthermore, the order in which the observations appear can change the final match; thus the importance of randomly ordering the observations before performing the match. An alternative is to employ an algorithm that matches observations in a global optimum manner, such as the minimum cost flow in a network. Global optimum methods are described in Rosenbaum (2002, p. 212) and Bertsekas (1998, Chapter 4). This paper only considers local optimum algorithms.

1 See Gu and Rosenbaum (1993), Rosenbaum (2002) and Smith and Todd (2005) for detailed descriptions of most of the matching methods currently available.

SAS IMPLEMENTATION

The following section describes how to implement the nearest available neighbor and the caliper matching methods along with their common variants using SAS. The nearest available neighbor with replacement is considered first since its implementation is relatively easy with common techniques. However, an efficiently implementation of propensity score methods in SAS requires the use of data step hashes, which are also described.

A commonly used SAS macro was previously developed by Parsons (2004) to perform a 1 to N matching. As implemented, the macro is a type of caliper and nearest available neighbor as described above, although it has several deviations from the standard algorithms that make it difficult to place it into one particular category. In the macro, the propensity scores of both treated and untreated units are rounded to the first seven decimal digits[2] and the treated and control units are sorted in ascending order by the propensity score (not the rounded propensity score). If the treated units have the same propensity score, they are randomly ordered. Then the first treated unit is selected and (for a one-to-one match) matched to the control unit with the same rounded propensity score. The process is repeated for all the treated units. In successive passes, the unmatched units are matched rounding the propensity scores to a reduced number of digits until only one decimal digit is considered. Note that rounding the propensity score is equivalent to establishing a caliper around the score; successive passes reduce the size of the caliper.

There are at least two problems with this implementation. First, it imposes a caliper to both the treated and potential controls propensity scores by rounding them. In caliper matching, a treated unit is matched to a control unit that is within the caliper, but there is no need to impose a restriction to the propensity score of the treated units since a good match could be missed. Thus, in this sense, the algorithm does not perform a local optimum match.[3] Most importantly, however, the matching is actually not random. To choose the best match, the macro relies on the treated and control units being sorted by the propensity score (see Parsons 2004, p.7). Therefore, treated units with lower propensity scores are more likely to both be matched and have better matches. As described earlier, the treated and control units should be randomly ordered.

DATASETS

To describe the SAS code used to implement the matching methods, two hypothetical datasets are used. The datasets are small enough that the matching can be performed with paper and pen. It is assumed that the propensity score has been estimated and that the observations are randomly ordered.

The following code creates the datasets Treatment and Control:

```
data Treatment;
  input pscoreT idT;
datalines;
0.110 1
0.130 3
0.110 2
;
data Control;
  input pscoreC idC;
datalines;
0.334 8
0.110 5
0.131 4
0.107 7
0.130 2
;
```

Both datasets have only two variables: the estimated propensity score and an id variable identifying an observation.

NEAREST NEIGHBOR WITH REPLACEMENT - TWO SET STATEMENTS

This method selects the first treated unit (idT=1) and then matches it to its closest control based on the absolute value of the difference between their propensity scores. Thus, in the first iteration, idT=1 is matched to idC=5. Since replacement is allowed, the observation idC=5 can be matched again. In the next iteration, the second treatment unit (idT=3) is selected until all three have been considered.

---

[2] The macro describes the first matching as an 8-digit match. However, the match is performed on the first seven decimal digits of the propensity score since the SAS function round() with a parameter of $1xE^{-7}$ is used. See Parsons (2004, p. 9).

[3] For example, a treated unit with a propensity score of 0.11111114 would not be matched to a control unit with a propensity score of 0.11111115 because the rounded propensity score of the treated unit is 0.1111111 while the rounded propensity score of the control unit is 0.1111112. A second pass could perform the match, but the first pass missed the best local match.

The following code performs the match:

```
* Nearest neighbor with replacement using two set statements;
1 data Matched(keep= IdSelectedControl MatchedToTreatID);
2   set Treatment;
3      do i= 1 to Ncontrol;
4         set Control point= i nobs= Ncontrol;
5         retain BestDistance IdSelectedControl MatchedToTreatID;
6         ScoreDistance = abs(pscoreT - pscoreC);
7         if i= 1 then BestDistance= 99;
8         if ScoreDistance < BestDistance then do;
9            BestDistance= ScoreDistance;
10           IdSelectedControl= idC;
11           MatchedToTreatID= idT;
12        end;
13        if i= NControl then output;
14     end;
15 run;
```

The set statement in line 2 starts reading the first observation of the Treatment dataset and the do loop that starts in line 3 and ends in line 14 reads the Control dataset from start to end. The end of the dataset is determined by the variable Ncontrol, which contains the number of observations in the Control dataset and is used to output the data in line 13. The retain statement is used to keep track of the best distance between the propensity scores. If a better match is found (condition in line 8), then the variable BestDistance is updated with the current value of ScoreDistance, defined in line 6. Note that this procedure does not rely on the datasets being sorted by the propensity scores and therefore the entire Control dataset has to be read for each treated unit under consideration. The final table is called Matched and contains only two variables: the id of the selected control (IdSelectedControl) and its respective match (MatchedToTreatID).

In this particular example, idT=1 is matched to idC=5, idT=3 to idC=2, and idT=2 to idC=5. Since replacement is allowed, idC=5 is matched twice.

NEAREST AVAILABLE NEIGHBOR WITH REPLACEMENT USING A HASH

The nearest available neighbor can also be accomplished using a hash (the advantage of a hash will become apparent in the case of matching without replacement). In SAS, hashes are implemented as part of the data step Component Object Interface. They allow users to efficiently store, search and retrieve data based on lookup keys, which can be data step variables. For practical purposes, one can simply think of hashes as datasets that exist for the duration of the data step and that can be iterated and modified. Perhaps the biggest challenge of using hashes in SAS is that their syntax is rather convoluted and unrelated to the data step syntax.

The following code matches treatment and control units using the nearest available neighbor method with replacement:

```
* Nearest neighbor with replacement using a hash;
1 data Matched(keep= IdSelectedControl MatchedToTreatID);
2   length pscoreC 8;
3   length idC 8;
4   if _N_= 1 then do;
5      declare hash h(dataset: "Control", ordered: 'no');
6      declare hiter iter('h');
7      h.defineKey('idC');
8      h.defineData('pscoreC', 'idC');
9      h.defineDone();
10     call missing(idC, pscoreC);
11  end;
12  set Treatment;
13  retain BestDistance 99;
14  rc= iter.first();
15  if (rc= 0) then BestDistance= 99;
16  do while (rc= 0);
17     ScoreDistance= abs(pscoreT - pscoreC);
18     if ScoreDistance < BestDistance then do;
19        BestDistance= ScoreDistance;
20        IdSelectedControl= idC;
21        MatchedToTreatID= idT;
22     end;
```

```
23     rc= iter.next();
24     if (rc~= 0) then output;
25  end;
26 run;
```

This code is similar to using two set statements. The difference is that instead of using a second set statement to iterate over the Control dataset, the Control dataset is stored in a hash and then the hash is iterated. The syntax that is part of the definition of the hash object starts in line 2 and ends in line 11. The hash called "h" is created (declared) in line 5. Line 5 specifies that the hash h contains the Control dataset. Besides "declaring" a hash, the hash iterator object needs to be explicitly declared, which is accomplished in line 6. The iterator is called "iter." Line 7 and 8 explicitly define the key and the data. The key in this case is the variable idC since it will be used to identify the control observations, and the hash data (defined in line 8) is made of the variables pscoreC and idC. Note that the key idC is also part of the data since the contents of the variable idC are used not only as identifiers (in line 20).

After declaring the hash, line 12 starts reading the Treatment dataset. The retain statement in line 13 is used to save the value of the best distance. In line 14, the variable rc equals 0 for the first observation of the hash h and it is used in line 15 to set the BestDistance variable to 99. The loop that iterates the hash h starts in line 16 and ends in line 24. The idea is to iterate the hash h in search of the best distance between the propensity scores. After all the observations in the Control dataset have been considered, the match information is saved by exporting the variables IdSelectedControl and MatchedToTreatID in line 24 by using the output statement. In other words, the information in the hash is passed to the dataset Matched.

NEAREST AVAILABLE NEIGHBOR WITHOUT REPLACEMENT

In the nearest available neighbor without replacement method, matched controls are not considered again once they have been matched. This is difficult to accomplish using two set statements since one cannot depend on the observations being in a particular order; otherwise, a pointer (option point in the set statement) could be used to start reading the data after a particular observation.

The following code performs the match using a hash:

```
* Nearest neighbor without replacement;
1 data Matched(keep= IdSelectedControl MatchedToTreatID);
2    length pscoreC 8;
3    length idC 8;
4     if _N_= 1 then do;
5        declare hash h(dataset: "Control", ordered: 'no');
6        declare hiter iter('h');
7        h.defineKey('idC');
8        h.defineData('pscoreC', 'idC');
9        h.defineDone();
10       call missing(idC, pscoreC);
11    end;
12    set Treatment;
13    retain BestDistance 99;
14    rc= iter.first();
15    if (rc=0) then BestDistance= 99;
16    do while (rc= 0);
17       ScoreDistance= abs(pscoreT - pscoreC);
18       if ScoreDistance < BestDistance then do;
19          BestDistance= ScoreDistance;
20          IdSelectedControl= idC;
21          MatchedToTreatID= idT;
22       end;
23       rc= iter.next();
24       if (rc~= 0) then do;
25          output;
26          rc1= h.remove(key: IdSelectedControl);
27       end;
28    end;
29 run;
```

The only difference between the above code and the case without replacement is the addition of line 26. The method "remove" is used to remove the observation identified by the key IdSelectedControl. When the next treated unit is read, the control unit matched previously is not part of the hash h.

CALIPER MATCHING WITH REPLACEMENT

The implementation of this method only requires the addition of two extra constraints. To avoid a bad match, only observations that are within a radius of the treated unit's propensity score are considered and the closest match is selected. For example, with a caliper of $1x10E^{-3}$ and for the case of idT=1, only controls with propensity scores between 0.109 and 0.111 would be

considered.
The following code performs the match.

```
* Caliper with replacement;
1 data Matched(keep= IdSelectedControl MatchedToTreatID);
2   length pscoreC 8;
3   length idC 8;
4   if _N_= 1 then do;
5      declare hash h(dataset: "Control", ordered: 'no');
6      declare hiter iter('h');
7      h.defineKey('idC');
8      h.defineData('pscoreC', 'idC');
9      h.defineDone();
10      call missing(idC, pscoreC);
11  end;
12  set Treatment;
13  retain BestDistance 99;
14  rc=iter.first();
15  if (rc=0) then BestDistance= 99;
16  do while (rc= 0);
17     if (pscoreT - 0.001) <= pscoreC <= (pscoreT + 0.001) then do;
18        ScoreDistance= abs(pscoreT - pscoreC);
19        if ScoreDistance < BestDistance then do;
20           BestDistance= ScoreDistance;
21           IdSelectedControl= idC;
22           MatchedToTreatID= idT;
23        end;
24     end;
25     rc = iter.next();
26     if (rc ~= 0) and BestDistance~= 99 then do;
27        output;
28     end;
29   end;
30 run;
```

The extra restriction that only considers propensity scores within the caliper is added in line 17 and closed in line 24. The other restriction is used to only output a match. With caliper matching, it is possible that some treated units do not have a match. The condition in line 26 states that the Matched dataset will only contain observations with a distance better than 99. Since the propensity score (or the logit of the propensity score) is less than 99, this restriction ensures that the dataset Matched does not contain unmatched treated units.

CALIPER MATCHING WITHOUT REPLACEMENT

As in the nearest available neighbor without replacement, the caliper matching without replacement code only requires an additional line that removes a matched control unit from the hash. To perform a caliper matching without replacement, the code used to perform matching with replacement simply requires an additional statement after line 27:

```
rc1= h.remove(key: IdSelectedControl)
```

RADIUS MATCHING

In terms of its implementation, radius matching is a simpler case of caliper matching since it removes the search of the best control match. It simply requires that the control units' propensity scores are within a radius. With a caliper of $1xE^{-3}$, for example, idT=3 is matched to both idC=4 and idC=2 since both are within 0.129 and 0.131.

The following code implements the match:

```
1 data Matched (keep = IdSelectedControl MatchedToTreatID);
2   length pscoreC 8;
3   length idC 8;
4   if _N_= 1 then do;
5      declare hash h(dataset: "Control", ordered: 'no');
6      declare hiter iter('h');
7      h.defineKey('idC');
8      h.defineData('pscoreC', 'idC');
9      h.defineDone();
10     call missing(idC, pscoreC);
11  end;
12  set Treatment;
13  rc=iter.first();
14  do while (rc = 0);
```

```
15     if (pscoreT - 0.001) <= pscoreC <= (pscoreT + 0.001) then do;
16        IdSelectedControl = idC;
17        MatchedToTreatID = idT;
18        output;
19    end;
20    rc = iter.next();
21  end;
22 run;
```

To perform a radius matching without replacement, the statement rc1= h.remove(key: IdSelectedControl) should be added after line 18. However, radius matching is usually performed with replacement (see Dehejia and Wahba 2002 for details).

MATCHING 1 to N

The codes used to implement the different matching methods perform a one-to-one match, but they can be adapted to perform a 1 to N match. Although not very efficient, an easy way to accomplish this is to iterate the hash N times. In the first pass, the first matched control is removed and successive passes select the next best match until each treated unit is matched to N controls.

APPLIED EXAMPLE

This section applies some of the algorithms described earlier to a well-known dataset. In an influential paper, LaLonde (1986) evaluated the effectiveness of nonexperimental estimators to asses the performance of job training programs on future income. He used as a benchmark data from the National Supported Work Demonstration, a social experiment in which a group of participants were randomized to either receive employment (as a form of on-the-job training) and counseling or serve as controls. The nonexperimental controls were selected from two different national surveys: the Current Population Survey (CPS) and the Panel Study of Income Dynamics (PSID). After estimating the effects of the program using a variety of econometrics techniques, LaLonde concluded that observational estimators produced inconsistent results that were very different from the experimental benchmark.

LaLonde, however, did not consider matching to select a sample from the nonexperimental controls. Using the same data, Dehejia and Wahba (1999, 2002) showed that matching with propensity scores methods could substantially reduce the nonexperimental bias. Their results, however, sparked a long controversy that is well summarized in Smith and Todd (2005) and Dehejia (2005). Smith and Todd strongly argued that Dehejia and Wahba's results depended on the particular sample analyzed and on the specification of the model used to estimate the propensity score. More importantly, the data lacked information on covariates that were relevant to the participation and outcome of the training program. For example, the controls were not matched based on geographic region; therefore, the participants and the controls were from different labor markets.

Table 1 shows the baseline characteristics of a subset analyzed by Dehejia and Wahba's (1999) (see the Reference section for a link to download the data). The means and standard deviations correspond to rows 3 and 7 of Table 1 in Dehejia and Wahba (1999). It is clear from Table 1 that the treatment group (N=185) is very different from the CPS data (N=15,992) in all variables except the proportion of Hispanics. Most in the group that received on-the-job training were black, single and had no formal education. As expected, the earnings in 1974 were significantly lower for the group that participated in the training program. The outcome variable is income in 1975, which was $1,532 for the treated group and $15,650 for the CPS controls. The benchmark difference from LaLonde's (1986) was about $1,794 in favor of the treated group.

Table 1. Baseline characteristics

Characteristic	Treatment (N=185)		Control (N=15,992)		Analysis	
	Mean	SD	Mean	SD	Test	p-value
Age (years)	25.82	7.16	33.23	11.05	t=13.89	<0.0001
Education (years)	10.35	2.01	12.03	2.87	t=11.24	<0.0001
Earnings 1974 (U.S. $)	2095.57	4886.62	14016.80	9569.80	t=32.47	<0.0001
	N	%	N	%		
Race (black)	156	84.32	1176	7.35	X=14334	<0.0001
Hispanic	11	5.95	1152	7.20	X =0.434	0.5300
Married	35	18.92	11382	71.17	X=240.47	<0.001
No Degree	131	70.81	4731	29.58	X=147.87	<0.0001

ESTIMATING THE PROPENSITY SCORE

The propensity score is estimated using PROC LOGISTIC. Besides the covariates in Table 1, the dataset SampleData contains a treatment indicator variable called treat, which equals to 1 if the observation was part of the training program. The following code estimates the propensity scores:

```
* Estimate the propensity score;
proc logistic data= SampleData;
   model treat= age age2 edu edu2 nodegree married race hisp
                ear74 edu*ear74 / risklimits;
   output out= DataPScore prob= pscore;
run;
```

The estimated propensity score is saved in the variable pscore and it is saved, along with all the covariates, in the dataset called DataPscore. Note that the model includes a quadratic term for age and earnings in 1974. This model is slightly different from the model specification used by Dehejia and Wahba (1999, 2002).

Instead of adapting the matching code described earlier, it is easier to modify the dataset DataPScore. In addition, following Dehejia and Wahba (1999), the matching is done based on the logit of the propensity score. The following code creates two datasets that follow the format of the two hypothetical datasets Treatment and Control used to described the matching algorithms:

```
* Create two tables;
data Treatment0(rename=(Lpscore=pscoreT id=idT))
     Control0(rename=(Lpscore=pscoreC id=idC));
   set DataPScore;
   Lpscore = log(pscore/(1-pscore));
   RandomNumber= ranuni(2006);
   if treat =1 then output Treatment0;
   else if treat= 0 then output Control0;
run;

* Randomly sort the tables;
proc sort data= Treatment0 out= Treatment(keep=pscoreT idT);
   by RandomNumber;
run;

proc sort data= Control0 out= Control(keep=pscoreC idC);
   by RandomNumber;
run;
```

The match can be performed by submitting the code of any of the matching methods described earlier. The final output is a dataset called Matched that contains only two variables: IdSelectedControl and MatchedToTreatId. To analyze the matched samples, it is convenient to merge them together adding all the variables. This can be accomplished by submitting the following code:

```
* Selected matched controls;
proc sql;
   create table MatchedControls as
   select *
   from control0
   where idC in (select IdSelectedControl from matched);
quit;

* Selected matched programs;
proc sql;
   create table MatchedTreatment as
   select *
   from treatment0
   where idT in (select MatchedToTreatId from matched);
quit;

* Final data;
data Analysis;
  set matchedcontrols matchedtreatment;
run;
```

Note that this code only works for the cases in which the matching was a one-to-one match without replacement. That is, the ids of the matched units are not repeated. A left join should be used if some of the ids are repeated.

Table 2 shows the matching results using the nearest available method without replacement. Note that now the samples appear to be balanced in all the observed characteristics.

Table 2. Nearest available neighbor without replacement

Characteristic	Treatment (N=185)		Control (N=185)		Analysis	
	Mean	SD	Mean	SD	Test	p-value
Age (years)	25.82	7.16	25.30	7.46	T=-0.685	0.495
Education (years)	10.35	2.01	10.43	2.02	T=0.398	0.699
Earnings 1974 (U.S. $)	2095.57	4886.62	2293.87	3744.82	T=0.447	0.662
	N	%	N	%		
Race (black)	156	84.32	153	82.70	X=0.176	0.674
Hispanic	11	5.95	10	5.41	X=0.051	0.822
Married	35	18.92	30	16.22	X=0.466	0.495
No Degree	131	70.81	122	65.95	X=1.013	0.314

Table 3 shows the matching results using the caliper matching method without replacement with a caliper of $1xE^{-6}$. Note that in this case many treated units were not matched. The matched samples appear to be well balanced, but using only the matched treatment and control units to establish the treatment effect may add another source of bias due to incomplete matching. Rosenbaum and Rubin (1985) described the bias due to incomplete and inexact matching.

Table 3. Caliper ($1xE^{-6}$) matching

Characteristic	Treatment (N=45)		Control (N=45)		Analysis	
	Mean	SD	Mean	SD	Test	p-value
Age (years)	22.51	6.30	22.22	5.49	t=0.231	0.817
Education (years)	10.89	1.58	10.93	1.68	t=0.130	0.901
Earnings 1974 (U.S. $)	2507.00	6679.40	1190.50	3870.00	t=-1.140	0.256
	N	%	N	%		
Race (black)	30	66.57	28	62.22	X =0.194	0.659
Hispanic	2	4.44	2	4.44	X =0.000	1.000
Married	7	15.56	6	13.33	X =0.089	0.764
No Degree	27	60.00	26	57.78	X =0.045	0.830

Tables 2 and 3 show that propensity score matching creates two balanced samples, but they are just two ways of implementing propensity scores given the model specified. Using different model specifications and matching methods, Dehejia and Wahba (1999) and Smith and Todd (2005) found a range of treatment effect estimates and no clear way to choose among them.

CONCLUSIONS

Propensity score matching methods provide a way to select control observations that are similar to individuals who received a particular treatment. One of the difficulties in applying matching methods is that there are a variety of algorithms available in the literature. The advantages and disadvantages of each method may be clear in theory and in simulation studies, but in practice there is always uncertainty about which one is the best method for a novel situation. However, propensity score matching methods are an important tool whenever there is certainty that the observed variables include most of the factors that are related to outcome and treatment participation. After all, no amount of statistics can substitute for the lack of good data.

REFERENCES

Bertsekas, D.P. (1998), Network Optimization: Continuous and Discrete Models, Athena Scientific, New Hampshire.

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.

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.

Dehejia, R. H. and S. Wahba (2002), Propensity Score Matching Methods for Nonexperimental Causal Studies, *The Review of Economics and Statistics*, 84(1), 151-161. Data available at: http://www.nber.org/%7Erdehejia/nswdata.html

Dehejia, R. H. (2005). Practical Propensity Score Matching: a Reply to Smith and Todd, *Journal of Econometrics*, 125(1-2), 355-364.

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.

LaLonde R. (1986), Evaluating the Econometric Evaluations of Training Programs with Experimental Data, *American Economic Review*, 76, 604-620.

Parsons, L.S. (2004), Performing a 1:N Case-Control Match on Propensity Score, *Proceedings of the 29th SAS Users Group International*, Montreal, Canada.

Rosenbaum, P. R. (2002), Observational Studies. Springer, New York.

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., and Rubin, D.R. (1986), The Bias Due to Incomplete Matching, *Biometrics*, 41, 103-116.

Rubin, D.B. (1980), Bias Reduction Using Mahalanobis-Metric Matching, *Biometrics*, 36, 293-298.

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

ACKNOWLEDGEMENTS

I would like to thank Yishu He for helpful comments and suggestions.

CONTACT INFORMATION

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

Marcelo Coca Perraillon
Adheris Inc.
One Van de Graaff Drive
Burlington, MA 01803
Email: mcperraillon@gmail.com