PhUSE 2014

Paper HE04

Simple and Efficient Matching Algorithms for Case-Control Matching

Berber Snoeijer, PHARMO, Utrecht, The Netherlands
Edith Heintjes, PHARMO, Utrecht, The Netherlands

ABSTRACT
Epidemiological studies comparing outcomes between patient groups with or without certain exposures to treatments or risk factors are prone to bias, caused by differences in patient characteristics. To allow drawing reliable conclusions from such studies, patient groups should be exchangeable. To achieve this, matching exposed patients (here called cases) to reference patients (here called controls) based on important characteristics may help to ensure exchangeability. In studies with rare outcomes a case control design may be more opportune, where patients with the outcome (cases) will be matched to patients without the outcome (controls) based on predefined criteria. Matching criteria can be either exact class variables (e.g. gender, specific comorbidity) or continuous variables which should be matched within a certain range (e.g. age, propensity score). Matching algorithms can take a lot of processing time and memory, which may result in failure to complete the match, or forcing researchers to match in several batches, which is undesirable. This paper will focus on a fast and efficient SAS algorithm that is used to match a predefined maximum number of controls to each case, based on these predefined factors.

INTRODUCTION
In many epidemiologic studies, control subjects are matched to cases to make the disease (case) and non-disease (control) groups comparable with respect to matching criteria and thus reduce bias. Similarly, unexposed patients (controls) are often matched to exposed patients (referred to as cases in this paper for convenience). Matching methods for those studies may differ according to the preferences of the study. Matching can be performed on exact variables like gender and specific co morbidity, on range variables like age and propensity score and often also on a combination of both.

Not all study designs require a fixed number of controls per case, especially when it is desirable in view of power considerations to have as many controls as possible. If a fixed number is not required, the ideal situation would still be a dataset with all cases matched and the controls divided evenly over all cases. The distance between the range values of case and corresponding controls should be as minimal as possible. To reduce computational time, the Greedy principle is often used instead of an optimal distribution method to find the closest match. This principle handles the cases one by one in a random order. After handling of each case, the corresponding matched controls will not be evaluated again for a later case regardless of whether that would be the only match for that case.

This paper describes the features of a macro we developed according to an optimal distribution technique that still uses acceptable and even less programming time in SAS. The direct reason for developing this macro was a study we had to perform with approximately 9000 cases and 225000 possible controls. This matching was performed using the greedy principle and took more than 12 hours in SAS when performing matching steps for each of the 9000 cases.

BACKWARDS CASE SELECTION
Epidemiology studies on real life data are often performed on big datasets. Therefore, techniques for efficient programming should be used to limit the programming time and to make sure that the system can still handle the process.

The first requirements for efficiently running a macro on large datasets are as follows:

- Try to limit the number of data steps
- Try to limit the number of sort steps
- Try to minimize the size of datasets
- Try to minimize the number of matching iterations

Considering these points and the facts that usually multiple controls are matched to one case, and controls are used only once, we decided not to select the best controls on a case by case basis but to check which match would be best for a certain control. Doing this we could match a great number of controls to cases in one step rather than case by case. The corresponding I/O time and thus run time was reduced considerably.

For this approach the controls should be matched to all possible cases that meet the criteria of a possible match. Using SQL this can be done efficiently and in one step. Thereafter, we add a random number to all case-control options and sort by this random number. If more than 1 control is matched to a particular case, we can select as many controls as desired and put the rest of the controls back into the selection pool. In this way, the selection pool reduces quickly and only a few iterations are necessary to match the remaining controls to cases that do not have the maximum number of controls yet. This makes the procedure efficient and fast.

The basic principle translated to a programming environment is as follows:

1) All cases are combined with all possible matches using PROC SQL
2) A random number is added to all possible combinations
3) The dataset is sorted by the control patient numbers and the new random number
4) The first match per control is picked
5) The dataset is sorted by the case patient numbers
6) The number of matches are counted and only matches up to the maximum number of matches per patient are kept

Step 2 to 6 can be repeated with the remaining possible combinations for the remaining cases which do not have the maximum number of matches until no possible combinations are left.

The SAS code we used for this is as follows (step 2 to 6 of the above listed actions):

```
%LET seed=123;
%LET maxmatch=4;
%LET reuse_control= no;

data _Input2;
      set _Input;
      randomnum = ranuni(&Seed);
run;

proc sort data = _Input2;
      by Pt_control randomnum;
run;

data _Result1;
      set _Input2;
      by Pt_control;
      %IF &reuse_control = no %THEN %STR(if first.pt_control then output;) ;
run;

proc sort data = _Result1;
      by Pt_case randomnum;
run;

data _result2;
      SET _result1;
      retain Matchno;
      by Pt_case;
      if first.pt_case then Matchno=1;
            ELSE MatchNo=MatchNo+1;
      if Matchno<=&MaxMatch then output _result2;
run;
```

This method is very fast and is perfect for exact matching. However, it does not yet pick the closest match to a case and the distribution of controls over cases is probably unbalanced. A few moderations and additions are necessary to balance. First of all we can add a limitation to the maximum number of controls matched per round. Then each case will have the same opportunity to get as many controls as possible. Of course this will increase the SAS run time, but still within acceptable levels.
Getting the closest match in an optimal way is somewhat more difficult and is described in the next paragraph.

CLOSEST MATCH

Next to variables that are matched on exact values there are also variables to be matched on a (maximum) range of values like age and propensity score. For these variables, it is important to match a control which is closest to the case. First of all, the absolute distance between the case and control is needed to be able to evaluate this. Then we let this difference prevail above the random number that we also have calculated. See the adapted SAS code below:

```
data _Input2;
      set _Input;
      randomnum = ranuni(&Seed);
      AbsDif = Abs(CaseVal-RefVal);
run;

proc sort data = _Input2;
      by Pt_control AbsDif randomnum;
run;

data _Result1;
      set _Input2;
      by Pt_control;
      %IF &reuse_control = no %THEN %STR(if first.pt_control then output;) ;
run;
```

However, to match as many cases as possible is not as easy as it seems. A match that is closest to case 1 could be also the only match for case 2 although it then is not as close to case 2. To illustrate this we present a test dataset below and propose methods which could overcome this problem.

Our example is a dataset of 3 cases with scores 1.5, 1.7 and 1.9 and a dataset with 6 possible controls with scores 1.6, 1.7, 1.8, 1.85, 1.9 and 2.0. All accepted combinations with the restriction that the difference cannot be more than 0.2 are presented in the table below.

Case ID	Case	Control ID	Control	Difference
1	1.5	10	1.6	0.1
1	1.5	11	1.7	0.2
2	1.7	10	1.6	0.1
2	1.7	11	1.7	0
2	1.7	12	1.8	0.1
2	1.7	13	1.85	0.15
2	1.7	14	1.9	0.2
3	1.9	11	1.7	0.2
3	1.9	12	1.8	0.1
3	1.9	13	1.85	0.05
3	1.9	14	1.9	0
3	1.9	15	2.0	0.1

To get the closest match, we will have to sort on the case and then indicate the closest matched control. Of course, control 1.7 is the best match for case 1.7. However, it is also one of the two only possible matches for case 1.5. The other possible match for 1.5 is as likely to be matched to 1.7 as to 1.5. To overcome this problem and to direct the control 1.7 to the less likely case of 1.5, we tried two possible priority indications.

First of all each possible match was given a priority number as indicated below by ordering per case all possible differences with the controls and adding a rank number accordingly. However, when sorting on this rank priority variable, the value 1.7 will still be matched to 1.7 and not to 1.5.

Case Id	Case Value	Control Id	Control value	Difference	Priority	Possible matches
1	1.5	10	1.6	0.1	1	2
1	1.5	11	1.7	0.2	2	2
2	1.7	10	1.6	0.1	2.5	5
2	1.7	11	1.7	0	1	5
2	1.7	12	1.8	0.1	2.5	5
2	1.7	13	1.85	0.15	4	5
2	1.7	14	1.9	0.2	5	5
3	1.9	11	1.7	0.2	5	5
3	1.9	12	1.8	0.1	3.5	5
3	1.9	13	1.85	0.05	2	5
3	1.9	14	1.9	0	1	5
3	1.9	15	2.0	0.1	3.5	5

Another option is to add the number of possible matches per case as a priority variable (See table below). This results in a priority of control 1.7 to be matched to case 1.5 instead of to case 1.7. In the second table, the values are sorted by control and priority variables. Each first match of the sorted control list will be included (see bold rows).

Case Id	Case value	Control Id	Control value	Difference	Distance Case Priority	Number of possible matches	Matched
1	**1.5**	**10**	**1.6**	**0.1**	**1**	**2**	**X**
2	1.7	10	1.6	0.1	2.5	5	
1	**1.5**	**11**	**1.7**	**0.2**	**2**	**2**	**X**
2	1.7	11	1.7	0	1	5	
3	1.9	11	1.7	0.2	5	5	
2	**1.7**	**12**	**1.8**	**0.1**	**2.5**	**5**	**X**
3	1.9	12	1.8	0.1	3.5	5	
3	**1.9**	**13**	**1.85**	**0.05**	**2**	**5**	**X**
2	1.7	13	1.85	0.15	4	5	
3	**1.9**	**14**	**1.9**	**0**	**1**	**5**	**X**
2	1.7	14	1.9	0.2	5	5	
3	**1.9**	**15**	**2.0**	**0.1**	**3.5**	**5**	**X**

In this way every control is matched to the best possible case. To get an even better distribution of cases over controls, the actions can be performed in a balanced way by only adding 1 control per case per matching round.

All possible matches including the matched control are removed from the match pool and all matched cases are removed as well. This makes the pool dataset smaller after each iteration and thus the remaining process faster and faster in each round. After removing the matched cases and controls, the priority and number of possible matches will be estimated again for each round (see below).

Case id	Case	Control id	Control	Difference	Priority	Possible matches
1	1.5	11	1.7	0.2	1	1
2	1.7	11	1.7	0	1	2
3	1.9	11	1.7	0.2	3	3
2	1.7	13	1.85	0.15	3	2
3	1.9	13	1.85	0.05	1	3
3	1.9	15	2.0	0.1	2.5	3

This method increases the probability that cases get any controls, and that all cases end up with similar numbers of matched controls. In this example, the result is a perfectly equal distribution of controls over the cases.

If you include the number of possible matches in your algorithm, you need to be aware that the possibility exists that some hard to match cases are matched preferentially with possibly larger differences in the matching criteria. This means that subsequent matches of the remaining cases may not be the optimal match for those remaining cases, because the optimal control patient is no longer available. In other words, you make a concession to find more matched cases, at the expense of the closest match of some cases.

EXPERIENCES WITH USING THE MACRO AND DIFFERENT OPTIONS

The following example shows us the results of matching a control dataset of 15000 possible matches to 1000 possible cases with a maximum of 10 matches per case. We examined using one or more of the different options described above; distance rank priority, priority of number of possible matches and whether we first add 1 match to each case before continuing to the next match or add as many controls per round as possible.
In this dataset it appeared that 23 out of the 1000 cases did not have any possible match which leaves a remaining 977 cases that may be matched. The table below gives an overview of the results of all combinations of the different options in number of cases matched, total number of final case-control pairs and number of iterations necessary to get to these results and the corresponding run time.

Match 1 control by round[1)]	Distance Rank Priority	Least number of matches priority	Total number of matched cases	Total number of case-control pairs	Number of iterations	Run Time[2)]
No	No	No	754	4555	16	16 sec
No	Yes	No	872	4989	5	18 sec
No	No	Yes	797	5533	6	21 sec
No	Yes	Yes	784	5536	7	29 sec
Yes	No	No	931	4653	79	1 min, 31 sec
Yes	Yes	No	951	4992	35	1 min, 43 sec
Yes	**No**	**Yes**	**964**	**5152**	**52**	**2 min, 23 sec**
Yes	Yes	Yes	963	5127	52	3 min, 40 sec

[1)] restricted to maximal 7 iterations per round, [2)] small variations due to server usage by others might occur.

Matching 1 control per round increased the total number of matched cases considerably. The corresponding run time increased as well. In addition, using one or both of the the priority options also increased the number of total matched cases.. The effect of these option was even bigger when matching more controls per round. The least number of matches priority gave the most optimal result in total number of matched cases, run time and total number of case-control pairs.

Another example with 2500 cases and 25000 possible matches with a maximum of 8 controls per case gave us the results as presented in the table below. A total of 39 cases did not have any match which leaves us to 2461 possible cases to be matched with controls. The number of possible matches of cases to controls is increased exponentially compared to the previous example. This has a direct effect on the run time. The results for the different options are however, comparable.

Match 1 control by round[1)]	Distance Rank Priority	Least number of matches priority	Run Time[2)]	Total number of matched cases	Total number of case-control pairs	Number of iterations
No	No	No	1 min, 4 sec	1670	8025	25
No	Yes	No	1 min, 0 sec	1924	8781	6
No	No	Yes	1 min, 19 sec	1715	9831	7
No	Yes	Yes	1 min, 57 sec	1685	9828	9
Yes	No	No	4 min, 41 sec	2223	8441	74
Yes	Yes	No	4 min, 37 sec	2290	8859	32
Yes	**No**	**Yes**	**5 min, 29 sec**	**2338**	**9190**	**39**
Yes	Yes	Yes	9 min, 37 sec	2308	9171	45

[1)] restricted to maximal 7 iterations per round, [2)] small variations due to server usage by others might occur.

Based on the results of the above examples, the least number of matches priority is chosen as the best option to perform an optimal distribution of controls over cases in case of non-fixed numbers of controls. The distance rank priority has no additional effect and gives a great increase in run time. It is therefore not used in our algorithms. The priority option has only influence on closest rank variables since for exact matching the number of matches will be equal for all cases that can be matched to the specific control.

The basic SAS code for this matching technique is as follows:

```
Proc SQL;
       Create table _input2 as
       select *,  ranuni(&Seed) AS randomnum, Count(*) as Nmatches
       from _InputMe
       group by pt_case
       order by pt_control, Nmatches, AbsDif, randomnum ;
Quit;

data _Result1;
       set _Input2;
       by Pt_control;
       %IF &reuse_control = no %THEN %STR(if first.pt_control then output;) ;
run;
```

Unfortunately there is still a problem. When the number of possible controls is very high compared to the number of cases, one case will take preference for many controls at the same time while the relative difference in this preference is very small. This may increase the number of iterations considerably since these cases will have to be matched first before others can be matched. In our example we have cases that have over 5000 possible controls while others have only a few. Therefore, we added a bit of conditional code such that there is no preference of a case with 5001 possible matches over a case with 5002 possible matches. See the adjusted code below.

```
Proc SQL;
       Create table _input2 as
       select *,  ranuni(&Seed) AS randomnum,
              case when (Count(*) <= 10) Then count(*)
                   when (Count(*) <= 100) Then ROUND(count(*),10.)
                   when (count(*) <= 1000) then round(Count(*),100.)
                   when (count(*) <= 10000) then round(count(*),1000.)
                   else 10000 end as Nmatches
       from _InputMe
       group by pt_case
       order by pt_control, Nmatches, AbsDif, randomnum ;
Quit;
```

CONCLUSION

The above described method of matching controls to cases is efficient and fast compared to the case by case methods that are often used. In this way the run time for matching cases to controls can be decrease while still an optimal distribution of controls over cases is found. Adding a priority option based on the number of possible matches per case increases the number of cases that get a match considerably and gives the best distribution of controls over the cases when the closes match should be picked.

There are many papers on case-control matching. Some of them describing part of the principle as described here and others that can only be used for exact matching. With the procedures and macro developed at PHARMO we can match on any combination of matching variables, exact matching, range matching and closest match.

CONTACT INFORMATION (HEADER 1)

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

Berber Snoeijer or Edith Heintjes
PHARMO Institute
Van Deventerlaan 30-40
3528 AE Utrecht
Work Phone: +31 30 7440 823
Email: berber.snoeijer@Pharmo.nl, Edith.Heintjes@pharmo.nl
Web: www.pharmo.nl