Matching with Clustered Data: the CMatching Package in R

by Massimo Cannas and Bruno Arpino

Abstract Matching is a well known technique to balance covariates distribution between treated and control units in non-experimental studies. In many fields, clustered data are a very common occurrence in the analysis of observational data and the clustering can add potentially interesting information. Matching algorithms should be adapted to properly exploit the hierarchical structure. In this article we present the CMatching package implementing matching algorithms for clustered data. The package provides functions for obtaining a matched dataset along with estimates of most common parameters of interest and model-based standard errors. A propensity score matching analysis, relating math proficiency with homework completion for students belonging to different schools (based on the NELS-88 data), illustrates in detail the use of the algorithms.

Background

Causal inference with observational data usually requires a preliminary stage of analysis corresponding to the *design* stage of an experimental study. The aim of this preliminary stage is to reduce the imbalance in covariates distribution across treated and untreated units due the non-random assignment of treatment before estimating the parameters of interest. Matching estimators are widely used for this task (Stuart, 2010). Matching can be done directly on the covariates (multivariate matching) or on the propensity score (Rosenbaum and Rubin, 1983). The latter is defined as the probability of the treatment given the covariates value and it has a central role for the estimation of causal effects. In fact, the propensity score is a one dimensional summary of the covariates and thus it mitigates the difficulty of matching observations in high dimensional spaces. Propensity score methods have flourished and several techniques are now well established both in theory and in practice, including stratification on the propensity score, propensity score weighting (PSW), and propensity score matching (PSM).

Whilst the implementation of matching techniques with unstructured data has became a standard tool for researchers in several fields (Imbens and Rubin, 2016), the increasing availability of clustered (nested, hierarchical) observational data poses new challenges. In a clustered observational study individuals are partitioned into clusters and the treatment is non-randomly assigned in each cluster so that confounders may exist both at the individual and at the cluster level. Note that this framework is different from clustered observational data where a treatment is non-randomly assigned for all units in the cluster, for which an optimal matching strategy has been suggested by Zubizarreta and Keele (2017). Such nested data structures are ubiquitous in the health and social sciences where patients are naturally clustered in hospitals and students in schools, just to make two notable examples. If relevant confounders are observed at both levels then a standard analysis, adjusting for all confounders, seems reasonable. However, when only the cluster label — but not the cluster level variables — is observed there is not a straightforward strategy to exploit the information on the clustering. Intuitively, the researcher having a strong belief on the importance of the cluster level confounders may adopt a *'within-cluster'* matching strategy. On the other extreme, a researcher may decide to ignore the clustering by using only the *pooled* data. It is important to note that this pooling strategy implicitly assumes that cluster level variables are not important confounders. Indeed, there have been a few proposals to adapt PSW and PSM to clustered data, see Cafri et al. (2018) for a review. Li et al. (2013) proposed several propensity score weighting algorithms for clustered data showing, both analytically and by simulation, that they reduce the bias of causal effects estimators when "clusters matter," that is, when cluster level covariates are important confounders. In the PSM context, Arpino and Mealli (2011) proposed to account for the clustering in the estimation of the propensity score via multilevel models. Recently, Rickles and Seltzer (2014) and Arpino and Cannas (2016) proposed caliper matching algorithms to perform PSM with clustered data. As we will discuss shortly, these algorithms can be used not only for PSM but also in the more general context of multivariate matching.

In the remaining of this paper, after reviewing the basic ideas underlying matching estimators, we briefly describe the available packages for matching in the R environment. Then, we describe the algorithms for matching with clustered data proposed by Arpino and Cannas (2016) and we present the package **CMatching** implementing these algorithms. The applicability of these algorithms is very broad and refers to all situations where cluster-level data are present (in medicine, epidemiology, economics, etc.). A section is devoted to illustrate the use of the package on data about students and schools, which is a common significant occurrence of clustered data.

Packages for matching unstructured data in R

A list of the most important packages for matching available for R users is shown in Table 1. The **Matching** package, which is required to run **CMatching**, is a remarkably complete package offering several matching options. `Matching` implements many greedy matching algorithms including genetic matching (Diamond and Sekhon, 2013). It also contains a general `MatchBalance` function to measure pre- and post-matching balance with a large suite of diagnostics. As for optimal matching, there are dedicated packages like **designmatch** and **optmatch**. The latter can also be called from **MatchIT**, a general purpose package implementing also the Coarsened Exact Matching approach of Iacus et al. (2011). Full matching is a particular form of optimal matching implemented by **quickmatch** with several custom options.

Package	Description	Reference
Matching	Greedy matching and balance analysis	Sekhon (2011)
MatchIT	Greedy matching and balance analysis	Iacus et al. (2011)
optmatch	Optimal matching	Hansen and Klopfer (2006)
quickmatch	Generalized full matching	Savje et al. (2018)
designmatch	Optimal matching and designs	Zubizarreta et al. (2018)

Table 1: General purpose packages available from CRAN implementing matching algorithms. The list is not exhaustive as there are several packages covering specialized matching routines: a list can be found at http://www.biostat.jhsph.edu/ estuart/propensityscoresoftware.html.

At the time of writing none of the packages described above offers specific routines for clustered data. The **CMatching** package fills this important gap implementing the algorithms for matching clustered data described in the next section.

Matching clustered data

Let us consider a clustered data structure $\mathfrak{D} = \{y_{ij}, x_{ij}, t_{ij}\}, \quad i = 1, \cdots n_j, \quad j = 1, \cdots J$. For observation i in cluster j we observe a vector of covariates X and a binary variable T specifying the treatment status of each observation. Here $n = \sum n_j$ is the total number of observations and J is the number of clusters in the data. We observe also a response variable Y whose average value we are willing to compare across treated and untreated units. A matching algorithm assigns a (possibly empty) subset of control units to each treated unit. The assignment is made with the aim of minimizing a loss function, typically expressed in terms of covariates distance between treated and untreated units. Matching algorithms can be classified as greedy or optimal depending whether the cost function is minimized locally or globally, respectively. Optimal matching algorithms are not affected by the order of the units being matched so they can reach the global optimum, but they are typically more computer-intensive than greedy algorithms proceedings step by step. To bound the possibility of bad matches in greedy matching, it is customary to define a maximum distance for two units to be matched, i.e., a *caliper*, which is usually expressed in standard deviation units of the covariates (or of the propensity score). A greedy matching procedure can then be articulated in the following steps:

1. fix the caliper;
2. match each treated with control unit(s) at minimum covariates distance (provided that distance < caliper);
3. measure the residual covariates' imbalance in the matched dataset and count the number of unmatched units (drops);
4. carefully consider both the balance and the number of drops: if they are satisfactory then proceed to the outcome analysis; otherwise stop or revise previous steps.

If matching proves successful in adjusting covariates, the researcher can proceed to outcome analysis where the causal estimand of interest is estimated from the matched data using a variety of techniques (Ho et al., 2007). On the other hand, if the procedure gives either an unsatisfactory balance or an excessive number of unmatched units, the investigator may try to modify some aspects of the procedure (e.g., the caliper, the way the distance is calculated).

Conceptually, the same procedure can be used also for hierarchical data. Indeed, it is not atypical to find analysis ignoring the clustering and pooling together all units. A pooling strategy implicitly assumes that the clustering is not relevant. However, in several cases the clustering does matter,

that is, the researcher can hypothesize or suspect that some important cluster-level confounders are unobserved. In this case, the information on the cluster labels can be exploited in at least two ways: i) forcing the matching to be implemented *within-cluster* only; ii) performing a *preferential* within-cluster matching, an intermediate approach between the two extremes of pooled and within-cluster matching (Arpino and Cannas, 2016). A within-cluster matching can be obtained by modifying step 2 above in the following way:

2′ match each treated with the control unit(s) *in group j* at minimum covariate distance (provided that distance < caliper).

This procedure may result in a large number of unmatched units (drops) so it increases the risk of substantial bias due to incomplete matching (Rosenbaum and Rubin, 1985), in particular when the clusters are small. This particular bias arises when a matched subset is not representative of the original population of treated units because of several *non random* drops. Even in the absence of bias due to incomplete matching, a high number of drops reduces the original sample size with possible negative consequences in terms of higher standard errors.

It is possible to profit as much as possible of the the exact balance of (unobserved) cluster-level covariates by first matching within clusters and then recovering some unmatched treated units in a second stage. This leads to the preferential within-cluster matching, which can be obtained by modifying step 2 above in the following way:

2″ *a)* match each treated with the control units(s) *in group j* at minimum covariate distance (distance < caliper);

2″ *b)* match each unmatched treated unit from previous step with the control unit(s) at minimum covariate distance in some group different from *j* (provided that distance < caliper).

Now consider the outcome variable Y. We can define for each unit potential outcomes $Y1$, $Y0$ as the outcome we would observe under assignment to the treatment and control group, respectively (Holland, 1986). Causal estimands of interest are the Average Treatment effect: $ATE = E[Y1 - Y0]$ or, more often, the Average Treatment effect on the treated: $ATT = TE[Y1 - Y0]$. Given that a unit is either assigned to the treatment or control group it is not possible to directly observe the individual causal effect on each unit; we have $Y = T \cdot Y1 + (1 - T) \cdot Y0$. In a randomized study T is independent of $(Y0, Y1)$ so, for $k = 0, 1$, we have

$$E(Yk) = E(Yk \mid T = k) = E(Y \mid T = k)$$

which can be estimated from the observed data. In a observational study, matching can be used to balance covariates across treated and control units and then the previous relation can be exploited to impute the unobserved potential outcomes from the *matched dataset*. In our clustered data context, after the matched dataset has been created using one of the algorithms above, the ATT and its standard error can be estimated using a simple regression model:

$$Y_{ij} = \alpha_j + T_{ij}\beta \tag{1}$$

that is, a linear regression model with clustered standard errors to take into account within-cluster dependence in the outcome (Arpino and Cannas, 2016). The resulting ATT estimate is the difference of outcome means across treated and controls, i.e., $\widehat{ATT} := mean(Y \mid T = 1) - mean(Y \mid T = 0)$, computed on the matched data. Standard errors are calculated using the cluster bootstrapping method for estimating variance-covariance matrices proposed by Cameron et al. (2011) and implemented in the package **multiwayvcov**. In general, calculating standard errors for PSM in clustered observational studies is a difficult problem requiring prudence from the researcher. While close formulae exist for weighting estimators (Li et al., 2013), standard error estimation after PSM matching relies upon approximation (Cafri et al., 2018), modelling assumption (Arpino and Cannas, 2016), or simulation (Rickles and Seltzer, 2014).

Multisets and matching output

In this section we briefly detail the two routines in the **CMatching** package. Multisets are useful to compactly describe pseudo code so we recall some definitions and basic properties herein. A multiset is a pair $\{U, m\}$ where U is a given set, usually called universe, and $m : x \to \mathbb{N} \cup \{0\}$ is the multiplicity function assigning each $x \in U$ its frequency in U. Both the summation symbol and union symbols are used to manipulate multisets and they have different meanings: if A and B are multisets then $C \equiv A \cup B$ is defined by $m_C(x) = max(m_A(x), m_B(x))$, while $C \equiv A + B$ is defined by $m_C(x) = m_A(x) + m_B(x)$. For example if $A \equiv \{1, 2, 2\}$ and $B \equiv \{1, 2, 3\}$ then $A \cup B \equiv \{1, 2, 2, 3\}$, $A + B \equiv \{1, 1, 2, 2, 2, 3\}$. In our framework U is the set of observations indexes and thus m gives

information about the number of times a given observation occurred in the matched dataset. Multisets then allow to naturally represent multiple matches arising from matching with replacement. When using multiset notation to describe the output of a matching algorithm, we are implicitly overlooking the fact that the output of a matching algorithm is richer as it also brings out the pairings, i.e., the associations between matched treated and untreated observations. However, it can be noted that this pairing is not relevant for calculating common estimates or common balance measures (e.g., the ATT), as they are invariant to permutations of the labels of the matched observations.

The routines `MatchW` and `MatchPW`

We denote with W_i and $\overline{W}_i$ the sets of treated observations matched within clusters and unmatched within clusters, respectively. In Algorithms 1 and 2 the summation symbol ($\sum$) denotes multiset sum.

Algorithm 1 Algorithm for within-cluster matching

1: **procedure** MATCHW(data)
2: find W_i for each i using `Match` function
3: $M := \sum_i W_i$
4: $mdata := data[M]$ ▷ extracts matched data
5: **if** data contains outcome variable Y **then**:
6: estimate $\widehat{ATT}$ and $sd(\widehat{ATT})$ from model on $mdata$
7: **else**
8: $\widehat{ATT}$<-$sd(\widehat{ATT})$<-NULL
9: **return** $mdata$, $\widehat{ATT}$ and $sd(\widehat{ATT})$

In the first two lines, common to both algorithms, the `Match` function is repeatedly run to produce the matched-within subsets M_i $i =, ..., J$. Then, in algorithm 1 the sum of the M_i in line 3 gives the matched subset M. Algorithm 2 is similar but after finding the M_i's an "additional" subset B is found by recovering some unmatched units (line 3) and then combined to give the final matched dataset. If a response variable Y was included the output of both algorithms also contains an estimate of the ATT (default, but the user can choose also other estimands) and its standard error.

Algorithm 2 Algorithm for preferential within-cluster matching

1: **procedure** MATCHPW(data)
2: find W_i and $\overline{W}_i$ for each i using `Match` function
3: $B :=$ `Match`$(\sum_i \overline{W}_i \cup$ all controls$)$
4: $M := \sum_i W_i + B$
5: $mdata := data[M]$ ▷ extracts matched data
6: **if** data contains outcome variable Y: **then**
7: estimate $\widehat{ATT}$ and $sd(\widehat{ATT})$ from model on $mdata$
8: **else**
9: $\widehat{ATT}$<-$sd(\widehat{ATT})$<-NULL
10: **return** $mdata$, $\widehat{ATT}$ and $sd(\widehat{ATT})$

Functions in the CMatching package

CMatching can be freely downloaded from CRAN repository and it contains the functions listed in Table 2. The main function `CMatch` performs within-cluster and preferential within-cluster matching via subfunctions `MatchW` and `MatchPW`, respectively. The output of the main function can be passed to functions `CMatchBalance` and `summary` to provide summaries of covariates balance and other characteristics of the matched dataset. `CMatch` exploits the `Match` function (see **Matching**) implementing matching for unstructured data. Given a covariate `X` and a binary treatment `T`, the call `Match(X,T,...)` gives the set of indexes of matched treated and matched control units. The `CMatch` function has the same arguments plus the optional arguments `G` (specifying the cluster variable) and `type` to choose between within-cluster matching or preferential-within-cluster matching. We highlight that we chose to frame the **CMatch** in the `Match` function style so that `Matching` users can easily implement PSM with clustered data in a familiar setting.

Function	Description	Input	Output
`CMatch`	Match	X, T, G	A matched dataset
`MatchW`	Match within	X, T, G	A matched dataset
`MatchPW`	Match preferentially within	X, T, G	A matched dataset
`summary.Match`	S3 method for CMatch objects	A matched dataset	General summaries
`CMatchBalance`	Balance analysis	A matched dataset	Balance summaries

Table 2: Main input and output of functions in **CMatching** package.

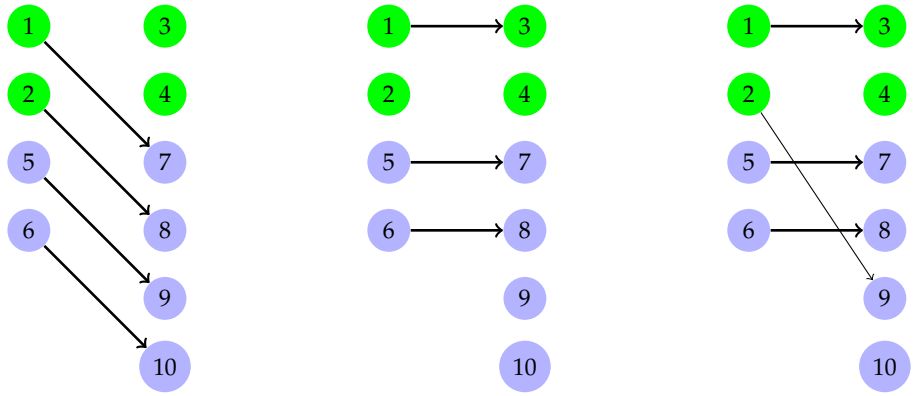

Figure 1: Different matching solutions for the toy dataset (`caliper = 2`). Green and violet circles indicate cluster 1 and cluster 2 units, respectively; arrows indicate matched pairs of treated (left) and control units (right). For each matching we report the absolute percent standardized mean difference of x in the matched subset (`asam`), a measure of residual imbalance.

A simple usage example

For an illustration let us consider an artificial dataset consisting of two clusters, the first containing two treated units and two control units, and the second containing two treated and four controls. We use `g` for the cluster identifier, `x` for the value of the individual level confounder, and `t` for the binary treatment indicator:

```
> toy
   [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
id    1    2  3.0  4.0  5.0    6    7    8  9.0  10.0
g     1    1  2.0  2.0  1.0    1    2    2  2.0   2.0
t     1    1  1.0  1.0  0.0    0    0    0  0.0   0.0
x     1    1  1.1  1.1  1.4    2    1    1  1.3   1.3
```

We also fix a caliper of 2 (in standard deviation units of X, i.e., all units at distance greater or equal than $2 \cdot sd(x) \approx 2 \cdot 0.312 = 0.624$ will not be matched together) and we assume that the ATT is the target parameter. Although artificial the dataset aims at representing the general situation where pooled matching results in matched treated and control units not distributed homogeneously across clusters (see Figure 1, left).

The pooled, within and preferential-within matchings for the toy data are depicted in Figure 1. For each matching we report the `asam`, a measure of residual imbalance given by the absolute percent mean difference of x across treated and controls divided by standard deviation of the treated observations alone. The asam is widely used as a measure of imbalance (Stuart, 2010); its value in the unmatched data is 491. The pooled matching (left) is a complete matching, i.e., all the treated units could be matched. However, note that units in pairs 1-7 and 2-8 may differ in cluster level covariates. Matching within-cluster (center) guarantees perfect balance in cluster level covariates but it is incomplete because unit 2 cannot be matched within-cluster with the given caliper. This is a typical disadvantage of within-cluster matching with smaller clusters. Unit 2 is matched with 9 in the preferential within matching (right), which again is a complete matching.

The above matching solutions can be obtained easily using `CMatch` as follows. For the pooled

matching it is enough to call Match (or, equivalently, CMatch without type specification) while for within and preferential-within matching it is enough to specify the appropriate type in the Match call:

```
#pooled matching
pm <- Match(Y=NULL, Tr=t, X=x, caliper=2)

# same output as before (with a warning about the absence of groups,
# ties=FALSE,replace=FALSE)
pm <- CMatch(type="within", Y=NULL, Tr=t, X=x, Group=NULL, caliper=2)

#within matching
wm <- CMatch(type="within", Y=NULL, Tr=t, X=x, Group=g, caliper=2)

#preferential-within matching
pwm <- CMatch(type="pwithin", Y=NULL, Tr=t, X=x, Group=g, caliper=2)
```

The output of these object is quite rich. However, a quick look at the matchings can be obtained by directly calling the index set of matched observations:

```
> pm$index.treated; pm$index.control
[1]  1  2   5  6
[1]  7  8  10 9
> wm$index.treated; wm$index.control
[1] 1 5 6
[1] 3 7 8
> pwm$index.treated; pwm$index.control
[1] 1 2 5 6
[1] 3 7 8 10
```

Note that vertical alignments in the table above correspond to arrows in Figure 1. With larger datasets and when multiple matches are allowed (i.e., when replace=TRUE) it is probably better to summarize the output. The output objects are of class "CMatch" and a summary method is available for these objects. The summary shows the number of treated and the number of controls *by group*. Moreover, when Y is not NULL it also shows the point estimate of ATT with its model-based estimate of the standard error:

```
> summary(wm)

Estimate...  0
SE.........  NULL

Original number of observations..............  10
Original number of treated obs...............  4
Original number of treated obs by group......

1 2
2 2

Matched number of observations...............  3
Matched number of observations  (unweighted).  3

Caliper (SDs)........................................   2
Number of obs dropped by 'exact' or 'caliper' .........   1
Number of obs dropped by 'exact' or 'caliper' by group

1 2
1 0
```

This summary method does not conflict with the corresponding method for class "Match" which is still available for objects of that class. From the summary above (see also Figure 1, center) we can easily see that matching within groups resulted in one unmatched treated unit from group two. The exact pairing can be recovered from the full output, in particular from the object mdata containing the list of matched and unmatched units. As we noticed in the introduction, it is essential to analyze covariate balance to evaluate the effectiveness of matching as a balancing tool. To this end objects of class "CMatch" can be input of the CMatchBalance function, a wrapper of MatchBalance which offers a large number of balance diagnostics:

```
> CMatchBalance( t~x , match.out=wm)

***** (V1) x *****
                        Before Matching                  After Matching
mean treatment........       1.05                        1.0667
mean control..........     1.3333                        1.1333
std mean diff.........    -490.75                       -115.47
(...)
```

From the output we see that the `asam` decreased from 491 to 115. One can more directly obtain the standardized difference in means of these matchings by subcomponents:

```
> bwm$After[[1]]["sdiff"]
$sdiff
[1] -115.4701

> bpwm$After[[1]]["sdiff"]
$sdiff
[1] -216.5064
```

Whilst artificial, the previous example prompts some general considerations:

- forcing within-cluster matching may result in suboptimal matches with respect to pooled matching. In the toy example, unit 2 is best matched with unit 8 but it is unmatched when `type=within` is chosen (or it could be matched with a less similar control in the same cluster). In both cases the increased bias (due to either incomplete matching or greater imbalance in the observed x) may be at least in part compensated by lower imbalance in cluster level variables;
- preferential-within matching may occasionally recover all unmatched treated units in the within step by matching them between in step 2. However, this complete matching is generally different from the complete pooled matching obtained by ignoring the clustering. In the toy example, unit 2 is recovered in the preferential step but the final matching has a higher imbalance than the pooled one. Again, it is up to the researcher to tune the trade-off between bias due to incomplete matching and bias due to unobserved differences in group covariates.

In applications, when cluster level confounders are unobserved, it is not straightforward to decide which of the matching strategies is the best. However, combining the within and preferential-within routines offered by **CMatching** with sound subject matter knowledge, the researcher can decide how much importance should be given to balance within-clusters based on the hypothesized strength of unobserved cluster level confounders. Note that **CMatching** uses the same caliper for clusters, under the assumption that the researcher is typically interested in estimating the causal effect of the treatment in the whole population of treated units and not by each cluster. This is the main difference between `MatchW` and `Matchby` from the **Matching** package. The latter exactly matches on a categorical variable and the caliper is recalculated in each subset and for this reason `MatchW` estimates generally differ from those obtained from `Matchby`. Another difference is that `Matchby` does not adjust standard errors for within-cluster correlation in the outcome. A third difference is that CMatching provides some statistics (e.g., number of drops) by cluster to better appreciate how well the matched dataset resembles the original clustered structure in terms for example of cluster size.

Demonstration of the CMatching package on NELS-88 data

The **CMatching** package includes several functions designed for matching with clustered data. In this section we illustrate the use of **CMatching** with a an educational example.

Schools dataset

The example is based on data coming from a nationally representative, longitudinal study of 8th graders in 1988 in the US, called NELS-88. **CMatching** contains a selected subsample of the NELS-88 data provided by Kraft and de Leeuw (1998) with 10 handpicked schools from the 1003 schools in the full dataset. The subsample, named `schools`, is a data frame with 260 observations on 19 variables recording either school or student's characteristics.

For the purpose of illustrating matching algorithms in **CMatching**, we consider estimation of the causal effect of doing homework on mathematical proficiency. More specifically, our treatment is a binary variable taking value 1 for students that spent *at least* one hour doing math homework per week and 0 otherwise. The latter is a transformation of the original variable "homework" giving two

almost equal-sized groups. The outcome is math proficiency measured on a discrete scale ranging from 0 to 80. For simplicity we first attach the dataset (`attach(schools)`) and name the treatment and the outcome variables as `T` and `Y`, respectively. The variable `schid` is the school identifier and we rename it as `Group`:

```
> T <- ifelse(homework>1, 1, 0)
> Y <- math
> Group <- schid
```

Since the NELS-88 study was an observational study, we do not expect a simple comparison of math scores across treated and control students (those doing and those not doing math homework) to give an unbiased estimate of the "homework effect" because of possible student-level and school-level confounders. For the purpose of our example, we will consider the following student-level confounders: "ses" (a standardised continuous measure of family socio-economic status), "sex" (1 = male, 2 = female) and "white" (1 = white, 0 other race). The NELS-88 study also collected information on school characteristics. In general, a researcher needs to include all potential confounders in the matching procedure, irrespective of the hierarchical level. Here we considered one school-level confounder: "public" (1 = public schools, 0 = private) but it is plausible to assume that one or more relevant confounders at the school-level are not available. It is clear that, to make the unconfoundedness assumption more plausible, richer data should be used. For example, students' motivation and parents' involvement are potentially important confounders. Thus, the following estimates should be interpreted with caution.

Before illustrating the use of **CMatching**, it is useful to get a better understanding of the data structure. In the school dataset we have a fairly balanced number of treated and control units (128 and 132, respectively). However, in some schools we have more treated than control students, with proportion of treated ranging from 20% to 78%:

```
> addmargins(table(Group, T))
```

```
       T
Group     0   1 Sum
  7472   17   6  23
  7829    6  14  20
  7930   12  12  24
  24725  15   7  22
  25456  17   5  22
  25642  16   4  20
  62821  15  52  67
  68448   8  13  21
  68493  15   6  21
  72292  11   9  20
  Sum   132 128 260
```

From the table above we can notice that the total school sample size is fairly homogeneous with the exception of one school (code = 62821) where the number of treated students (52) is considerably higher than the number of control students (15). These considerations are important for the implementation of the within-cluster and preferential within-cluster matching algorithms. In fact, within-cluster matching can be difficult in groups where the proportion of treated units is high because there are relatively few controls that can potentially serve as a match. Preferential-within-cluster matching would be less restrictive.

This preliminary descriptive analysis is also useful to check if treated and control units are present in each group. In fact, if a group is only composed of treated or control students then within-cluster matching cannot be implemented. Groups with only treated or controls should be dropped before the within-cluster matching algorithm is implemented. We now describe how **Cmatching** can be used to implement matching in our school-clustered dataset.

Propensity score matching

CMatching requires to estimate the propensity score before implementing the matching. Here we estimate propensity scores using a simple logistic regression with only main effects and then estimate the predicted probability for each student:

```
> pmodel <- glm(T~ses + as.factor(sex) + white + public, family=binomial(link="logit"))
> eps <- fitted(pmodel)
```

We do not report the output of the propensity score model because in PSM the propensity score is only instrumental to implement the matching. Within-cluster propensity score matching can be implemented by using the function `CMatch` with the option `type="within"`:

```
> psm_w <- CMatch(type="within", Y=Y, Tr=T, X=eps, Group=Group)
```

The previous command implements matching on the estimated propensity score, `eps`, by using default settings of `Match` (one-to-one matching with replacement and a caliper of 0.25 standard deviations of the estimated propensity score). The output is an object of class `"CMatch"` and a customized summary method for objects of this class gives the estimated ATT and the main features of the matched dataset:

```
> summary(psm_w)

Estimate...  5.2353
SE.........  2.0677

Original number of observations..............  260
Original number of treated obs...............  128
Original number of treated obs by group......

 7472  7829  7930 24725 25456 25642 62821 68448 68493 72292
    6    14    12     7     5     4    52    13     6     9

Matched number of observations...............  119
Matched number of observations  (unweighted).  120

Caliper (SDs)........................................   0.25
Number of obs dropped by 'exact' or 'caliper' .........   9
Number of obs dropped by 'exact' or 'caliper' by group

 7472  7829  7930 24725 25456 25642 62821 68448 68493 72292
    0     2     0     0     0     0     2     5     0     0
```

The summary starts reporting the original total number of students in our sample (260), the total number of treated students (128) and how they are distributed across the different schools. It is of utmost importance to check how many treated units could be matched to avoid bias from incomplete matching. For this reason the output indicates that 119 students in the treatment group found a match (`"Matched number of observations"`), while the remaining 9 were dropped. Note that the unweighted number of treated observations that found a match (`"Matched number of observations (unweighted)"`) is different because of *ties*. Ties management can be controlled by option `ties` of the `Match` function: if `ties=TRUE` when one treated observation matches with more than one control unit all possible pairs are included in the matched dataset and each pair is weighted equal to the inverse of the number of matchedcontrols. If instead `ties=FALSE` is used ties are randomly broken. Note that the summary reports the number of *treated* matched and dropped units because it is assumed by default the ATT is the target estimand. Then the output also reports how the 9 unmatched treated students are distributed across schools. For example, we notice that in one school (68448), 5 of the 13 treated students did not find a match. This is because for these 5 students there was no control student in the same school with a propensity score falling within the caliper. The report also recalls the caliper, which was set to 0.25 standard deviations of the estimated propensity score in this example. The caliper can be set in standard deviation units using the homonymous argument `caliper`. It may be more useful to calculate the percentage of dropped units instead of the absolute numbers. These percentages are not reported in the summary but they can be easily retrieved from the `CMatch` output. For example, we can calculate the percentage of unmatched treated units, both overall and by school:

```
# percentage of drops
> psm_w$ndrops / psm_w$orig.treated.nobs

[1] 0.0703

# percentage of drops by school
> psm_w$orig.dropped.nobs.by.group / table(Group)

Group
 7472  7829  7930 24725 25456 25642 62821 68448 68493 72292
 0.00  0.10  0.00  0.00  0.00  0.00  0.03  0.24  0.00  0.00
```

confirming that the percentage of drops is very low in all schools. We could also similarly calculate the percentage of drops over *treated* observations, which turn out to be high for school 64448. The next step before accepting a matching solution is the evaluation of the achieved balance of confounders across the treatment and control groups. To this end the package contains function CMatchBalance that can be applied to an object of class "CMatch" to compare the balance before and after matching (the matched dataset must be specified in the match.out argument):

```
> b_psm_w <- CMatchBalance(T~ses + as.factor(sex) + white + public, data=schools,
                           match.out=psm_w)

***** (V1) ses *****
                       Before Matching           After Matching
mean treatment........    0.23211                   0.24655
mean control.........   -0.36947                   0.14807
std mean diff.........     61.315                    10.086


***** (V2) as.factor(sex)2 *****
                       Before Matching           After Matching
mean treatment........    0.52344                   0.52941
mean control.........    0.46212                   0.56303
std mean diff.........     12.229                    -6.706


***** (V3) white *****
                       Before Matching           After Matching
mean treatment........    0.73438                    0.7563
mean control.........     0.7197                   0.71429
std mean diff.........     3.3103                    9.7458


***** (V4) public *****
                       Before Matching           After Matching
mean treatment........    0.59375                   0.57983
mean control.........    0.88636                   0.57983
std mean diff.........    -59.346                         0

(...)
```

The output reports the balance for each variable to allow the researcher to assess the effectiveness of matching in balancing each variable. Many balance metrics are provided but for simplicity of exposition we focus on comparing on the standardized mean difference between the two groups of students. Note that the asam can be easily obtained by averaging the standardized mean differences:

```
vec   <-vector()
for(i in 1:length()) {vec[[i]] <- b_psm_w$AfterMatching[[i]]$sdiff}
> mean(abs(vec))
[1] 6.634452
```

from which we can see that the initial asam of 34 (see Table 3) sharply decreased after matching. Balance improved dramatically for ses and public (results not shown). In fact, for the latter it was possible to attain exact matching. This is guaranteed by within-cluster matching because it forces treated and control students to belong to the same school. For the same reason, within-cluster matching also guarantees a perfect balance of all other school-level variables (even unobservable) not included in the propensity score estimation. The balance improved also for the sex variable but not for the dummy white (from 3.31% to 9.75%). In a real study, the investigator may attain a matching solution with an even better balance of the dummies for white and sex by changing the propensity score model or one or more options in the matching algorithms. For example, a smaller caliper could be tried. Note that CMatchBalance is a wrapper of the MatchBalance function (**Matching** package) so it measures balance globally and not by group. This is acceptable also in a hierarchical context since we first and foremost consider the overall balance. While a group-by-group balance analysis may be useful it is only the *average* asam which matters when estimating the ATT on the *whole* population of treated units.

We highlighted that the within-cluster algorithm always guarantees a perfect balance in all cluster-level confounders as in the example above. However, note that it was not possible to match some

treated observations and in part this can be due to the matching constrained to happen only within clusters. The researcher can relax the constraint using preferential within-cluster matching. This algorithm can be invoked using the option `type="pwithin"` in the `CMatch` function:

```
> psm_pw <- CMatch(type="pwithin", Y=Y, Tr=T, X=eps, Group=Group)
> summary(psm_pw)
```

	PSM		MAH†	
Statistics	**within**	**pwithin**	**within**	**pwithin**
Orig. number of obs.	260	260	260	260
Orig. number of treated obs.	128	128	128	128
Matched number of treated obs.	119	128	84	128
Number of drops	9	0	44	0
ASAM before	34.05	34.05	34.05	34.05
ASAM after	6.63	8.13	0.47	6.31
ATT	5.24	5.18	4.25	4.34
SE	2.07	2.14	2.23	1.83

† PSM: propensity score matching; MAH: Mahalanobis matching.

Table 3: Matching to allow a fair comparison of math test score of students doing (treated) and not doing homework in a school clustered dataset (NELS-88 data): comparing matching solutions obtained from CMatching.

A comparison of results between within and preferential within matching is given in Table 3. The preferential within-cluster matching was able to match all treated students ("`Matched number of obs`" = 128), i.e., the number of unmatched treated students is 0 ("`Number of drops`" = 0). In this example, the 9 treated students that did not find a matched control within the caliper in the same school found a control match in another school. Looking to the overall balance attained by matching preferentially within, we can notice that preferential within-cluster matching showed a slightly higher asam than within-matching. In fact there is no clear "winner" between the two algorithms: the balance of the individual level variables `ses` and `white` improves slightly with the preferential within-cluster matching while for `sex` the within-cluster matching is considerably better (not shown). Importantly, using preferential within-cluster matching the absolute standardized mean difference for the school-level variable `public` is 3.2% This is not a high value because most of the treated units actually found a match within schools. However, this finding points to the fact that preferential within-cluster matching is not able to perfectly balance cluster level variables as the within-cluster approach.

Finally, having achieved a satisfactory balance with a very low number of drops we can estimate the ATT on the matched dataset. When the argument `Y` is not `NULL`, an estimate is automatically given otherwise the output of the `CMatch` function only gives information about the matching. The estimated average effect of studying math for at least one hour per week on students' math score is 5.24 with a standard error of 2.07 when matching within schools (Table 3). It is worth mentioning that the reported SE is model based and adjusts for non-independence of students within schools. From Table 3 we can see that the estimated ATT and SE for the preferential-within school approach are very similar to those obtained with the within-cluster approach. We stress that the estimated ATT should be considered carefully and only after checking the matching solution.

In conclusion, preferential within-cluster matching is expected to improve the solution of the within-cluster matching in terms of a reduced number of unmatched units. On the other hand, within-cluster matching guarantees a perfect balance of school-level variables (both observed and unobserved) while preferential within does not. The researcher, choosing between the two algorithms, has to consider the trade-off between having a perfect balance of cluster level variables (within-cluster matching) or reducing the number of unmatched treated units (preferential within-cluster matching). The researcher can also implement both approaches and compare the results as a sort of sensitivity analysis.

Multivariate covariate matching

Instead of matching on the propensity score, the researcher may match directly on the covariates space. This strategy can be advantageous when the number of covariates is fairly low and it is expected to match exactly a large number of units on the original space. The syntax is very similar to the above for

propensity score matching: the only difference is that the user indicates the covariate matrix instead of the propensity score in the *X* argument:

```
> mal_w <- CMatch(type="within", Y=Y, Tr=T, X=cbind(ses, sex, white, public),
                  Group=Group)
```

When *X* is a matrix, the covariate distance between a treated and a control unit is measured by Mahalanobis metrics, i.e., essentially the multivariate Euclidean distance scaled by the standard deviation of each covariate, a metrics which warrants an homogeneous imbalance reduction property for (approximately) ellipsoidally distributed covariates (Rubin, 1980). From Table 3, columns 3-4, we can see that the balance of covariates was indeed very good. Note that the estimated ATT using Mahalanobis matching is lower than the corresponding estimate obtained with propensity score matching. However, within-cluster matching using the Mahalanobis distance has generated a large number of unmatched treated units (44). Therefore, in this case preferential within-cluster matching could be used to avoid an high proportion of drops.

Other strategies

Other strategies for controlling unobserved cluster covariates in PSM have been suggested by Arpino and Mealli (2011). The basic idea is to account for the clustering in the estimation of the propensity score by using random- or fixed-effects models. This strategy can be combined with the matching strategies presented before in order to 'doubly' account for the clustering. This can be done easily with **CMatching**. As an example we consider estimating the propensity score with a logit model with dummies for schools and then matching preferentially within-schools using the estimated propensity score:

```
# estimate ps
> mod <- glm(T ~ ses + parented + public + sex + race + urban
             schid - 1, family=binomial(link="logit"), data=schools)
eps <- fitted(mod)

# match within on eps
> dpsm <- CMatch(type="pwithin", Y=math, Tr=T, X=eps, Group=NULL)
```

In concluding this section, we also mention some other matching strategies which can be implemented using **CMatching** and some programming effort. First, the utility of the algorithms naturally extends when there are more than two levels. In this case, it can be useful to match preferentially on increasingly general levels, for example by allowing individuals to be matched first between regions and then between countries. Another natural extension involves data where units have multiple membership at the same hierarchical level. In this case it is possible to combine match within (or preferential-within) across levels, for example by matching students both within schools and within living district (e.g. 1 out of 4 possible combinations).

Summary

In this paper we presented the package **CMatching** implementing matching algorithms for clustered data. The package allows users to implement two algorithms: i) within-cluster matching and ii) preferential within-cluster matching. The algorithms provide a model-based estimate of the causal effect and its standard error adjusted for within-cluster correlation among observations. In addition, a tailored summary method and a balance function are provided to analyze the output. We illustrated the case for within and preferential within-cluster matching analyzing data on students enrolled in different schools for which it is reasonable to assume important confounding at the school level. Finally, since the analysis of clustered observational data is an active area of research, we are willing to improve on standard error calculations for matching estimators with clustered data if new theoretical results in the causal inference literature will become available.

Bibliography

B. Arpino and M. Cannas. Propensity score matching with clustered data. An application to the estimation of the impact of caesarean section on the Apgar score. *Statistics in Medicine*, 35(12): 2074–2091, 2016. URL https://10.1002/sim.6880. sim.6880. [p1, 3]

B. Arpino and F. Mealli. The specification of the propensity score in multilevel observational studies. *Computational Statistics & Data Analysis*, 55:1770–1780, 2011. URL https://dx.doi.org/10.1016/j.csda.2010.11.008. [p1, 12]

G. Cafri, W. Wang, P. H. Chan, and P. C. Austin. A review and empirical comparison of causal inference methods for clustered observational data with application to the evaluation of the effectiveness of medical devices. *Statistical Methods in Medical Research*, 0(0):0962280218799540, 2018. URL https://10.1177/0962280218799540. [p1, 3]

A. C. Cameron, J. B. Gelbach, and D. L. Miller. Robust inference with multiway clustering. *Journal of Business & Economic Statistics*, 29(2):238–249, 2011. URL https://10.1198/jbes.2010.07136. [p3]

A. Diamond and J. S. Sekhon. Genetic matching for estimating causal effects: A general multivariate matching method for achieving balance in observational studies. *Review of Economics and Statistics*, 95(3):932–945, 2013. URL https://doi.org/10.1162/REST_a_00318. [p2]

B. B. Hansen and S. O. Klopfer. Optimal full matching and related designs via network flows. *Journal of Computational and Graphical Statistics*, 15(3):609–627, 2006. URL https://10.1198/106186006X137047. [p2]

D. E. Ho, K. Imai, G. King, and E. A. Stuart. Matching as nonparametric preprocessing for reducing model dependence in parametric causal inference. *Political Analysis*, 15(3):199–236, 2007. URL https://10.1093/pan/mpl013. [p2]

P. W. Holland. Statistics and causal inference. *Journal of the American Statistical Association*, 81(396): 945–960, 1986. [p3]

S. M. Iacus, G. King, and G. Porro. Multivariate matching methods that are monotonic imbalance bounding. *Journal of the American Statistical Association*, 106(493):345–361, 2011. URL https://doi.org/10.1198/jasa.2011.tm09599. [p2]

G. W. Imbens and D. B. Rubin. *Causal Inference for Statistics, Social and Biomedical Sciences*. John Wiley & Sons, 2016. [p1]

I. Kraft and J. de Leeuw. *Introducing Multilevel Modeling*. London, Sage, 1998. [p7]

F. Li, A. M. Zaslavsky, and M. B. Landrum. Propensity score weighting with multilevel data. *Statistics in Medicine*, 32(19):3373–3387, 2013. ISSN 1097-0258. URL https://10.1002/sim.5786. [p1, 3]

J. H. Rickles and M. Seltzer. A two-stage propensity score matching strategy for treatment effect estimation in a multisite observational study. *Journal of Educational and Behavioral Statistics*, 39(6): 612–636, 2014. URL https://10.3102/1076998614559748. [p1, 3]

P. Rosenbaum and D. Rubin. The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70:41–55, 1983. URL https://10.2307/2335942. [p1]

P. Rosenbaum and D. Rubin. Constructing a control group using multivariate matched sampling methods that incorporate the propensity score. *The American Statistician*, 39:33–38, 1985. URL https://doi.org/10.1080/00031305.1985.10479383. [p3]

D. B. Rubin. Bias reduction using mahalanobis-metric matching. *Biometrics*, 36(2):293–298, 1980. URL https://10.2307/2529981. [p12]

F. Savje, J. Sekhon, and M. Higgins. *Quickmatch: Quick Generalized Full Matching*, 2018. URL https://CRAN.R-project.org/package=quickmatch. R package version 0.2.1. [p2]

J. S. Sekhon. Multivariate and propensity score matching software with automated balance optimization: The Matching package for R. *Journal of Statistical Software*, 42(7):1–52, 2011. URL https://dx.doi.org/10.18637/jss.v042.i07. [p2]

E. A. Stuart. Matching methods for causal inference: A review and a look forward. *Statist. Sci.*, 25(1): 1–21, 2010. URL https://10.1214/09-STS313. [p1, 5]

J. R. Zubizarreta, C. Kilcioglu, and J. P. Vielma. *Designmatch: Matched Samples That Are Balanced and Representative by Design*, 2018. URL https://CRAN.R-project.org/package=designmatch. R package version 0.3.1. [p2]

R. J. Zubizarreta and L. Keele. Optimal multilevel matching in clustered observational studies: A case study of the effectiveness of private schools under a large-scale voucher system. *Journal of the American Statistical Association*, 112:0, 2017. URL https://doi.org/10.1080/01621459.2016.1240683. [p1]

Massimo Cannas
University of Cagliari
Via Sant'Ignazio 87, 09123 Cagliari, Italy
(ORCiD: https://orcid.org/0000-0002-1227-5875)
massimo.cannas@unica.it

Bruno Arpino
University of Florence
Viale Morgagni, 59 50134 Firenze, Italy
RECSM-UPF
Carrer Ramon Trias Fargas 25-27, 08005 Barcelona, Spain
(ORCiD: https://orcid.org/0000-0002-8374-3066)
bruno.arpino@unifi.it