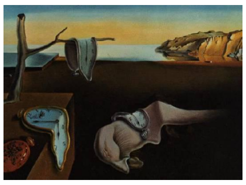

Journal of Statistical Software
April 2008, Volume 25, Issue 11. http://www.jstatsoft.org/

Invariant and Metric Free Proximities for Data Matching: An R Package

Stefano M. Iacus
University of Milan

Giuseppe Porro
University of Trieste

Abstract

Data matching is a typical statistical problem in non experimental and/or observational studies or, more generally, in cross-sectional studies in which one or more data sets are to be compared. Several methods are available in the literature, most of which based on a particular metric or on statistical models, either parametric or nonparametric. In this paper we present two methods to calculate a proximity which have the property of being invariant under monotonic transformations. These methods require at most the notion of ordering. An open-source software in the form of a R package is also presented.

Keywords: data matching, R.

1. Why a metric free proximity?

In social sciences researchers often need to estimate the effect of a "treatment" (like a social program, a medical therapy, etc.) on a population of individuals. The estimation generally entails the comparison of an outcome variable between the group of the subjects exposed to the treatment and a "control" group which did not receive the treatment.

The evaluation of the differential treatment effect on the outcome for *treated versus control* individuals has to be done given the same pre-treatment conditions (see Heckman, Ichimura, and Todd 1997; Heckman, Ichimura, Smith, and Todd 1998). In experimental studies two similar groups of individuals are randomly selected from a population: one is then exposed to the treatment and the other is not. In observational studies the assignment to the treatment is usually non-randomized and, therefore, the distributions of pre-treatment covariates are different between the two groups. Finding control units similar to each treated individual represents the preliminary problem of the analysis. Therefore, a technique is required for *matching* observations coming from different data sets. If the match fails partially or completely, it means that the distributions of the covariates in the two groups do not overlap.

This is a case of (partial or total) lack of *common support*.

When there are many variables an exact match might become an unfeasible task due to dimensionality. Hence, since the seminal paper by Cochran and Rubin (1973), many authors have faced the matching problem and several matching techniques have been developed to overcome this dimensionality issue (see also Rubin 1973a,b). Cochran and Rubin (1973) proposed to solve the problem of multivariate matching using the Mahalanobis distance to match the nearest available individuals. Later Rosenbaum and Rubin (1983) introduced the notion of *propensity score* (PS) as the probability that an individual receives the treatment, conditional on his/her covariates. Rosenbaum (1989) introduced further the notion of *optimal matching*, that is a matching strategy to group treated and control units in such a way that minimizes the overall distance between observations. More recently, Diamond and Sekhon (2005) proposed a matching technique based on genetic algorithms.

The drawback of all unconstrained methods based on distances or propensity scores is that, if two data sets have no common support, a match can always be found among the "less distant" observations, but the achieved matching may be useless. In such cases, it is more effective the use of a *caliper* (a radius or a bound) on the distance or on the propensity score or a mix of the two as explained in Gu and Rosenbaum (1993).

Propensity score matching has been brought back to the attention of the statistical community after the work of Dehejia and Wahba (1999). In their paper the authors suggest that propensity score matching is a good device to reduce bias in the estimation of the average treatment effect in observational studies (no matter the data sets to be matched). The debate that followed (Dehejia and Wahba 2002; Smith and Todd 2005a,b; Dehejia 2005) was mainly focused on the sensitivity of the match to the model used to estimate the propensity score.

To summarize, from the recent literature it emerges that propensity score methods seem to be too sensitive to model specification but distance based methods apply properly only to data which lie in R^k possibly under multi-normal assumptions on the distribution of the data. Hence the need of a measure of proximity which is not related to any notion of statistical model or does not require additional assumptions on the structure of the data. In the next sections we present two implementations of the same idea of an invariant and metric free proximity. The main argument behind this notion of proximity is that the proximities π_{ij} are interpreted as the "belief of observation i and j to be equal in attributes" in the sense that they occupy the same region of the space (whichever the structure of the space!).

We will introduce two different methods to generate a proximity. The first approach, presented in Section 2.1 is to use regression trees to obtain rank related proximities. This is a Monte Carlo method already known to the literature which we present for completeness. Section 2.2 shows a faster and non randomized method which is introduced for the first time in this paper. This second method also allows for an easier interpretation of the proximities obtained. A comparison of these two methods is presented at the end of the section. Applications to data matching are presented in Section 3 and the corresponding package **rrp**, available from the Comprehensive R Archive Network at `http://CRAN.R-project.org/package=rrp`, for the R language (R Development Core Team 2008) is described in Section 4.

2. Two different algorithms

To obtain a metric free proximity, instead of defining a notion of center and fixing a radius, we make use of ranks to decide whether two (or more) observations are in the same portion of the space. For non-ordinal variables, the ranks are even superfluous. Such a rank-based proximity is also invariant under monotonic transformations of the ranked data because ranks are preserved in this case[1]. It should be noticed that the definition of rank in this paper is generalized in the following sense: if data are raw, then ranks take the usual meaning. When observations are coarsened into intervals, each observation is associated to the interval to which it belongs and hence the rank of the observation is the rank of the corresponding interval. An example of this rank assignment is given at the end of the present section.

The proximities π_{ij} are such that $\pi_{ij} \in [0,1]$. Up to our algorithms, $\pi_{ij} = 1$ means that the observations i and j are "likely to be equal", $\pi_{ij} = 0$ means i and j are completely different observations. When $\pi_{ij} \in (0,1)$ the interpretation of this value is strictly related to the algorithm in use.

Let $x_i = (x_{i1}, \ldots, x_{ip})$ be the vector of attributes $(X_1, X_2, \ldots, X_p)$ for observation $i = 1, \ldots, n$, where n is the sample size. Working with ranks, there is always the possibility to have extremal data disturbing the analysis, given that we do not make use of any distance. In order to reduce the risk, we assume that each numeric variable X_k in the data set is preliminarily discretized by dividing the support of X_k into m intervals, so that extremal values are likely to be isolated in cells near the border of the support of the data. After discretization, the variable X_k is reclassified as an ordinal categorical variable. Both boolean and unordered categorical variables are treated as unordered categorical variable. So, from now on we assume that data consists only of categorical variables (either ordinal or not, which might have been obtained by discretization). Once the data are ready, we proceed with rank assignments. It might happen that, in case of discretization in m intervals, some intervals result empty. In such a case, we assign progressive ranks to the observations keeping the information that some cells are empty. The following example clarifies the procedure of assigning the ranks: suppose we have the following 7 data for variable X_k: 1, 1, 1, 5, 5, 6, 8. Suppose we decided to split the support of X_k, i.e., the interval $[1, 8]$, in $m = 4$ intervals as follows: $[1, 2.75]$, $(2.75, 4.50]$, $(4.50, 6.25]$ and $(6.25, 8]$. No observation fall in class #2, i.e., $(2.75, 4.50]$, and the ranks of the data will be assigned taking this information into account, as follows:

x_{ik}	1	1	1	5	5	6	8
R_{ik}	1	1	1	3	3	3	4

where R_{ik} is the rank assigned to observation i along variable k. For unordered variables we do not make use of ranks, so observations with different values will be treated as different observations.

2.1. The RRP proximity

The RRP algorithm discussed in Iacus and Porro (2007a,b) is as follows: to each observation i a *fictitious* and exogenous response variable $U_i \sim U(0,1)$ is assigned, where $U(0,1)$ is the uniform distribution on [0,1]. A regression tree that models the random vector $(U_1, \ldots, U_n)$ as a function of the attributes $(X_1, X_2, \ldots, X_p)$ is grown[2] yielding as a result a *random*, *recursive*

[1]It also means that (in the case of discretized observations) if a monotonic transformation of the data is made, it should also be applied to the cut points of the interval considered.

[2]It should be stressed that this response variable U is *not* the outcome variable of any statistical model: U is introduced, *deus ex machina*, only to have randomness in this Monte Carlo method. The use of regression trees is merely instrumental.

and non empty *partition* of the data. A proximity measure π from this random partition is induced defining the following set of binary parameters: $\pi_{ij} = 1$ for all observations i and j in the same leaf of the regression tree; $\pi_{ij} = 0$ otherwise. This partition and the proximity measure entirely depend on the realization of the random vector $(U_1, \dots, U_n)$: therefore the procedure is replicated `R` times and the final proximity measure is defined as the average of the π_{ij}'s over all the replications. Regression trees works on the ranks, hence the suggested discretization of the data is well suited for the method. The RRP algorithm is the following:

```
1. Choose a value of R (usually R = 250). Set r = 1.
2. while(r <= R)
    (a) draw $n$ pseudo-random numbers $u_1, \dots, u_n$ from the $U(0,1)$ law
    (b) grow a regression tree $u_i \sim (x_{i1}, x_{i2}, \dots, x_{ip})$
    (c) set $\pi_{ij}^{(r)} = 1$ if observation $i$ and $j$ are in the same terminal node, otherwise set $\pi_{ij}^{(r)} = 0$
    (d) r = r+1
3. end while
4. define $\Pi^{RRP}$ and $\Delta^{RRP}$ as follows:
```

$$\Pi^{RRP} = \left[\pi_{ij}^{RRP} = \frac{1}{R}\sum_{r=1}^{R} \pi_{ij}^{(r)}\right], \qquad \Delta^{RRP} = \left[\delta_{ij}^{RRP}\right] = 1 - \Pi^{RRP} \tag{1}$$

We call Π^{RRP} the *RRP-proximity matrix* and Δ^{RRP} the RRP-*dissimilarity matrix*.

2.2. The rank-based proximity

The same idea can be exploited using directly the ranks without making use of a Monte Carlo approach and regression trees. This second method is cleaner and for obvious reasons more efficient. The main argument is to consider for each observation in the data set its rank for each variable X_k and setting $\pi_{ij}^{(k)} = 1$ if $|R_{ik} - R_{jk}| \leq 1$, i.e., observation i and j have ranks equal or consecutive and $\pi_{ij}^{(k)} = 0$ otherwise. The final proximity is obtained by averaging over all the variables:

```
1. for(k in 1:p)
        if($X_k$ is ordered) let $\pi_{ij}^{(k)} = 1$ if $|R_{ik} - R_{jk}| \leq 1$ and $\pi_{ij}^{(k)} = 0$ otherwise.
        if($X_k$ is unordered) let $\pi_{ij}^{(k)} = 1$ if $x_{ik} = x_{jk}$ and $\pi_{ij}^{(k)} = 0$ otherwise.
2. end for
3. define $\Pi^{RANK}$ and $\Delta^{RANK}$ as follows:
```

$$\Pi^{RANK} = \left[\pi_{ij}^{RANK} = \frac{1}{p}\sum_{k=1}^{p} \pi_{ij}^{(k)}\right], \qquad \Delta^{RANK} = 1 - \Pi^{RANK} \tag{2}$$

We call Π^{RANK} the *RANK-proximity matrix* and Δ^{RANK} the *RANK-dissimilarity matrix*.

2.3. Comparison and properties of the two proximities

Clearly, both Δ^{RRP} and Δ^{RANK} are not metrics, therefore it is hard to make comparisons with other widely used metrics (e.g., Euclidean, Mahalanobis, Manhattan), but an important difference with respect to distances is that both dissimilarity matrices (1) and (2) should not be used in hierarchical clustering, because all totally different (up to our methods) observations will have the same dissimilarity, (i.e., $\delta_{ij} = 1$), making impossible to aggregate the "closest" units. Finally, both dissimilarities are not full rank (in the sense of Little and Rubin (2002), page 69), i.e., $\delta_{ij} = 0$ even if the condition $x_{ik} = x_{jk}$ is not met for all attributes $k = 1, \ldots, p$, due to discretization.

An advantage of the proximity defined in (2) is that it has a clear interpretation. Indeed, $\pi_{ij}^{RANK} = q$ means that observation i and j have proximity $\pi_{ij}^{(k)} = 1$ along $q\%$ of the variables X_k, $k = 1, \ldots, p$. Nevertheless, even if $\pi_{ij}^{RANK} = \pi_{ik}^{RANK} = 0.75$ means "observation i and j and i and k are similar in 75% of the variables", this is not necessarily the same subset of variables. Unfortunately, the same interpretation cannot be attributed to the *RRP*-proximities.

The *RRP*-proximity in (1) and the *RANK*-proximity in (2) are numerically different for values internal to the interval $(0, 1)$, but they both allow to identify nearly the same "twin" observations (i.e., $\pi_{ij} = 1$) and the "completely different" units (i.e., $\pi_{ij} = 0$).

Both proximities are metric free and invariant under monotonic transformations of the data but the rank proximity (2) is easier to understand, simplest to implement and considerably faster to obtain.

Other approaches to data matching based on ranks (defined in the usual way) do exist. For example, let M_k be the maximum rank for variable k and $r_k(i)$ the rank of observation i along variable X_k, then

$$z_k(i) = \frac{r_k(i) - 1}{M_k - 1}$$

is such that $z_k(i) \in [0, 1]$ is a score which is also invariant under monotonic transformation of the data. The vector $(z_1(i), z_2(i), \ldots, z_p(i))$ should be used to replace $(x_{i1}, x_{i2}, \ldots, x_{ip})$ in the data matrix and then any distance based matching method could be applied on the new matrix. Alternatively, it is possible to assign the score $z_i = (z_1(i) + z_2(i) + \cdots + z_p(i))/p$ to each observation and treat it as an new *balance score*[3].

The function `rrp.dist` in the package **rrp** implements the algorithm for the RRP method and the function `rank.dist` implements the other algorithm. Both functions may return an R object of class `dist` for memory saving and compatibility with other matching algorithms. In the case of `rank.dist` the default object which is returned is a list of the same length of the observations. Each element i of the list contains a named vector of proximities where the names correspond to the row names j's of the original data set for which the proximity between observation i and the j's are positive.

The *RANK*-proximity algorithm makes also possible to weight differently the variables included in the match. If, for example, there are some hints or requirements to have stricter

[3]See Rosenbaum and Rubin (1983) for a definition of balance score.

match on some of the covariates, it is possibile to redefine the proximity (2) as follows

$$\Pi^{RANK} = \left[\pi_{ij}^{RANK} = \frac{\sum\limits_{k=1}^{p} \pi_{ij}^{(k)} w_k}{\sum\limits_{k=1}^{p} w_k} \right], \qquad \Delta^{RANK} = 1 - \Pi^{RANK} \tag{3}$$

where $w_k > 0$ are the weights. Obviously, the interpretation of the values of $\pi_{ij}^{RANK} \in (0,1)$ is less clear in this case.

3. Applications to data matching

The problem of data matching has been described in the Introduction. Several packages of the R statistical environment (R Development Core Team 2008) implement different matching methods. Among these, we mention **MatchIt** (Ho, Imai, King, and Stuart 2007) which is a generic framework for data matching, **Matching** (Diamond and Sekhon 2005) which implements also genetic matching, **optmatch** by Hansen (2004) implements optimal full matching as described in Rosenbaum (1991). In the next sections we present an analysis of randomly generated data which allow for an easy interpretation of the role of the proximities in the identification of a common support and another data matching application on real data borrowed from the econometric literature.

3.1. Application to simulated data

As an example of application we show the ability of the two proximities to identify the common support on the test data sets presented in Figure 1. In these examples there are two groups to be matched: in one case (up) the two groups, say A and B present a subset of observations which have a common support and in the other case (down) the two groups are completely separated. The upper data (called `dataA` in the script) is randomly generated as follows

```
R> require("rrp")
R> set.seed(123)
R> nt <- 200
R> nc <- 200
R> n <- nt + nc
R> theta <- runif(nt) * 2 * pi
R> r <- runif(nt) * 0.8
R> x1 <- cos(theta) * r * 1.3
R> y1 <- sin(theta) * r * 0.6
R> theta <- runif(nc) * 2 * pi
R> r <- runif(nc) * 0.8
R> x2 <- 0.5 + cos(theta) * r * 0.5
R> y2 <- sin(theta) * r * 1.4
R> dataA <- data.frame(X1 = c(x1, x2), X2 = c(y1, y2))
```

and the common support for these points is identified as follows

```
R> tsubjects <- 1:nt
R> csubjects <- (nt + 1):n
R> treated <- c(rep(TRUE, nt), rep(FALSE, nc))
R> maxY <- max(dataA[tsubjects, 2])
R> minY <- min(dataA[tsubjects, 2])
R> idxc <- which((dataA[csubjects, 2] <= maxY) &
+     (dataA[csubjects, 2] >= minY))
R> minX <- min(dataA[csubjects[idxc], 1])
R> maxX <- max(dataA[csubjects[idxc], 1])
R> idxt <- which((dataA[tsubjects, 1] <= maxX) &
+     (dataA[tsubjects, 1] >= minX))
R> cat(paste("number of treated units in the common support:",
+      length(idxt), "\n"))

number of treated units in the common support: 62
```

The second set of data (called `dataB` in the script) is generated as follows

```
R> set.seed(123)
R> theta <- runif(nt) * 2 * pi
R> r <- runif(nt) * 0.8
R> x1 <- cos(theta) * r * 1.3
R> y1 <- sin(theta) * r * 0.6
R> theta <- runif(nc) * 2 * pi
R> r <- runif(nc) * 0.8
R> x2 <- 1.5 + cos(theta) * r * 0.5
R> y2 <- sin(theta) * r * 1.4
R> dataB <- data.frame(X1 = c(x1, x2), X2 = c(y1, y2))
```

In order to study the common support we introduce an index based on the proximity matrix. This is index is denoted by $S(\lambda)$, $\lambda \in [0,1]$, and defined as follows:

$$S(\lambda) = \frac{\#\left\{i \in T : \max_{j \in \mathbf{B}} \pi_{ij} \geq \lambda\right\}}{n_A}, \qquad 0 \leq \lambda \leq 1,$$

where n_A is the number of observation from group A and $\mathbf{A}$ and $\mathbf{B}$ are, respectively, the set of indeces of observations from group A and B. For fixed λ, the function $S(\lambda)$ is the proportion of units in group A that have a proximity greater or equal λ with some units in group B[4]. Note that $S(\lambda)$ is a non increasing function of λ and it rapidly goes to zero if no unit in A has been found similar to units in group B. Making use of the `rrp.dist` function in package **rrp**, we obtain the *RRP*-proximity matrix for both data sets[5] in Figure 1.

[4]Note that, in observational studies only one of the two groups matter. Usually, A is the set of indexes of the treated units and B the set of indexes of the control group.

[5]In both cases, each numeric variable is discretized in 14 intervals, hence 15 cutpoints are needed in the script.

```
R> MyCut <- 15
R> dati1 <- dataA
R> dati2 <- dataB
R> for (i in 1:2) dati1[, i] <- ordered(cut(dataA[, i],
+     seq(min(dataA[, i], na.rm = TRUE), max(dataA[, i], na.rm = TRUE),
+     length = MyCut), include.lowest = TRUE))
R> for (i in 1:2) dati2[, i] <- ordered(cut(dataB[, i],
+     seq(min(dataB[, i], na.rm = TRUE), max(dataB[, i], na.rm = TRUE),
+     length = MyCut), include.lowest = TRUE))
R> lambda <- seq(0, 1, length = 50)
R> nl <- length(lambda)
R> set.seed(123)
R> D1 <- rrp.dist(dati1, msplit = 5, cut.in = 0, asdist = TRUE)
R> P1 <- 1 - as.matrix(D1)
R> px1 <- P1[tsubjects, csubjects]
R> S1 <- numeric(nl)
R> for (l in 1:nl) S1[l] <- sum(apply(px1, 1, function(x)
+     (length(which(x >= lambda[l])) > 0)))/nt
R> set.seed(123)
R> D2 <- rrp.dist(dati2, msplit = 5, cut.in = 0, asdist = TRUE)
R> P2 <- 1 - as.matrix(D2)
R> px2 <- P2[tsubjects, csubjects]
R> S2 <- numeric(nl)
R> for (l in 1:nl) S2[l] <- sum(apply(px2, 1, function(x)
+     (length(which(x >= lambda[l])) > 0)))/nt
```

In both cases, we calculated the curve $S(\lambda)$ whose graph is also represented in Figure 1.

```
R> par(mar = c(5, 4, 1, 1))
R> par(mfrow = c(2, 2))
R> plot(dataA, col = treated + 2, pch = ifelse(treated, 18, 17),
+     xlim = c(-1.5, 2), ylim = c(-1.5, 1.5))
R> rect(minX, minY, maxX, maxY, lty = 3)
R> plot(lambda, S1, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
R> plot(dataB, col = treated + 2, pch = ifelse(treated, 18, 17),
+     xlim = c(-1.5, 2), ylim = c(-1.5, 1.5))
R> plot(lambda, S2, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
```

For the non overlapping case $S(\lambda)$ goes toward zero as soon as $\lambda > 0$. In the top-left plot of Figure 1, a small rectangle was depicted. This rectangle contains the observations in the "plausible" common support defined as the intersection of the smallest rectangle including all units from group A and the smallest rectangle which includes all units from group B. A related notion of common support which involves the convex hull of the observations was introduced in King and Zeng (2006). The units in the plausible common support, coincide with the units actually matched by our method, i.e., for which the proximity is 1. Applying the *RANK*-

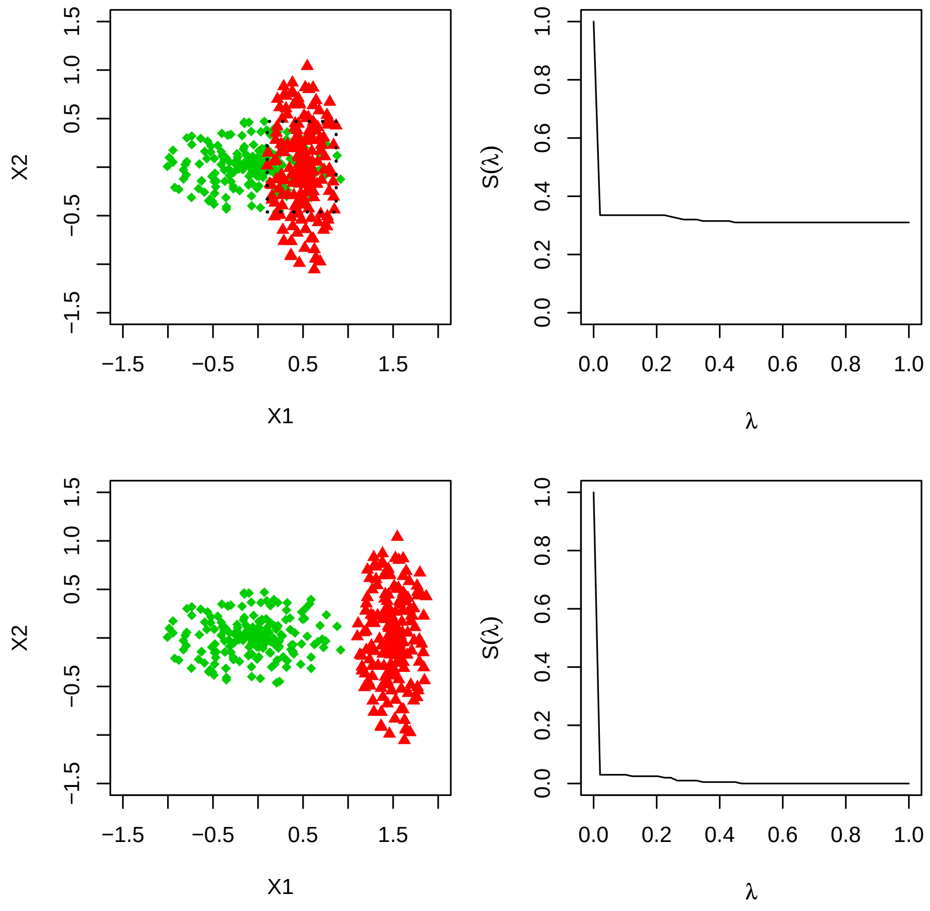

Figure 1: Overlapping (up) and disjoint (down) data sets. In the disjoint case, the curve $S(\lambda)$ rapidly goes to zero for $\lambda > 0$.

proximity algorithm on the same data sets in Figure 1 and using the same discretization, we obtain a new proximity matrix.

```
R> S <- function(lambda, P, group) {
+     g2 <- which(!group)
+     g1 <- which(group)
+     f <- function(x) {
+         if (lambda == 0)
+             return(TRUE)
+         nm <- as.numeric(names(x))
+         idx <- match(g2, nm)
+         idx <- as.numeric(na.omit(idx))
+         length(which(x[idx] >= lambda)) > 0
+     }
+     sum(as.numeric(lapply(P[g1], f)))/length(g1)
+ }
```

```
R> D3 <- rank.dist(dati1, asdist = FALSE)
R> gn <- function(x) S(x, D3, treated)
R> S3 <- sapply(lambda, gn)
R> D4 <- rank.dist(dati2, asdist = FALSE)
R> gn <- function(x) S(x, D4, treated)
R> S4 <- sapply(lambda, gn)
```

Figure 2 shows the curves $S(\lambda)$ for the same data sets obtained with RRP (left) and with rank-based algorithm (right). Plots are generated as follows

```
R> par(mar = c(5, 4, 1, 1))
R> par(mfrow = c(2, 2))
R> plot(lambda, S1, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
R> plot(lambda, S3, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
R> plot(lambda, S2, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
R> plot(lambda, S4, type = "l", ylim = c(0, 1), xlab = expression(lambda),
+     ylab = expression(S(lambda)))
```

What emerges is that the two proximity matrices contain the same information about the observations in the two data sets which actually match.

3.2. An econometric example

Many matching algorithms have been tested against the Lalonde data set. These data (LL in the following) are taken from the National Supported Work (NSW) Demonstration, a job training program implemented during the Seventies in the United States and analysed by Lalonde (1986). From April 1975 to August 1977 the program was carried out as a randomized experiment: some applicants were assigned to the program while some others, randomly chosen, were assigned to a control group and not allowed to participate to the program. The program provided training to the participants for 12-18 months and helped them in finding a job. As an effect, the program was supposed to yield an increase in the earnings of participants: therefore, real earnings in 1978 is the outcome variable of the analysis. Several pre-treatment variables were registered about the applicants (both participants and control individuals): age (`age`), years of education (`education`), marital status (`married`), lack of an high school diploma (`nodegree`), ethnic indicators (`black`, `hispanic`) and real earnings in 1974 (`re74`) and 1975 (`re75`). Two indicator variables `u74` and `u75` are included in the analysis to signal unemployement in 1974 and 1975. We will not discuss here the problem of the estimation of average treatment effect for these data which were considered by many authors (see Dehejia and Wahba 1999, 2002; Smith and Todd 2005a,b; Dehejia 2005) including Iacus and Porro (2007a,b) for the RRP algorithm. In this section, we just show the performance of the two methods proposed in this paper on the data set. In his study, Lalonde used several non-experimental control groups, coming from the Panel Study of Income Dynamics (PSID) and the Current Population Survey-Social Security Administration File (CPS) and tried to replicate the experimental target. He concluded that methods based on non-experimental data cannot correctly estimate the effect of the randomized experiment,

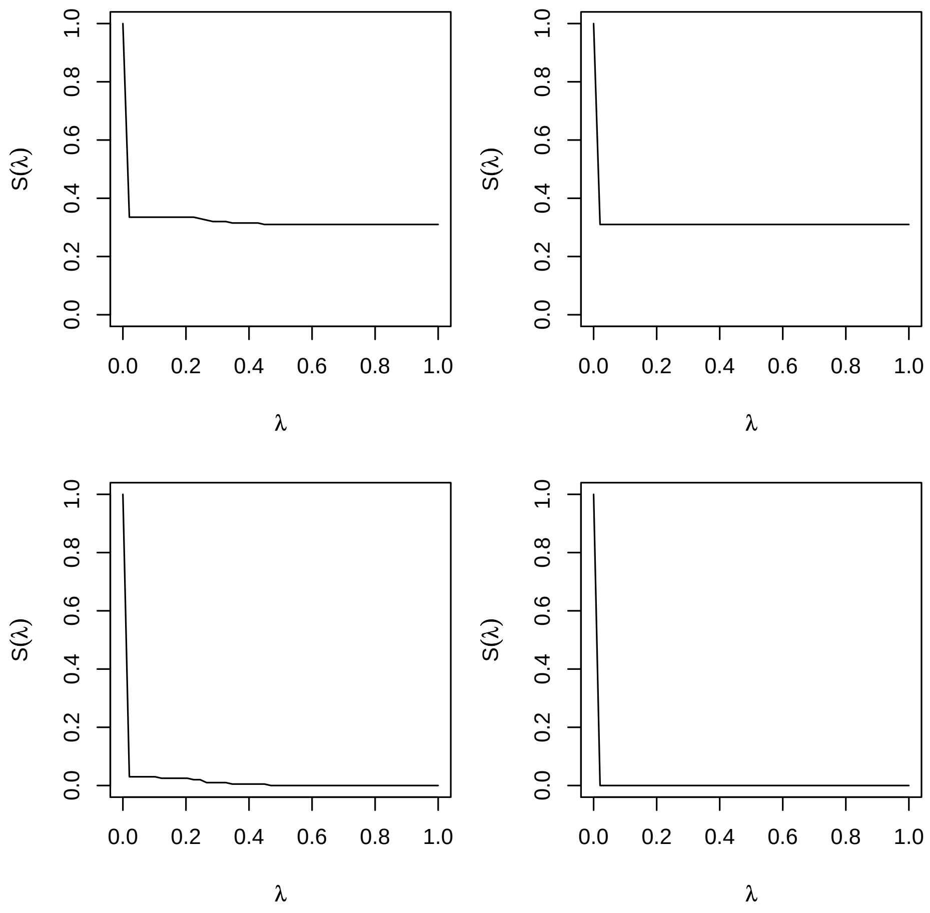

Figure 2: Overlapping (up) and disjoint (down) data sets. The curve $S(\lambda)$ obtained from the *RRP*-proximity (left) and from the *RANK*-proximity (right). The curvature is different, but both methods identify the same number of units in the common support.

consequently shedding some doubts on the reliability of these procedures. In more recent years, Dehejia and Wahba (1999, 2002) tried to show that the matching method based on propensity scores can be successful in replicating the experimental average treatment effect estimated by Lalonde even in a non-experimental context. To this aim, they selected from the experimental sample used by Lalonde a subsample (`DW` sample) made of 185 treated and 260 control units. As a non-experimental control group, they used a sample of 2490 units coming from `PSID`[6]. Many authors (see cited references) argued about the real possibility of matching the `DW` and `PSID` data. The application of the *RRP* and *RANK*-proximity provides evidence of the fact that `DW` and `PSID` data sets cannot be matched. Next R code performs such analysis:

```
R> require("rrp")
R> data("DWvsPSID")
```

[6]It is the PSID-1 sample used in Lalonde (1986).

```
R> n <- dim(DWvsPSID)[1]
R> ctr <- which(DWvsPSID$treated == 0)
R> trt <- which(DWvsPSID$treated == 1)
R> group <- (DWvsPSID$treated == 1)
R> DWvsPSID$u74 <- factor(DWvsPSID$re74 > 0)
R> DWvsPSID$u75 <- factor(DWvsPSID$re75 > 0)
R> DWvsPSID$black <- factor(DWvsPSID$black)
R> DWvsPSID$married <- factor(DWvsPSID$married)
R> DWvsPSID$nodegree <- factor(DWvsPSID$nodegree)
R> DWvsPSID$hispanic <- factor(DWvsPSID$hispanic)
```

```
R> str(DWvsPSID)
```

```
'data.frame':        2675 obs. of  12 variables:
 $ treated  : num  1 1 1 1 1 1 1 1 1 1 ...
 $ age      : int  33 33 35 42 22 27 22 42 41 35 ...
 $ education: int  12 12 9 9 12 13 12 14 14 8 ...
 $ black    : Factor w/ 2 levels "0","1": 1 2 2 2 1 1 1 2 1 2 ...
 $ married  : Factor w/ 2 levels "0","1": 2 2 2 2 1 2 2 2 1 2 ...
 $ nodegree : Factor w/ 2 levels "0","1": 1 1 2 2 1 1 1 1 1 2 ...
 $ re74     : num      0 20280 13602     0  6760 ...
 $ re75     : num      0 10941 13831  3059  8456 ...
 $ re78     : num  12418 15953 12804  1294 12591 ...
 $ hispanic : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 2 1 1 1 ...
 $ u74      : Factor w/ 2 levels "FALSE","TRUE": 1 2 2 1 2 2 2 1 1 2 ...
 $ u75      : Factor w/ 2 levels "FALSE","TRUE": 1 2 2 2 2 2 2 1 1 2 ...
```

We remove the group variable and `re78`

```
R> DWvsPSID <- DWvsPSID[-c(1, 9)]
```

We now calculate the *RANK*-proximity. A list will be returned.

```
R> px <- rank.dist(DWvsPSID, cut.in = 20)
```

and identify the twins between treated and control units:

```
R> thr <- 0.9
R> trt.m <- NULL
R> ctr.m <- NULL
R> for (i in trt) {
+     tmp <- as.numeric(names(which(px[[i]] > thr)))
+     tmp <- tmp[tmp %in% ctr]
+     if (length(tmp) > 0) {
+         trt.m <- c(trt.m, i)
+         ctr.m <- unique(c(ctr.m, tmp))
+     }
+ }
R> ctr.m <- sort(ctr.m)
R> trt.m <- sort(trt.m)
```

If we now look at the summary statistics for the pre-matched treated and control units

```
R> summary(DWvsPSID[trt, ])
```

```
      age          education     black   married nodegree      re74      
 Min.   :17.00   Min.   : 4.00   0: 29   0:150   0: 54    Min.   :    0  
 1st Qu.:20.00   1st Qu.: 9.00   1:156   1: 35   1:131    1st Qu.:    0  
 Median :25.00   Median :11.00                            Median :    0  
 Mean   :25.82   Mean   :10.35                            Mean   : 2096  
 3rd Qu.:29.00   3rd Qu.:12.00                            3rd Qu.: 1291  
 Max.   :48.00   Max.   :16.00                            Max.   :35040  
      re75       hispanic    u74         u75     
 Min.   :    0   0:174    FALSE:131   FALSE:111  
 1st Qu.:    0   1: 11    TRUE : 54   TRUE : 74  
 Median :    0                                   
 Mean   : 1532                                   
 3rd Qu.: 1817                                   
 Max.   :25142                                   
```

```
R> summary(DWvsPSID[ctr, ])
```

```
      age          education     black      married  nodegree      re74       
 Min.   :18.00   Min.   : 0.00   0:1866   0: 333   0:1730   Min.   :     0  
 1st Qu.:26.00   1st Qu.:11.00   1: 624   1:2157   1: 760   1st Qu.: 10776  
 Median :33.00   Median :12.00                              Median : 18417  
 Mean   :34.85   Mean   :12.12                              Mean   : 19429  
 3rd Qu.:44.00   3rd Qu.:14.00                              3rd Qu.: 26450  
 Max.   :55.00   Max.   :17.00                              Max.   :137149  
      re75        hispanic    u74          u75      
 Min.   :     0   0:2409   FALSE: 215   FALSE: 249  
 1st Qu.:  9847   1:  81   TRUE :2275   TRUE :2241  
 Median : 17903                                     
 Mean   : 19063                                     
 3rd Qu.: 26497                                     
 Max.   :156653                                     
```

and the post-match observations

```
R> summary(DWvsPSID[trt.m, ])
```

```
      age          education     black  married nodegree      re74      
 Min.   :17.00   Min.   : 8.00   0: 4   0:35    0:20     Min.   :    0  
 1st Qu.:21.00   1st Qu.:10.00   1:37   1: 6    1:21     1st Qu.:    0  
 Median :23.00   Median :11.00                              Median : 1291
 Mean   :23.32   Mean   :11.07                              Mean   : 3667
 3rd Qu.:25.00   3rd Qu.:12.00                              3rd Qu.: 5506
 Max.   :35.00   Max.   :12.00                              Max.   :20280
      re75          hispanic    u74        u75
 Min.   :    0.0   0:41      FALSE:17   FALSE:17
 1st Qu.:    0.0   1: 0      TRUE :24   TRUE :24
 Median :  334.0
 Mean   : 1974.3
 3rd Qu.: 2842.8
 Max.   :13830.6
```

```
R> summary(DWvsPSID[ctr.m, ])
```

```
      age          education      black   married nodegree      re74
 Min.   :18.00   Min.   : 8.00   0: 5    0:23    0:15     Min.   :    0
 1st Qu.:21.00   1st Qu.:10.00   1:27    1: 9    1:17     1st Qu.:    0
 Median :23.00   Median :11.00                            Median : 3037
 Mean   :24.25   Mean   :10.81                            Mean   : 4281
 3rd Qu.:25.25   3rd Qu.:12.00                            3rd Qu.: 6897
 Max.   :34.00   Max.   :12.00                            Max.   :18613
      re75       hispanic    u74        u75
 Min.   :    0   0:32     FALSE: 9   FALSE: 9
 1st Qu.:    0   1: 0     TRUE :23   TRUE :23
 Median : 3778
 Mean   : 3919
 3rd Qu.: 5662
 Max.   :16113
```

we can see the effectiveness of the match. After the match the distributions of treated and control units looks more similar between the two groups. Unfortunately, as it can be seen, the observations involved in the match are only a few of the original ones:

```
pre-match treated=185, controls=2490

post-match treated=41, controls=32
```

showing that reliable match can only be attained for an extremely small subset of the data. The same results can be obtained using the *RRP*-proximity as shown in Iacus and Porro (2007b).

4. A brief account about the software

This section is intended to be a short introduction to the **rrp** package for the R language (R Development Core Team 2008). For this reason, it reviews also functions related to applications of the proximities not discussed in the above: the interested reader can refer to Iacus and Porro

(2007a,b) for further details. We assume that the reader has some knowledge of the R language even if the code should appear understandable to non R experts too. The package is available from the Comprehensive R Archive Network at http://CRAN.R-project.org/package=rrp.

One of the main function of the package is `rrp.dist` which generates a dissimilarity matrix, say `D` (the proximity can be obtained as `1 - D`). This matrix can be used in further analysis. The `rrp.dist` function accepts several parameters. The ones relevant to the applications of this paper are

```
  rrp.dist(X, msplit = 10, Rep = 250, cut.in = 15, asdist = FALSE)
```

where `X` is the data matrix, `msplit` is the minimum split parameter[7] (by default 10) which corresponds to the same argument in the **rpart** package, `Rep` is the number of RRP replications (by default 250), `cut.in` is the number of equally spaced cut points (by default 15) used to split the support of numeric variables of `X`, i.e., the number of intervals equals to `cut.in -1`. The parameter `asdist` specifies if the return value should be an R object of class `dist`. If `asdist` is set to `FALSE` (the default value) the return value is an object of class `externalptr` (external pointer) for which coercion methods exist in the package. This choice was made to increase efficiency in presence of big data sets and to speed up the algorithm because the dist object is allocated in memory only once and it is not passed (and hence copied) back and forth between successive R calls. The function `XPtrToDist` converts an external pointer to a dist object and `newXPtr` creates a brand new external pointer object. For example `newXPtr(n, k = 0)` creates an external pointer of class `XPtr` (a class specific to the **rrp** package) as if it was a `dist` object of size `n` and initializes it with zeroes.

```
R> a <- newXPtr(10, 5)
R> a

<pointer: 0x8dabb90>
attr(,"class")
[1] "externalptr" "XPtr"
attr(,"Size")
[1] 10

R> str(a)

Classes 'externalptr', 'XPtr' <externalptr>
 - attr(*, "Size")= num 10
```

Then `XPtrToDist` can perform the conversion, i.e.:

```
R> (XPtrToDist(a))

   1 2 3 4 5 6 7 8 9
2  5
```

[7]The minimum split parameter is the minimal number of observations that must exist in a node, in order for a split to be attempted.

```
3  5 5
4  5 5 5
5  5 5 5 5
6  5 5 5 5 5
7  5 5 5 5 5 5
8  5 5 5 5 5 5 5
9  5 5 5 5 5 5 5 5
10 5 5 5 5 5 5 5 5 5
```

```
R> as.dist(matrix(5, 10, 10))
```

```
   1 2 3 4 5 6 7 8 9
2  5
3  5 5
4  5 5 5
5  5 5 5 5
6  5 5 5 5 5
7  5 5 5 5 5 5
8  5 5 5 5 5 5 5
9  5 5 5 5 5 5 5 5
10 5 5 5 5 5 5 5 5 5
```

The functions `addXPtr(d, x, k)` and `mulXPtr(d, x, k)` perform summation and multiplication of elements of the `XPtr` object allowing to specify indexes (almost) as if they were matrix in the following way:

```
R> M <- matrix(0, 5, 5)
R> d <- newXPtr(5, 0)
R> x <- list(1:3, 4:5)
R> addXPtr(d, x, c(-1, +1))
```

which is the equivalent of (apart for the diagonal elements)

```
R> M[1:3, 1:3] <- M[1:3, 1:3] - 1
R> M[4:5, 4:5] <- M[4:5, 4:5] + 1
```

Indeed, we have

```
R> (XPtrToDist(d))
```

```
   1  2  3  4
2 -1
3 -1 -1
4  0  0  0
5  0  0  0  1
```

```
R> as.dist(M)
```

```
   1  2  3  4
2 -1
3 -1 -1
4  0  0  0
5  0  0  0  1
```

The **rrp** package also contains the `addDist` function which applies to true `dist` objects and the functions `setDist` and `setXPtr` to set elements of `dist` and `XPtr` objects respectively. Also, the function `applyXPtr` which is a function of type `apply*` is available in the package.

The function `rank.dist` has a similar interface to `rrp.dist`. The most relevant parameters are the following:

```
rank.dist(X, cut.in = 0, thr = 0.75, weights, asdist = FALSE)
```

The `thr` argument is a threshold below which the value of the proximity is not retained by the algorithm. The argument `weights` is a vector of weights which by default is a vector of one's and `asdist` when set to `FALSE` (the default) returns a `list` with the same length of the observations. Each element of the list is a named vector containing the proximities greater or equal to `thr`. The names of the vector correspond to the row names of `X` for the observations for which the proximity is greater or equal `thr`. Each vector is also sorted in decreasing order to allow for fast nearest neighbor classification. The choice of returning a `list` instead of an external pointer was made because for the rank-based proximity (2) it is quite easy to choose a reasonable threshold for the proximities, hence the corresponding `dist` (or `XPtr`) object is essentially a sparse vector. In our experience, the `list` representation is more efficient in terms of memory occupation for big data sets.

4.1. Other useful functions of the package

We now show an example of classification on the Iris dataset. We remind that the Iris data set contains 4 measurement variables and 1 class variable (called `Species`). We randomly sample 10 observations as test set and use the remaining observations as training set. The dissimilarity matrix is built on the whole data set by excluding the class variable (which is variable number 5) from the data matrix.

```
R> require("rrp")
R> data("iris")
R> str(iris, strict = "cut", width = 70)

'data.frame':        150 obs. of  5 variables:
'data.frame':        150 obs. of  5 variables:
 $ Sepal.Length: num  5.1 4.9 4.7 4.6 5 5.4 4.6 5 4.4 4.9 ...
 $ Sepal.Width : num  3.5 3 3.2 3.1 3.6 3.9 3.4 3.4 2.9 3.1 ...
 $ Petal.Length: num  1.4 1.4 1.3 1.5 1.4 1.7 1.4 1.5 1.4 1.5 ...
 $ Petal.Width : num  0.2 0.2 0.2 0.2 0.2 0.4 0.3 0.2 0.2 0.1 ...
 $ Species     : Factor w/ 3 levels "setosa","versicolor",..: 1 1 1 ..

R> set.seed(123)
R> test <- sample(1:150, 10)
R> test
```

```
 [1]  44 118  61 130 138   7  77 128  79  65
```

```
R> train <- (1:150)[-test]
R> D <- rrp.dist(iris[, -5])
```

The function `rrp.dist` gives some feedback about the ongoing iterations (suppressed in this case). Once the matrix is obtained, we can use the `rrp.class` function to perform the nearest neighbor classification. The interface of the function is easy to use.

```
  rrp.class(x, cl, train, test, k = 1)
```

The function can be called by passing the dissimilarity matrix `D` (parameter `x`), the true class for the training set (`cl`), the vector of indexes corresponding to the training (`train`) and the test sets (`test`). By default the function performs a nearest neighbor classifier using `k=1` neighbors. As a result, a vector of predicted classes is returned and we tabulate it against the vector of true classes. The result is as follows:

```
R> pred <- rrp.class(D, iris$Species[train], train, test)
R> table(pred, iris$Species[test])
```

```
pred         setosa versicolor virginica
  setosa          2          0         0
  versicolor      0          4         0
  virginica       0          0         4
```

and, in this (rather fortunate) case, we obtain no missclassification. The package also contains the function `rrp.predict` function useful for continuous response variables. We provide here a working example using the birth weight data. The reader might want to refer to help for the data set for a description of the variables. The following code preprocesses the `birtwt` data in order to obtain a proper data matrix to handle. This example is borrowed from Venables and Ripley (2002).

```
R> require("MASS")
R> data("birthwt")
R> attach(birthwt)
R> race <- factor(race, labels = c("white", "black", "other"))
R> ptd <- factor(ptl > 0)
R> ftv <- factor(ftv)
R> levels(ftv)[-(1:2)] <- "2+"
R> bwt <- data.frame(bwt, age, lwt, race, smoke = (smoke > 0), ptd,
+     ht = (ht > 0), ui = (ui > 0), ftv)
R> detach()
R> rm(race, ptd, ftv)
```

Once the data are ready, we select the test and the training set and run `rrp.dist` (some output is omitted)

```
R> set.seed(123)
R> n <- dim(bwt)[1]
R> test <- sample(1:n, 15)
R> train <- (1:n)[-test]
R> D <- rrp.dist(bwt[, -1])
```

With the dissimilarity matrix in hands, we can proceed to prediction using `rrp.predict`, which has an interface consistent with `rrp.class`:

```
  rrp.predict(x, y, train, test, k = 1)
```

where the only difference is in that `y` is the vector of the response variable of the training set units. Therefore,

```
R> true.wht <- bwt$bwt[test]
R> pred.wht <- rrp.predict(D, bwt$bwt[train], train, test)
R> mean(pred.wht - true.wht)

[1] -28.2
```

where `-28.2` is the average bias after prediction.

The other function we review is `rrp.impute` which is used for data imputation. The function requires both the data matrix `data` containing missing values and the *RRP*-dissimilarity matrix `D`. If the dissimilarity matrix is not passed to the function, it will be calculated inside the function itself:

```
  rrp.impute(data, D = NULL, k = 1, msplit = 10, Rep = 250, cut.in = 15)
```

This function returns a copy of the matrix `data` after imputation, called `new.data`, and a copy of the dissimilarity matrix `D`. Data in made using a nearest neighbor algorithm on the proximity matrix. The following is an example of missing data imputation for the Iris data set in which 10 observations are chosen at random and for each of these 10, two missing values are imputed choosing randomly 2 over 5 covariates:

```
R> data("iris")
R> X <- iris
R> n <- dim(X)[1]
R> set.seed(123)
```

we generate missing data in 10 observations

```
R> miss <- sample(1:n, 10)
R> for (i in miss) X[i, sample(1:5, 2)] <- NA
R> X[miss, ]

    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
44           5.0          NA          1.6         0.6       <NA>
118          7.7         3.8           NA          NA  virginica
```

```
61              NA         2.0          3.5          NA versicolor
130             NA          NA          5.8         1.6  virginica
138            6.4          NA          5.5          NA  virginica
7              4.6         3.4           NA         0.3       <NA>
77             6.8         2.8          4.8          NA       <NA>
128            6.1         3.0           NA          NA  virginica
79             6.0         2.9           NA         1.5       <NA>
65              NA          NA          3.6         1.3 versicolor
```

We now run the `rrp.impute` function on the data and look at the data (`x$new.data`) after imputation:

```
R> x <- rrp.impute(X)
R> x$new.data[miss, ]
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
44           5.0         3.4          1.6         0.6     setosa
118          7.7         3.8          6.4         2.0  virginica
61           5.0         2.0          3.5         1.0 versicolor
130          7.2         3.2          5.8         1.6  virginica
138          6.4         2.8          5.5         2.1  virginica
7            4.6         3.4          1.6         0.3     setosa
77           6.8         2.8          4.8         1.7 versicolor
128          6.1         3.0          4.8         1.8  virginica
79           6.0         2.9          4.2         1.5 versicolor
65           5.8         2.7          3.6         1.3 versicolor
```

It is interesting to look at the original full data as well:

```
R> iris[miss, ]
```

```
    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
44           5.0         3.5          1.6         0.6     setosa
118          7.7         3.8          6.7         2.2  virginica
61           5.0         2.0          3.5         1.0 versicolor
130          7.2         3.0          5.8         1.6  virginica
138          6.4         3.1          5.5         1.8  virginica
7            4.6         3.4          1.4         0.3     setosa
77           6.8         2.8          4.8         1.4 versicolor
128          6.1         3.0          4.9         1.8  virginica
79           6.0         2.9          4.5         1.5 versicolor
65           5.6         2.9          3.6         1.3 versicolor
```

These examples can be run directly using the commands `example` section of each R command in the package **rrp**. The functions `rank.class`, `rank.predict` and `rank.impute` have not yet been implemented in the current version of the **rrp** package (2.6) although there is a plan to write these functions for the *RANK*-proximity.

References

Cochran W, Rubin DB (1973). "Controlling Bias in Observational Studies: A Review." *Sankhya A*, **35**, 417–446.

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

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

Dehejia R, Wahba S (2002). "Propensity Score Matching Methods for Non-Experimental Causal Studies." *Review of Economics and Statistics*, **84**(1), 151–161.

Diamond A, Sekhon JS (2005). "Genetic Matching for Estimating Causal Effects: A General Multivariate Matching Method for Achieving Balance in Observational Studies." URL `http://sekhon.polisci.berkeley.edu/papers/GenMatch.pdf`.

Gu XS, Rosenbaum PR (1993). "Comparison of Multivariates Matching Methods: Structures, Distances and Algorithms." *Journal of Computational and Graphical Statistics*, **2**, 405–420.

Hansen BB (2004). "Full Matching in an Observational Study of Coaching for the SAT." *Journal of the American Statistical Association*, **99**, 609–618.

Heckman J, Ichimura H, Smith J, Todd P (1998). "Characterizing Selection Bias Using Experimental Data." *Econometrica*, **66**(5), 1017–1098.

Heckman J, Ichimura H, Todd P (1997). "Matching as Econometric Evaluation Estimator: Evidence From Evaluating a Job Training Programme." *Review of Economic Studies*, **64**(4), 605–654.

Ho D, Imai K, King G, Stuart E (2007). "Matching as Nonparametric Preprocessing for Reducing Model Dependence in Parametric Causal Inference." *Political Analysis*, **15**(3), 199–236.

Iacus SM, Porro G (2007a). "Missing Data Imputation, Matching and Other Applications of Random Recursive Partitioning." *Computational Statistics and Data Analysis*, **52**(2), 773–789.

Iacus SM, Porro G (2007b). "Random Recursive Partitioning: A Matching Method for the Estimation of the Average Treatment Effect." *Journal of Applied Econometrics*. Forthcoming.

King G, Zeng L (2006). "The Dangers of Extreme Counterfactuals." *Political Analysis*, **14**(2), 131–159.

Lalonde R (1986). "Evaluating the Econometric Evaluations of Training Programs." *American Economic Review*, **76**, 604–620.

Little RJA, Rubin DB (2002). *Statistical Analysis with Missing Data*. John Wiley & Sons, Hoboken, NJ.

R Development Core Team (2008). *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, URL http://www.R-project.org/.

Rosenbaum P (1991). "A Characterization of Optimal Designs for Observational Studies." *Journal of the Royal Statistical Society, Series B*, **53**, 597–610.

Rosenbaum PR (1989). "Optimal Matching in Observational Studies." *Journal of the American Statistical Association*, **84**, 1024–1032.

Rosenbaum PR, Rubin DB (1983). "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika*, **270**, 41–55.

Rubin DB (1973a). "Matching to Remove Bias in Observational Studies." *Biometrics*, **29**, 159–183.

Rubin DB (1973b). "The Use of Matched Sampling and Regression Adjustment to Remove Bias in Observational Studies." *Biometrics*, **29**, 185–203.

Smith J, Todd P (2005a). "Does Matching Overcome Lalonde's Critique of Nonexperimental Estimators?" *Journal of Econometrics*, **15**(1-2), 305–353.

Smith J, Todd P (2005b). "Rejoinder (to Dehejia, 2005)." *Journal of Econometrics*, **125**(1-2), 365–375.

Venables WN, Ripley BD (2002). *Modern Applied Statistics with S*. Springer-Verlag, New York, 4th edition.

Affiliation:

Stefano Maria Iacus
Department of Economics, Business and Statistics
University of Milan
Via Conservatorio 7, I-20122 Milano, Italy
E-mail: stefano.iacus@unimi.it
URL: http://www.economia.unimi.it/iacus/

Giuseppe Porro
Department of Economics and Statistics
University of Trieste
P.le Europa 1, I-34127 Trieste, Italy
E-mail: giuseppe.porro@econ.univ.trieste.it

Journal of Statistical Software http://www.jstatsoft.org/
published by the American Statistical Association http://www.amstat.org/
Volume 25, Issue 11 *Submitted:* 2007-03-19
April 2008 *Accepted:* 2008-03-31