MatchIt: Nonparametric Preprocessing for Parametric Causal Inference

Daniel E. Ho,[1] Kosuke Imai,[2] Gary King,[3] Elizabeth A. Stuart[4]

August 16, 2005

[1]Institute for Quantitative Social Science, 34 Kirkland, Cambridge MA 02138, USA; http://people.iq.harvard.edu/ dho/, daniel.e.ho@gmail.com.

[2]Assistant Professor, Department of Politics, Princeton University (Corwin Hall 041, Department of Politics, Princeton University, Princeton NJ 08544, USA; http://imai.princeton.edu, kimai@Princeton.Edu).

[3]David Florence Professor of Government, Harvard University (Institute for Quantitative Social Science, 34 Kirkland Street, Harvard University, Cambridge MA 02138; http://GKing.Harvard.Edu, King@Harvard.Edu, (617) 495-2027).

[4]Researcher, Mathematica Policy Research, Inc. Ph.D. Department of Statistics, Harvard University. (600 Maryland Ave., SW, Suite 550, Washington, DC 20024, USA; http://www.mathematica-mpr.com, EStuart@Mathematica-MPR.com).

Contents

1 Introduction

MatchIt implements the suggestions of Ho, Imai, King and Stuart (2005) for improving parametric statistical models and reducing model dependence by preprocessing data with semi- and non-parametric matching methods. After preprocessing with MatchIt, researchers can use whatever parametric model they would have used without MatchIt, but produce inferences that are substantially more robust and less sensitive to modeling assumptions. Matched data sets created by MatchIt can also be entered easily into Zelig (Imai, King and Lau, 2004), or other programs, for subsequent parametric analyses. MatchIt reduces the dependence of causal inferences on commonly made, but hard-to-justify, statistical modeling assumptions via a wide range of sophisticated matching methods. In addition, we have written MatchIt so that adding methods to its structure is easy, if you have the inclination.

1.1 Software Requirements

MatchIt works in conjunction with the R programming language and statistical software, and will run on any platform where R is installed (Windows, Unix, or Mac OS X). R is available free for download at the Comprehensive R Archive Network (CRAN) at http://www.r-project.org/. MatchIt has been tested on the most recent version of R (2.1.1). A good way to learn R, if you don't know it already, is to learn Zelig (available at http://gking.harvard.edu) which includes a self-contained introduction to R and can then be used to analyze the data after running MatchIt.

1.2 Installing MatchIt

To install MatchIt for all platforms, type at the R command prompt:

```
> install.packages("MatchIt")
```

and MatchIt will install itself onto your system automatically. During the process you may either decide to keep or discard the installation files, which will not affect the way MatchIt runs. You only need to do this once. Some users may also find it helpful to install the package with version control (see Subsection 6.4).

1.3 Loading MatchIt

As with any R package, you need install MatchIt only once, but you must load it prior to each use. You can do this for each R session by typing

```
> library(MatchIt)
```

at the R command prompt.

Alternatively, you can specify R to load MatchIt automatically at launch so that you can skip the step of typing `library(MatchIt)` at the beginning of every R session. To do this, edit the `Rprofile` file located in the R program subdirectory, e.g. `C:/R/rw2011/etc/`, for Windows systems or the `.Rprofile` file located in the home directory for Unix/Linux and Mac OS X systems. Using a text editor such as Windows notepad and emacs, add the following line to the file:

```
> options(defaultPackages = c(getOption("defaultPackages"), "MatchIt"))
```

For this change to take effect, you need to restart R.

1.4 Updating MatchIt

We recommend that you periodically update MatchIt at the R prompt by typing:

```
> update.packages()
> library(MatchIt)
```

which will update all the libraries including MatchIt.

2 Statistical Overview

MatchIt is designed for studies with a dependent variable (or a set of dependent variables) that is a function of a dichotomous causal (or "treatment") variable, with values known as the "treated" and "control" groups, and a set of "pretreatment" covariates, i.e., that are causally prior to the administration of the treatment. MatchIt works for experimental data, but is also designed for observational designs where the treatment variable is observed rather than manipulated by the investigator. MatchIt can be used for other types of causal variables by dichotomizing them, perhaps in multiple ways (see also Imai and van Dyk, 2004).

2.1 Preprocessing via Matching

The goal of matching is to adjust the data prior to the parametric analysis so that (1) the relationship between t_i and X_i is eliminated or reduced, and (2) no bias and little inefficiency is induced. This means that during matching we can select, duplicate, or selectively drop observations from an existing sample without inducing bias, as long as we do so using a rule that is a function only of t_i and X_i. MatchIt provides, implements, and evaluates the choice of these rules. After matching, the preprocessed dataset will be such that t_i and X_i are unrelated or less related than in the raw data, meaning that the treatment and control groups have the same background characteristics, or in other words that this relationship holds: $\tilde{p}(X|t = 1) \approx \tilde{p}(X|t = 0)$, where $\tilde{p}$ refers to the observed empirical density of the data (think histogram), rather than a population density.

The simplest way to obtain good matches (as defined above) is to use one-to-one exact matching, which pairs each treated unit with one control unit for which the values of X_i are identical. However, with many covariates and finite numbers of potential matches, it is often very difficult to obtain exact matches. Luckily, good matching only requires that the *distribution* of X given $t = 0$ match the distribution of X given $t = 1$, and so individual matched pairs are not required. Indeed, many of the other methods implemented in MatchIt only attempt to balance the overall covariate distributions, without necessarily finding one-to-one exact matches.

2.2 Checking Balance

The goal of a matching procedure is to replicate a randomized experiment in terms of finding units who look as if they could have been randomly assigned to treatment or control status. That is, we want the distribution of covariates to be the same in the matched treated and matched control groups. Thus, a crucial part of any matching procedure is assessing how close the matches are in the two groups (the "balance"). Because the outcome variable is not used in the matching procedure, a variety of matching methods can be assessed, and the matching procedure that leads to the best balance chosen. MatchIt provides a number of ways to assess the balance of covariates after matching, including numerical summaries such as the standardized bias between the groups (difference in means divided by the treated group standard deviation) and graphical summaries such as quantile-quantile plots that compare the empirical distributions of each covariate. These diagnostics can be done on the covariates included in the matching procedure, as well as on other covariates on which close matches are desired.

2.3 Conducting Analyses after Matching

The most common way that parametric analyses are used to compute quantities of interest is by holding constant some explanatory variables, changing others, and computing predicted or expected values and taking the difference or ratio. In the case of causal inference, this would probably mean looking at the effect on the expected value of the outcome variable when changing T from 0 to 1, while holding constant the pretreatment control variables X at their their means or medians. This, and indeed any other ordinary procedure, would be a perfectly reasonable way to proceed with analysis after matching.

Another way to proceed with analysis after MatchIt is to compute the average treatment effect on the treated. For example, for the treated group, the potential outcomes under control, Y_{0i}, are missing, whereas

the outcomes under treatment, Y_{1i}, are observed, and the goal of the analysis is to impute the missing outcomes, Y_{0i} in observations where $T_i = 1$, which we do via simulation using a parametric statistical model (as described below). Once those potential outcomes are imputed from the model, the estimate of individual i's treatment effect is $Y_{1i} - \widehat{Y}_{0i}$ where $\widehat{Y}_{0i}$ is a simulation of the average missing potential outcome for unit i (that is it is a simulation of the value of the dependent variable for unit i under the counterfactual condition where $T_i = 0$). The in-sample average treatment effect for the treated individuals can then be obtained by averaging this difference over all observations i where in fact $T_i = 1$. (A similar procedure can also be used to estimate various other quantities of interest such as the average treatment effect for all observations.) An advantage of this simulation approach is that the uncertainty estimates such as standard errors and confidence intervals are obtained easily by the usual rules in fitting the parametric model.

The imputation from the model can be done in at least two ways. Recall that the model is used to impute *the value that the outcome variable would take among the treated units if those treated units were actually controls*. Thus, one reasonable approach would be to fit a model to the whole set of matched data and create simulated predicted values of the dependent variable for the treated units with T_i switched counterfactually from 1 to 0. An alternative approach would be to fit a model without T by using only the outcomes of the matched control units (i.e., using only observations where $T_i = 0$). Then, given this fitted model, the missing outcomes Y_{i0} are imputed for the matched treated units by using the values of the explanatory variables for the treated units. The first approach will usually have lower variance, since all observations are used, and the second may have less bias, since no assumption of constant parameters is needed. See (Ho et al., 2005) for more details.

3 User's Guide to MatchIt

We adopt the same notation as in Ho, Imai, King and Stuart (2005). Unless otherwise noted, let i index the n units in the dataset, n_1 denote the number of treated units, n_0 denote the number of control units (such that $n = n_0 + n_1$), and x_i indicate a vector of pretreatment (or control) variables for unit i. Let $t_i = 1$ when unit i is assigned treatment, and $t_i = 0$ when unit i is assigned control. (The labels "treatment" and "control" are arbitrary and can be switched for convenience.) Denote y_{1i} as the potential outcome of unit i under treatment and y_{0i} the potential outcome of unit i under control. The variables y_{1i} and y_{0i} are jointly unobservable, and for each i, we observe one $y_i = t_i y_{1i} + (1 - t_i) y_{0i}$, and not the other.

3.1 Example Data: National Supported Work Demonstration

For a running example, we use data from the job training program analyzed in Lalonde (1986) and Dehejia and Wahba (1999). A subsample of the data consisting of the National Supported Work Demonstration (NSW) treated group and the comparison sample from the Population Survey of Income Dynamics (PSID) is included in MatchIt.[1] The variables in this dataset are described in Table 1. One causal effect of interest is the impact that participation in the job training program, `treat == 1`, had on real earnings in 1978, `re78`, for those that participated in the program. The average treatment effect on the treated (ATT) is defined as,

$$\text{ATT} = \frac{1}{\sum_{i=1}^{n} t_i} \sum_{i:t_i=1} E[Y_i(1) - Y_i(0)], \tag{1}$$

where $Y_i(1)$ represents the potential outcome under the treatment of the job program, and $Y_i(0)$ under control. Note that the first term inside the expectation (the right hand side of Equation 1) is *observed*, whereas the second term is the *unobserved* counterfactual of real earnings if participants had not participated. The nature of causal inference is that one of the two terms in the difference will always be unobserved.

[1] This data set, `lalonde`, was created using NSWRE74_TREATED.TXT and CPS3_CONTROLS.TXT from http://www.columbia.edu/∼rd247/nswdata.

Variable name	Description
Outcome variable (Y)	
`re78`	Real earnings (1978)
Treatment variable (T)	
`treat`	Treated in job training program from March 1975-June 1977 (1 if treated, 0 if not treated)
Pre-treatment covariates (X)	
`age`	Age
`educ`	Years of education
`black`	Race black (1 if black, 0 otherwise)
`hispan`	Race hispanic (1 if Hispanic, 0 otherwise)
`married`	Marital status (1 if married, 0 otherwise)
`nodegree`	High school degree (1 if no degree, 0 otherwise)
`re74`	Real earnings (1974)
`re75`	Real earnings (1975)

Table 1: Description of Lalonde data

3.2 Preprocessing via Matching

`matchit()` implements a variety of matching procedures. A general syntax is

```
> m.out <- matchit(treat ~ x1 + x2, data = mydata)
```

where `treat` is the dichotomous treatment variable, and `x1` and `x2` are pre-treatment covariates. This command creates the MatchIt object called `m.out`. A quick summary of the matching procedure can be printed on the screen by typing

```
> m.out
```

or

```
> print(m.out)
```

Examples of various matching procedures are given for the reminder of this section, and Section 4.1 lists all possible specifications of `matchit()`.

3.2.1 Exact Matching

The simplest version of matching is exact. This technique matches *each* treated unit to *all* possible control units with exactly the same value on all the covariates, forming subclasses such that within each subclass all units (treatment and control) have the same covariate values. Exact restrictions on a subset of covariates can also be specified in nearest neighbor matching (see Section 3.2.3).

The following example script can be run by typing `demo(exact)` at the R prompt:

```
> library(MatchIt)
> data(lalonde)
> m.out <- matchit(treat ~ educ + black + hispan, data = lalonde,
+     method = "exact")
```

The object `m.out` contains all the information on the matched units. The matching forms all possible subclasses based on unique values of the covariates in the right hand side of the formula; within each subclass, all units have the same covariate values. To obtain basic information about the matching procedure:
print a short summary

```
> m.out

Call: matchit(formula = treat ~ educ + black + hispan, data = lalonde,     method = "exact")

Exact Subclasses: 25

Sample sizes:
          Control Treated
Full          429     185
Matched       376     184
Unmatched      53       1
```

The printout includes the original call to MatchIt and the fact that 184 treatment units were exactly matched to 376 control units on race and education. There was 1 treated unit and 53 control units that weren't matched, and therefore are excluded from the matched data set. See Sections 4.1.1 and 4.1.2 for a list of complete input options for exact matching. Now, proceed to Section 3.3 to learn about how the `summary()` command can be used with exact matching.

3.2.2 Subclassification

When there are many covariates (or many values of some covariates) on which matches are desired, finding sufficient exact matches will often be impossible. In that case, subclassification is sometimes desirable. Various subclassification schemes exist, including the one based on a scalar distance measure such as the estimated propensity score using the `distance` option (see Section 4.1.1). Subclassification will form subclasses based on this distance measure. Within each subclass, the distribution of covariates in the treatment and control groups should be similar.

Subclassification is implemented in MatchIt using `method = "subclass"`. See also the sections on full matching (Section 3.2.4) and nearest-neighbor matching (Section 3.2.3), which provide additional ways of performing subclassification. The following example script can be run by typing `demo(subclass)` at the R prompt:

```
# Run the default subclassification scheme

> m.out <- matchit(treat ~ re74 + re75 + educ + black + hispan +
+     age, data = lalonde, method = "subclass")
```

The above syntax forms forms 6 subclasses, which is the default number of subclases, based on a distance measure estimated using logistic regression. By default, each subclass will have approximately the same number of treated units. See Sections 4.1.1 and 4.1.3 for a complete list of input options for subclassification.

Subclassification may also be used in conjunction with nearest neighbor matching described below in Subsection 3.2.3, by leaving the default of `method = "nearest"` but adding the option `subclass`. When you choose this command, MatchIt matches in the same way, but after the nearest neighbor matches are chosen it places them into subclasses, and adds a variable to the output object with the subclass numbers.

3.2.3 Nearest Neighbor Matching

Nearest neighbor matching selects the r best control matches for each individual in the treatment group (excluding those discarded using the `discard`) option. The matching is done using a distance measure specified by the `distance` option. Matches are chosen for each treated unit one at a time, and at each matching step we choose the control unit that is not yet matched but is closest to the treated unit on the distance measure. There are many variations on nearest neighbor matching, which are described in further detail in Sections 4.1.1 and 4.1.4.

Nearest neighbor matching is implemented in MatchIt using the `method="nearest"` option. The following example script can be run by typing `demo(nearest)` at the R prompt.

To conduct propensity score matching with pre-treatment covariates composed of real earnings in 1974 and 1975, education, race, and age:

```
> m.out <- matchit(treat ~ re74 + re75 + educ + black + hispan +
+     age, data = lalonde, method = "nearest")
```

You may again check basic statistics of the MatchIt object by the `print` command:

```
> print(m.out)

Call: matchit(formula = treat ~ re74 + re75 + educ + black + hispan +     age, data = lalonde, method =

Sample sizes:
          Control Treated
Full          429     185
Matched       185     185
Unmatched     244       0
Discarded       0       0
```

We see that 185 control units were matched to the 185 treated units (a "1-1" match).

Additional Examples Here, we illustrate various options of nearest neighbor matching by providing additional examples based on the Lalonde data. Users should refer to Sections 4.1.1 and 4.1.4 for a complete list of options for nearest neighbor matching.

1. Nearest neighbor matching on propensity score estimated using logistic regression where 2 matches are chosen for each treated unit.

   ```
   > m.out1 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
   +                   distance = "logit", ratio=2)
   ```

2. Mahalanobis matching on `re74` and `re75`.

   ```
   > m.out <- matchit(treat ~ re74+re75, data=lalonde, method="nearest",
   +      distance="mahalanobis")
   ```

3. Mahalanobis matching on `re74` and `re75` within nearest neighbor matching on distance measure (propensity score), with restriction of exact matches on `married`.

   ```
   > m.out2 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
   +                   distance = "logit", mahvars=c("re74", "re75"), exact=c("married"),
   +                   caliper=.25)
   ```

4. Nearest neighbor matching after discarding all units outside of the common support of the estimated distance measure.

   ```
   > m.out3 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
   +                   distance = "logit", discard= "both")
   ```

5. Nearest neighbor matching with replacement where two control units are matched with one treated unit.

   ```
   > m.out4 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
   +                   distance = "logit", replace = TRUE, ratio = 2)
   ```

6. Nearest neighbor matching followed by formation of 5 subclasses

   ```
   m.out5 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
                     distance = "logit", subclass=5)
   ```

3.2.4 Full Matching

Full matching is a a particular type of subclassification that uses all treated and control units (Rosenbaum, 2002; Hansen, 2004). A fully matched sample is composed of matched sets, where each matched set contains either one treated unit and one or more controls (or one control unit and one or more treated units). The only units not placed into a subclass will be those discarded (if a `discard` option was specified) because they are outside of common support. Full matching is optimal in terms of minimizing a weighted average of the estimated distance measure between each treated subject and each control subject within each subclass. See Sections 4.1.1 and 4.1.5 for a complete list of optional inputs for full matching.

Full matching can be performed with MatchIt by setting `method = "full"`. We use an add-on package called `optmatch` (Hansen, 2004), which must be installed separately by typing at the R command prompt,

```
> install.packages("optmatch", contriburl = "http://www.stat.lsa.umich.edu/~bbh/optmatch")
```

For more information about the package, see http://www.stat.lsa.umich.edu/~bbh/optmatch.html. The following example script can be run by typing `demo(full)` at the R prompt. We first load the Lalonde data, confuct full matching, and then print a short summary.
conduct full matching using the propensity score based on logistic regression

```
> m.out <- matchit(treat ~ age + educ + black + hispan + married +
+     nodegree + re74 + re75, data = lalonde, method = "full",
+     distance = "logit")
> m.out

Call: matchit(formula = treat ~ age + educ + black + hispan + married +     nodegree + re74 + re75, data

Sample sizes:
          Control Treated
Full          429     185
Matched       429     185
Unmatched       0       0
Discarded       0       0
```

We see that the matching has utilized all units (the treated and control group sample sizes are the same for the full and matched samples).

3.2.5 Optimal Matching

The default nearest neighbor matching method in MatchIt is "greedy" matching, where the closest control match for each treated unit is chosen one at a time, without trying to minimize a global distance measure. Another method, "optimal" matching, finds the matched samples with the smallest average absolute distance between each matched pair. With large control pools, greedy and optimal matching may lead to very similar (or the same) sets of matches; Gu and Rosenbaum (1993) have shown that greedy and optimal matching generally choose the same sets of controls for the matched samples, but that optimal matching does a better job of minimizing the distance within each pair. In addition, optimal matching can be particularly helpful when there are not many appropriate control matches for the treated units. See Gu and Rosenbaum (1993) or Rosenbaum (2002) for more information on optimal matching. See Sections 4.1.1 and 4.1.6 for a complete list of optional inputs for optimal matching.

Optimal matching is performed with MatchIt by setting `method = "optimal"`. We use an add-on package called `optmatch` (Hansen, 2004), which must be installed separately by typing at the R command prompt,

```
> install.packages("optmatch", contriburl = "http://www.stat.lsa.umich.edu/~bbh/optmatch")
```

For more information about the package, see http://www.stat.lsa.umich.edu/~bbh/optmatch.html. The following example script can be run by typing `demo(optimal)` at the R prompt.
optimal ratio matching using the propensity score based on logistic regression

```
> m.out <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde,
+     method = "optimal", distance = "logit", ratio = 2)
> m.out

Call: matchit(formula = treat ~ re74 + re75 + age + educ, data = lalonde,     method = "optimal", distar

Sample sizes:
          Control Treated
Full          429     185
Matched       370     185
Unmatched      59       0
Discarded       0       0
```

3.3 Checking Balance

In MatchIt, there are two ways to check balance after any matching procedure. Given the MatchIt output object `m.out`, one can use the following commands

1. `summary(m.out)` gives numerical summaries of the resulting balance of covariates.
2. `plot(m.out)` gives graphical summaries to assess balance of covariates.

Below, we give three examples (exact matching, nearest neigbor matching, and sublassification) to illustrate the use of these two functions. Similar commands can be used for all other matching procedures, and they will give similar outputs. See Section 4.2 for details.

Examples

1. Using the example of exact matching from Section 3.2.1, we obtain more information on the matching using the `summary()` command. To also show the covariates values in each subclass, we use the `covariates = TRUE` option:

   # balance diagnostics through statistics

   ```
   > m.out <- matchit(treat ~ educ + black + hispan, data = lalonde,
   +     method = "exact")
   > summary(m.out, covariates = TRUE)

   Call:
   matchit(formula = treat ~ educ + black + hispan, data = lalonde,     method = "exact")


   Sample sizes for full and exactly matched data:

             Control Treated
   Full          429     185
   Matched       376     184
   Discarded      53       1

   Matched sample sizes by subclass:
   ```

```
   Treated Control Total educ black hispan
1       39      13    52   11     1      0
2        4      10    14    9     0      1
3       30      23    53   12     1      0
4       15       8    23    8     1      0
5       22      13    35    9     1      0
6        1       2     3   16     1      0
7        7      83    90   12     0      0
8        7       3    10   13     1      0
9       28      11    39   10     1      0
10       1       2     3    7     1      0
11       1      13    14   14     0      0
12       2      20    22    9     0      0
13       2       5     7   10     0      1
14       4       2     6    4     1      0
15       4       2     6   14     1      0
16       3       3     6    5     1      0
17       2       7     9    8     0      1
18       4      29    33   11     0      0
19       1      21    22    7     0      0
20       1       2     3    6     1      0
21       1      14    15   13     0      0
22       2      12    14   12     0      1
23       1      29    30    8     0      0
24       1      40    41   10     0      0
25       1       9    10   11     0      1
```

MatchIt has created 25 subclasses where the values of the race and education covariates are equal. The MatchIt output object will include `weights` and `subclass`, which can be used to identify which units were put into the same subclass. Units that did not have the same covariate values as anyone in the other treatment group have `subclass = NA`.

For example, we can use the `sublass` component of `m.out` to identify which units are in each subclass and to verify their covariate values. To see ID numbers of the first 10 units in subclass 1, in which all subjects are black and have 11 years of education ("NSW" refers to the National Supported Work Demonstration participants):

```
> row.names(m.out$X)[m.out$subclass == 1][1:10]

 [1] "NSW1"  "NSW4"  "NSW8"  "NSW21" "NSW24" "NSW26" "NSW27" "NSW30" "NSW33"
[10] "NSW45"
```

We can also confirm the covariate values of the units in each subclass. For example, to see the covariate values of the first 5 units in subclass 2 ("PSID" refers to individuals in the Panel Survey of Income Dynamics):

```
> m.out$X[m.out$subclass == 2 & !is.na(m.out$subclass), ][1:5,
+     ]

       educ black hispan
NSW2      9     0      1
NSW100    9     0      1
NSW112    9     0      1
```

```
NSW173    9     0      1
PSID9     9     0      1
```

2. For all other types of matching, the `summary()` command first gives measures of the balance between the treated and control groups in the full (original) data set, and then the same measures of balance in the matched data set. If the matching worked well, the measures of balance should be smaller in the matched data set as compared with the full data set.

   To illustrate this we use the example of nearext neighbor matching from Section 3.2.3:

```
> m.out <- matchit(treat ~ re74 + re75 + educ + black + hispan +
+     age, data = lalonde, method = "nearest")
> summary(m.out)

Call:
matchit(formula = treat ~ re74 + re75 + educ + black + hispan +     age, data = lalonde, method = "

Summary of balance for all data:

         Means Treated Means Control Treated SD Std. Bias   QQ Med  QQ Mean
distance         0.566         0.187      0.211     1.792    0.732    0.535
re74          2095.574      5619.237   4886.620    -0.721 3430.277 5155.851
re75          1532.055      2466.484   3219.251    -0.290 1373.280 1511.706
educ            10.346        10.235      2.011     0.055    1.414    0.962
black            0.843         0.203      0.365     1.757    1.414    0.903
hispan           0.059         0.142      0.237    -0.349    0.000    0.119
age             25.816        28.030      7.155    -0.309    1.414    4.650
            QQ Max
distance     0.840
re74     13044.159
re75      9609.595
educ         5.657
black        1.414
hispan       1.414
age         14.142

Summary of balance for matched data:

         Means Treated Means Control Treated SD Std. Bias  QQ Med QQ Mean
distance         0.566         0.365      0.211     0.951   0.150   0.284
re74          2095.574      2466.304   4886.620    -0.076 410.082 978.726
re75          1532.055      1960.355   3219.251    -0.133 638.038 887.982
educ            10.346        10.470      2.011    -0.062   1.414   1.307
black            0.843         0.470      0.365     1.023   0.000   0.527
hispan           0.059         0.276      0.237    -0.912   0.000   0.306
age             25.816        26.054      7.155    -0.033   2.828   3.868
            QQ Max
distance     0.693
re74     18556.957
re75     16073.541
educ         5.657
black        1.414
hispan       1.414
```

```
age         11.314

Percent Balance Improvement:

          Mean and Std. Bias  QQ Med QQ Mean QQ Max
distance              46.94   79.43   46.79  17.59
re74                  89.48   88.05   81.02 -42.26
re75                  54.16   53.54   41.26 -67.27
educ                 -12.50    0.00  -35.88   0.00
black                 41.76  100.00   41.56   0.00
hispan              -161.35    0.00 -157.70   0.00
age                   89.26 -100.00   16.81  20.00

Sample sizes:

           Control Treated
All            429     185
Matched        185     185
Unmatched      244       0
Discarded        0       0
```

The `summary()` command provides simple statistics of the propensity score and the covariates used in the matching for the full and matched samples, including means, standardized bias, and Quantile-Quantile (Q-Q) plot statistics. These are used to assess whether there was a reduction in bias in the covariates. The `summary` command will additionally report (a) the original call of the MatchIt object, (b) how many units were matched, unmatched, or discarded due to the `discard` option (described below), and (c) the percent improvement in balance for each of the balance measures, defined as $100((|a| - |b|)/|a|)$, where a is the balance before and b is the balance after matching.

For each set of units (full sample and matched sample), the following statistics are provided: the "Means Treated" and "Means Control" columns show the weighted means in the treated and control groups; the "Treated SD" is the standard deviation of the covariate in the set of treated units; the "Std. Bias" is the difference in means in the treated and control groups, divided by the "Treated SD" calculated using all treated units. The same standard deviation is used for calculating the standardized bias in the full data and the matched data set so that the success of the matching at reducing bias in the covariate means can be easily assessed; standardizing by the same quantity puts the two differences in means on the same scale. Good matches will generally have standardized biases less than approximately 0.25; values greater than that imply that the groups have means that are more than a quarter of a standard deviation apart.

The final three columns of the summary output give summary statistics of a Q-Q plot (see below for more information on these plots). Those columns give the median, mean, and maximum orthogonal deviations from the 45-degree line of the Q-Q plot. If the empirical distributions of the two groups (treated and control) were exactly the same, all points would lie on the 45-degree line and thus all of these distances would be 0. Values greater than 0 indicate deviations between the groups in some part of the empirical distributions. The plots themselves, described below, can provide further insight into which part of the covariate distribution has differences between the groups.

In this example of nearest neighbor matching, we see that the matching has reduced the bias in the propensity score from 1.79 to 0.95 and has reduced the bias in some of the covariates (e.g., 1974 and 1975 income). Job training participants on average earned roughly \$3,524 less in 1974 and \$934 less in 1975 than non-participants, in the full sample. In the matched sample, the earnings difference is reduced to \$371 in 1974 and \$428 in 1975. This one-to-one matching algorithm has thus chosen 185 control individuals who are more similar to the treated group in 1974 income and 1975 income, which

is summarized by the 89.5 and 54.2 percent balance improvement in these covariates. However there are still fairly large differences on some of the covariates, such as the race variables. In this situation, if close matches on the race variables are desired, it may make sense to try Mahalanobis or exact matching on the race variables in addition to the propensity score matching. The large remaining bias in the propensity score (0.95 standard deviations) also indicates that it may be desirable to do subclassification instead of or in addition to the matching, as described below.

Two options to the `summary` command can also help with assessing balance and respecifying the propensity score model, as necessary. First, the `interactions=T` option with `summary` shows the balance of all squares and interactions of the covariates used in the matching procedure. Large differences in higher order interactions usually are a good indication that the assignment model needs to be respecified. Similarly, the `addlvariables` option with `summary` will provide balance measures on additional variables not included in the original matching procedure. If a variable (or interaction of variables) not included in the original distance measure has large imbalances in the matched groups, including that variable in the next model specification may improve the resulting balance on that variable. Dehejia and Wahba (1999) provide a detailed example of a propensity score specification algorithm. Because the outcome variable is not used in the matching procedure, a variety of matching methods can be tried, and the one that leads to the best resulting balance chosen.

We can also examine the balance graphically using the `plot()` command, which provides two types of plots: jitter plots of the distance measure, and Q-Q plots of each covariate. Figures 1 and 2 display these two sample plots. In the jitter plot, you may identify units by observation name by clicking the first mouse button near the units.

```
> plot(m.out)
> plot(m.out, type = "jitter")
```

The balance of the individual covariates is shown in the Q-Q plots. If the empirical distributions are the same in the treated and control groups, the points would all lie on the 45 degree line. Deviations from the 45 degree line indicate differences in the empirical distribution. The jitter plot shows the overall distribution of propensity scores in the treated and control groups. In the jitter plot, the size of each diamond is proportional to the weight given to that unit; matched units are in black while unmatched units are in grey.

Examining these graphs, we see that the matched samples are fairly well matched on the propensity score and the income variables, but that there are some remaining imbalances in the race and age variables, as was also seen using the `summary` command.

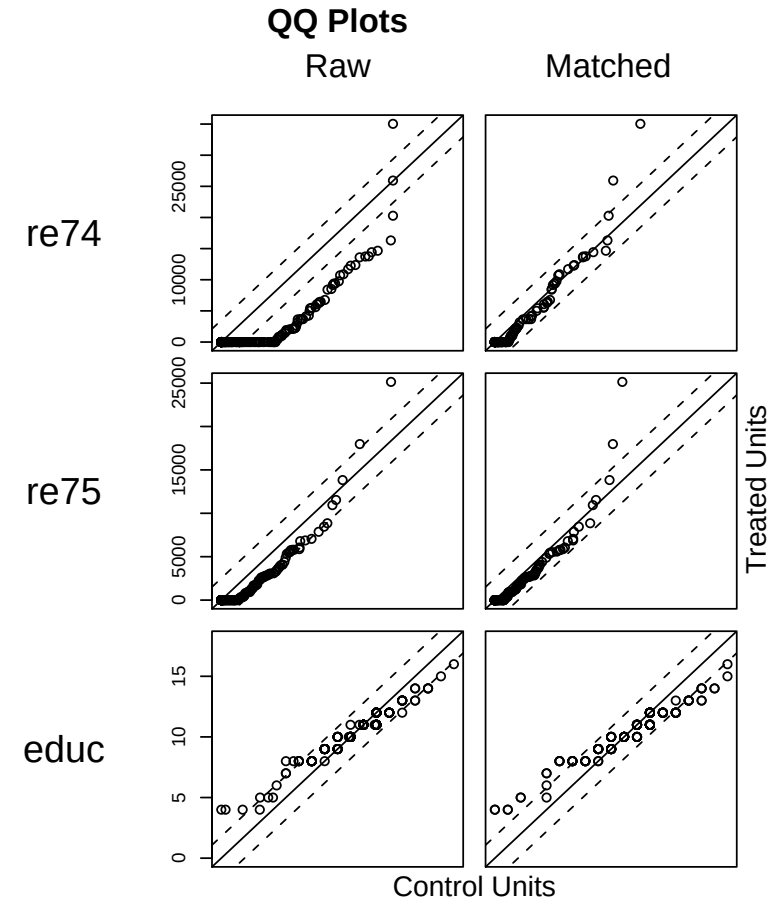
3. Now for subclassification, using the example from Section 3.2.2, we check the balance of covariates using the `summary()` and `plot()` commands.

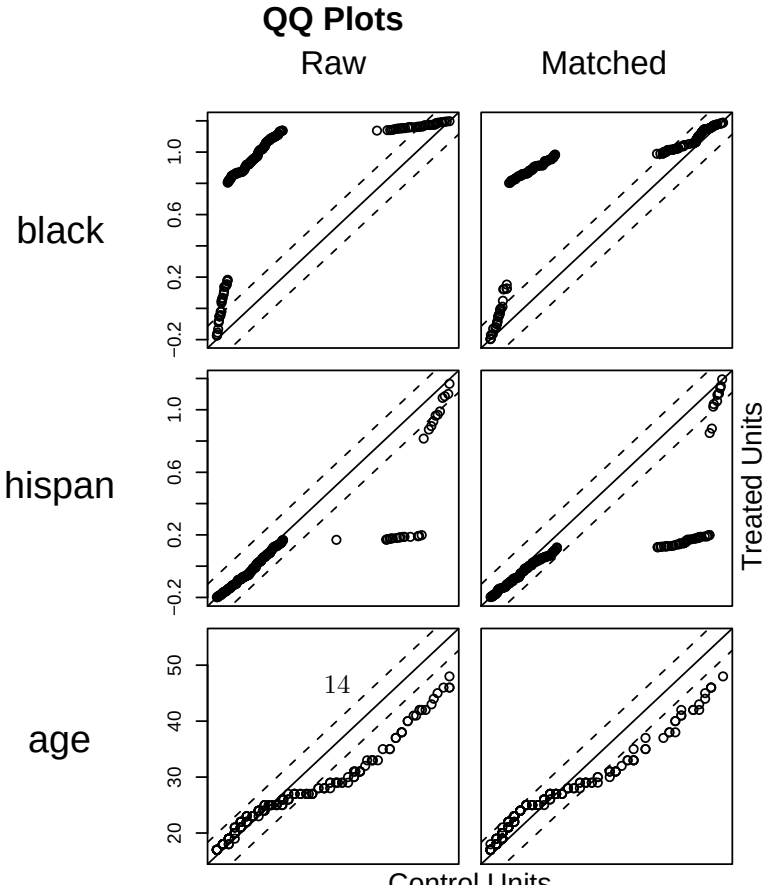
```
> m.out <- matchit(treat ~ re74 + re75 + educ + black + hispan +
+     age, data = lalonde, method = "subclass")
> summary(m.out)

Call:
matchit(formula = treat ~ re74 + re75 + educ + black + hispan +     age, data = lalonde, method = "

Summary of balance for all data:

         Means Treated Means Control Treated SD Std. Bias   QQ Med  QQ Mean
distance         0.566         0.187      0.211     1.792    0.732    0.535
re74          2095.574      5619.237   4886.620    -0.721 3430.277 5155.851
re75          1532.055      2466.484   3219.251    -0.290 1373.280 1511.706
educ            10.346        10.235      2.011     0.055    1.414    0.962
```


QQ Plots
Raw
Matched
re74
re75
educ
Treated Units
Control Units


QQ Plots
Raw
Matched
black
hispan
age
Treated Units
Control Units

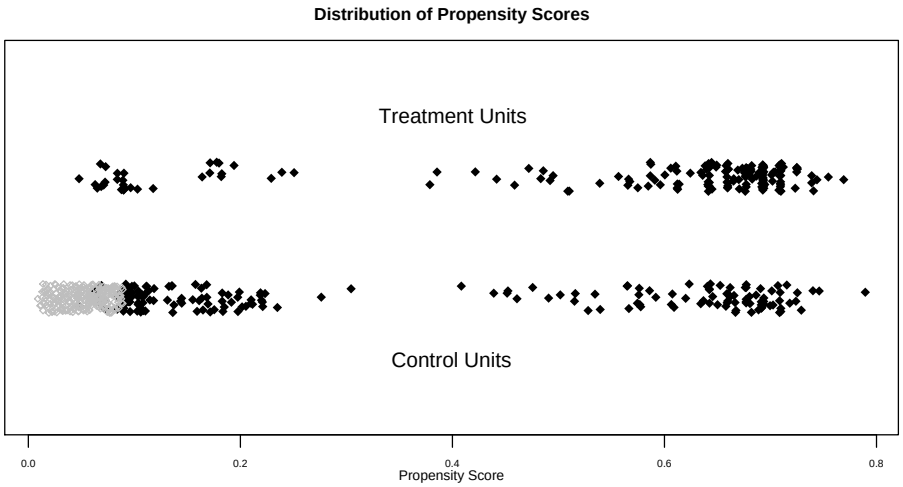


Figure 2: Sample diagnostic jitter plot: Nearest Neighbor matching with the MatchIt call `matchit(treat re74 + re75 + educ + black + hispan + age, data=lalonde, method="nearest")`. Matched units shown in black, unmatched units shown in grey.

```
black           0.843        0.203      0.365     1.757    1.414    0.903
hispan          0.059        0.142      0.237    -0.349    0.000    0.119
age            25.816       28.030      7.155    -0.309    1.414    4.650
           QQ Max
distance     0.840
re74     13044.159
re75      9609.595
educ         5.657
black        1.414
hispan       1.414
age         14.142


Summary of balance by subclasses:

, , Subclass 1

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.122         0.076      0.059     0.217     0.055     0.065
re74          3628.026      6332.407   8002.323    -0.553  4588.884  5184.224
re75          2232.359      2624.566   3340.297    -0.122   663.357  1146.136
educ            10.613        10.265      1.801     0.173     1.414     1.369
black            0.065         0.006      0.250     0.161     0.000     0.046
hispan           0.355         0.177      0.486     0.749     0.000     0.228
age             24.968        28.564      6.151    -0.503     2.828     6.562
           QQ Max
distance     0.123
re74     12979.299
re75      9631.728
```

```
educ         9.899
black        1.414
hispan       1.414
age         19.799

, , Subclass 2

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.539         0.534      0.070     0.021     0.016     0.020
re74          7529.849      7209.650   5753.799     0.066   657.495  1234.428
re75          4108.624      3484.381   5692.184     0.194  1710.911  2118.164
educ             9.000         8.692      2.989     0.153     0.707     1.295
black            1.000         1.000      0.000     0.000     0.000     0.000
hispan           0.000         0.000      0.000     0.000     0.000     0.000
age             28.097        33.269      8.138    -0.723     9.334     8.050
            QQ Max
distance     0.065
re74      4508.668
re75     16073.541
educ         5.657
black        0.000
hispan       0.000
age         19.799

, , Subclass 3

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.646         0.645      0.011     0.004     0.001     0.003
re74           892.324      1403.987   1624.758    -0.105   543.081   825.572
re75           707.407      1463.220   1504.611    -0.235   291.168   944.259
educ             8.933         9.118      1.172    -0.092     0.000     0.437
black            1.000         1.000      0.000     0.000     0.000     0.000
hispan           0.000         0.000      0.000     0.000     0.000     0.000
age             23.867        19.647      7.718     0.590     3.801     6.260
            QQ Max
distance     0.013
re74      2117.652
re75      8965.316
educ         2.828
black        0.000
hispan       0.000
age         19.976

, , Subclass 4

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.677         0.678      0.007    -0.003     0.004     0.004
re74           390.290       710.654    737.073    -0.066     0.000   441.668
re75          1193.816       683.647   1667.376     0.158   151.719   769.614
educ            10.000        10.571      0.775    -0.284     1.414     0.808
black            1.000         1.000      0.000     0.000     0.000     0.000
```

```
hispan           0.000         0.000      0.000     0.000     0.000     0.000
age             23.548        22.190      6.971     0.190     2.828     4.478
            QQ Max
distance     0.012
re74      1684.659
re75      3231.361
educ         2.828
black        0.000
hispan       0.000
age         11.314

, , Subclass 5

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.694         0.697      0.003    -0.013     0.001     0.003
re74            94.139       325.798    384.061    -0.047     0.000   157.765
re75           404.855       307.680   1339.194     0.030     0.000  1270.772
educ            11.000        11.429      0.258    -0.213     0.000     0.606
black            1.000         1.000      0.000     0.000     0.000     0.000
hispan           0.000         0.000      0.000     0.000     0.000     0.000
age             26.710        22.286      5.399     0.618     9.899     6.869
            QQ Max
distance     0.011
re74       800.767
re75      7783.902
educ         1.414
black        0.000
hispan       0.000
age         12.728

, , Subclass 6

         Means Treated Means Control Treated SD Std. Bias    QQ Med   QQ Mean
distance         0.720         0.724      0.016    -0.022     0.003     0.005
re74             0.000       270.099      0.000    -0.055     0.000   381.978
re75           518.669      1663.721   1644.262    -0.356     0.000  1329.640
educ            12.484        12.643      1.180    -0.079     0.000     0.233
black            1.000         1.000      0.000     0.000     0.000     0.000
hispan           0.000         0.000      0.000     0.000     0.000     0.000
age             27.645        27.000      7.423     0.090     2.448     3.318
            QQ Max
distance     0.029
re74      3627.005
re75      6825.954
educ         1.414
black        0.000
hispan       0.000
age         11.314


Sample sizes by subclasses:
```

```
        Subclass 1 Subclass 2 Subclass 3 Subclass 4 Subclass 5 Subclass 6
Treated         31         31         30         31         31         31
Control        344         26         17         21          7         14
Total          375         57         47         52         38         45

Summary of balance across subclasses

          Means Treated Means Control Treated SD Std. Bias  QQ Med  QQ Mean
distance          0.566         0.559      0.016     0.034   0.013    0.017
re74           2095.574      2715.819   1678.241    -0.127 967.190 1373.887
re75           1532.055      1705.840   1219.402    -0.054 470.490 1264.821
educ             10.346        10.460      0.660    -0.057   0.592    0.793
black             0.843         0.833      0.042     0.027   0.000    0.008
hispan            0.059         0.030      0.082     0.125   0.000    0.038
age              25.816        25.524      2.867     0.041   5.197    5.921
           QQ Max
distance    0.042
re74     4298.064
re75     8750.814
educ        4.013
black       0.237
hispan      0.237
age        15.799

Percent Balance Improvement:

          Mean and Std. Bias  QQ Med QQ Mean  QQ Max
distance              98.088   98.16   96.84  94.967
re74                  82.398   71.80   73.35  67.050
re75                  81.402   65.74   16.33   8.937
educ                  -3.301   58.11   17.56  29.054
black                 98.464  100.00   99.15  83.243
hispan                64.046    0.00   67.79  83.243
age                   86.817 -267.50  -27.34 -11.716
```

The `summary` output for subclassification is the same as that for nearest neighbor matching, except that the balance statistics are shown separately for each subclass, and the overall balance in the matched samples is calculated by aggregating across the subclasses. Consistent with the calculations for the full and matched samples, the standardized bias within each subclass is also calculated using the standard deviation of the full treated group, again so that the differences in means are all scaled by the same amount and not affected by changes in sample sizes.

In this example, the bias between the groups within each sublcass is generally small, with the exception of Subclass 1, which has a large number of control units with small propensity scores. The percent balance improvement also indicates that the subclassification has been successful at forming subclasses within which the distribution of covariates is more similar than in the full data set. In particular, 80 to 90 percent reduction in the standardized bias was achieved in 4 of the 6 covariates through subclassification; little bias reduction was achieved on the education variable, which had very small bias to start (standardized bias of only 0.055 in the full data set). For this example, it appears as though subclassification has been able to form better matched samples than the simple 1-1 nearest neighbor matching method described above.

We can also examine the balance graphically. Figures 3 and 4 display the two diagnostic plots. With subclassification, separate Q-Q plots are printed for each subclass; for each, the graphs shown in the left-hand column are those for the full data set. The jitter plot is the same as that for nearest neighbor matching, with the addition of vertical lines indicating the subclass cut-points.

```
> m.out <- matchit(treat ~ re74 + re75 + educ + black + hispan +
+     age, data = lalonde, method = "subclass")

> plot(m.out)
> plot(m.out, type = "jitter")
```

In Figure 3 we see that the empirical distributions of the treated and control units are much more similar in Subclass 3 (shown in the "Matched" column of figures) than they are in the full data set (shown in the "Raw" column of figures). We see that in this case, the distributions of most of the covariates are very similar in Subclass 3, with the exception of `age`, which on which the treated units generally have larger values than the control units (indicated by the points falling above the 45 degree line). The jitter plot indicates that using the `discard` option may be desirable here, to reduce any bias due to the many control units with low propensity scores.

3.4 Conducting Analyses after Matching

Once matching is complete and balance has been achieved, the MatchIt object (output from the `matchit()` function) can be used with any other analysis procedure; MatchIt is designed to make those analysis procedures less dependent on the modeling assumptions and thus work better. Therefore, any analysis you might have conducted using the original data set can be conducted with less model dependence using the matched data set. To obtain the matched data set, use `match.data(m.out)`, where `m.out` is a MatchIt object. See Section 4.2.4 for more options and details.

In this section, we describe our recommended approach (Ho et al., 2005), which uses Zelig to conduct parametric causal inference after preprocessing the data through MatchIt. The resulting matched data sets can also be exported to other statistical programs using commands such as `write.table()` and `write.csv()`. In addition, the commands in the `foreign` package allow users to save matched data sets as binary files for various statistical software including STATA and SPSS.

Zelig (Imai, King and Lau, 2004) is an easy-to-use R package that estimates a large variety of statistical models, gives easily interpretable results by simulating quantities of interest, provides numerical and graphical summaries, and is easily extensible. The package along with the complete documentation is available at http://gking.harvard.edu/zelig/. MatchIt and Zelig can be easily used together to enable estimation of causal effects in very general settings with a variety of statistical models.

First, we use `match.data()` to create the matched data by excluding unmatched units from the original data, and including information about the particular matching procedure (i.e., weights, subclasses, and the distance measure).

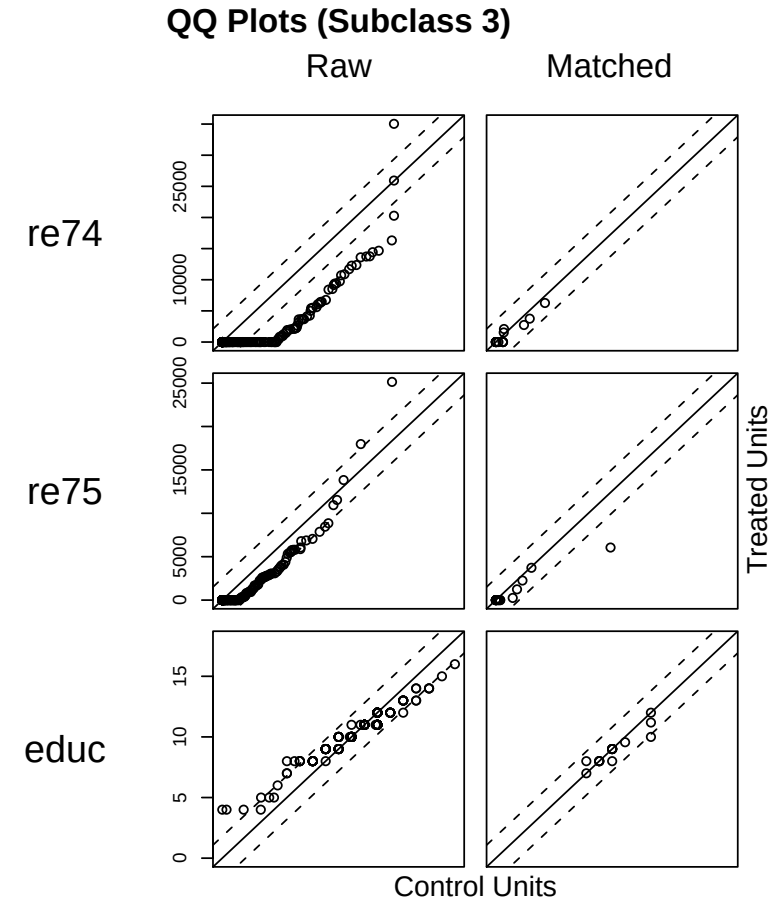
```
> mdata <- match.data(m.out)
```

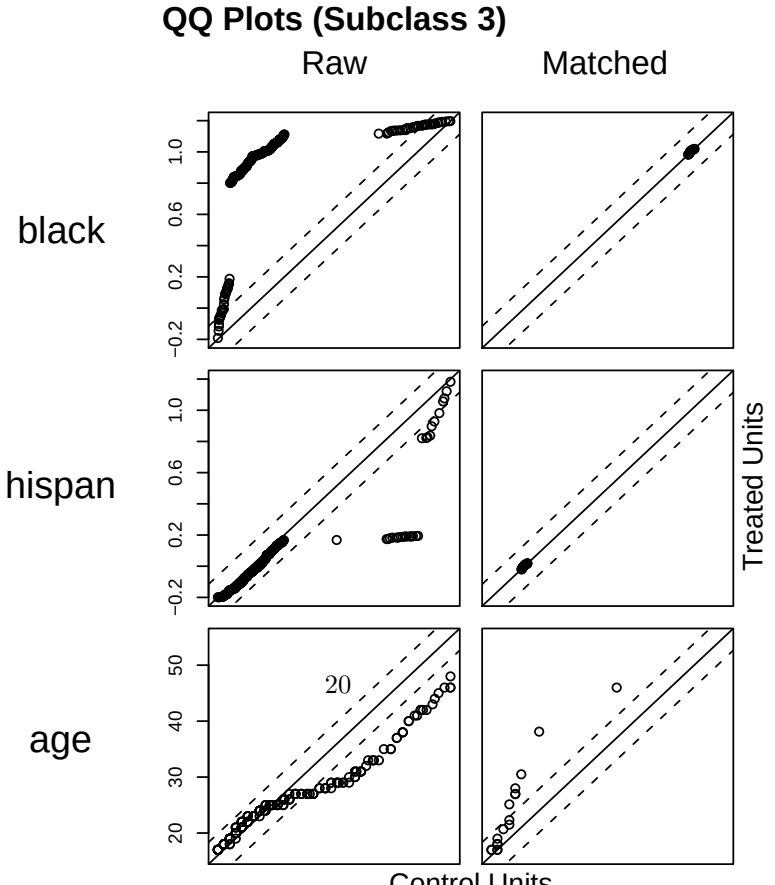
where `m.out` is the MatchIt object from `matchit()` and `mdata` is the resulting matched data. This matched data set can then be used for subsequent parametric analyses thorough Zelig or any other statistical programs. If one uses Zelig, then a syntax for least squares regression is

```
> z.out <- zelig(Y ~ treat + x1 + x2, model = "ls", data = mdata)
```

where `Y` is the outcome variable, and `z.out` is the output object from `zelig`. For more details, see Section 3.4.

To illustrate this approach, we provide several examples using the Lalonde data. Users can run these example commands by typing `demo(Zelig)` at the R prompt. We use the linear least squares model in these examples. However, a wide range of other models are available in Zelig (for the list of supported models, see http://gking.harvard.edu/zelig/docs/Models_Zelig_Can.html), and they can be used


QQ Plots (Subclass 3)
Raw
Matched
re74
re75
educ
0
5000
10000
15000
25000
5
10
15
Treated Units
Control Units


QQ Plots (Subclass 3)
Raw
Matched
black
hispan
age
−0.2
0.2
0.6
1.0
20
30
40
50
Treated Units
Control Units

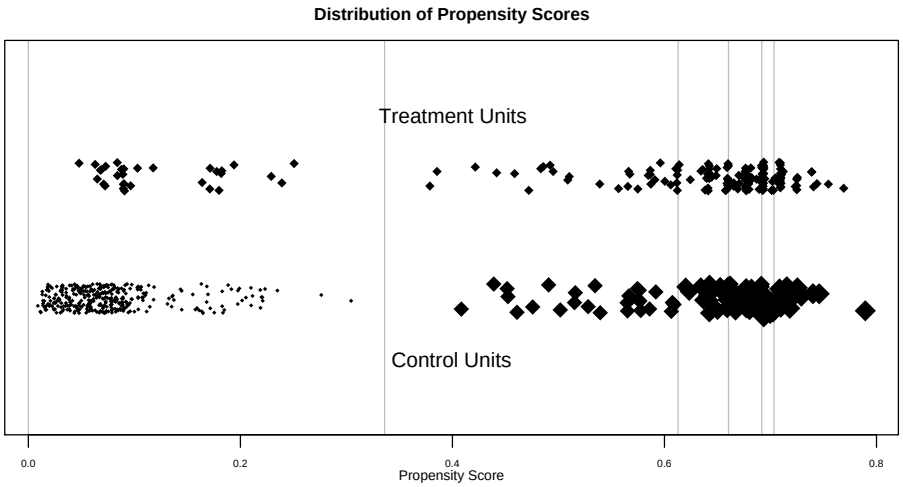


Figure 4: Sample diagnostic jitter plot: Subclassification with MatchIt call `matchit(treat re74 + re75 + educ + black + hispan + age, data = lalonde, method = "subclass")` followed by `plot(m.out, type="jitter")`. Vertical lines indicate the subclass cut points and the area of each diamond is proportional to the weights of the matched units for each subclass. Unmatched units would be shown in grey.

in the exactly same way. If you have not installed Zelig, follow the installation procedure described at http://gking.harvard.edu/zelig/docs/Installation.html

1. Nearest neighbor matching using propensity scores:

```
> data(lalonde)
> library(Zelig)
> m.out1 <- matchit(treat ~ age + educ + black + hispan + nodegree +
+     married + re74 + re75, method = "nearest", data = lalonde)
> z.out1 <- zelig(re78 ~ age + educ + black + hispan + nodegree +
+     married + re74 + re75 + distance, data = match.data(m.out1,
+     "control"), model = "ls")
> x.out1 <- setx(z.out1, data = match.data(m.out1, "treat"), fn = NULL,
+     cond = TRUE)
> s.out1 <- sim(z.out1, x = x.out1)
> summary(s.out1)


  Model: ls
  Number of simulations: 1000

Mean Values of Observed Data (n = 185)
(Intercept)         age        educ       black      hispan    nodegree
  1.000e+00   2.582e+01   1.035e+01   8.432e-01   5.946e-02   7.081e-01
    married        re74        re75    distance
  1.892e-01   2.096e+03   1.532e+03   5.774e-01

Pooled Expected Values: E(Y|X)
```

```
   mean      sd    2.5%   97.5%
 5005.4  2292.0   732.9 10074.8

Pooled Average Treatment Effect: Y - EV
  mean     sd   2.5%  97.5%
1343.7  584.2  186.7 2504.9
```

The estimated average treatment effect on the treated is thus \$1343.73, with a 95% interval of (\$186.72, \$2504.88).

2. Estimating the average treatment effects on both the treated and the control groups. We use the same `matchit()` output as in the first example above.

```
> z.out2 <- zelig(re78 ~ age + educ + black + hispan + nodegree +
+     married + re74 + re75 + distance, data = match.data(m.out1,
+     "control"), model = "ls")
> x.out2 <- setx(z.out2, data = match.data(m.out1, "control"),
+     fn = NULL, cond = TRUE)
> s.out2 <- sim(z.out2, x = x.out2)
> ate.all <- c(s.out1$qi$ate.ev, -s.out2$qi$ate.ev)
> mean(ate.all)

[1] 666.2


> sd(ate.all)

[1] 850.6


> quantile(ate.all, c(0.025, 0.975))

  2.5%  97.5%
-705.3 2303.7
```

3. Subclassification: In this case, the average treatment effect estimates are obtained for each subclass separately as well as for the overall sample. Estimating the treatment effects separately for each subclass, and then aggregating across subclasses, can significantly increase the robustness of the ultimate results since the paramatric analysis within each subclass requires only local rather than global assumptions.

```
> m.out2 <- matchit(treat ~ age + educ + black + hispan + nodegree +
+     married + re74 + re75, data = lalonde, method = "subclass",
+     subclass = 4)
> z.out3 <- zelig(re78 ~ re74 + re75 + distance, data = match.data(m.out2,
+     "control"), model = "ls", by = "subclass")
> x.out3 <- setx(z.out3, data = match.data(m.out2, "treat"), fn = NULL,
+     cond = TRUE)
> s.out3 <- sim(z.out3, x = x.out3, num = 100)
> summary(s.out3)
```

```
  Model: ls
  Number of simulations: 25

Mean Values of Observed Data (n = 46)
(Intercept)        re74        re75    distance
     1.0000   5430.5389   2929.0394      0.2392

Pooled Expected Values: E(Y|X)
 mean    sd  2.5% 97.5%
 4244  5775 -7629 16451

Pooled Average Treatment Effect: Y - EV
  mean     sd   2.5%  97.5%
2100.7 1873.4 -850.5 6640.1


> summary(s.out3, subset = 1)


Results for 1

  Model: ls
  Number of simulations: 25

Mean Values of Observed Data (n = 46)
(Intercept)        re74        re75    distance
     1.0000   5430.5389   2929.0394      0.2392

Pooled Expected Values: E(Y|X)
 mean    sd  2.5% 97.5%
 6865  3843  2975 17015

Pooled Average Treatment Effect: Y - EV
  mean     sd   2.5%  97.5%
 327.1  463.0 -441.6 1215.3



> summary(s.out3, subset = 2)


Results for 1

  Model: ls
  Number of simulations: 25

Mean Values of Observed Data (n = 45)
(Intercept)        re74        re75    distance
     1.0000   1777.4221    972.3441      0.6039

Pooled Expected Values: E(Y|X)
```

```
  mean     sd   2.5%  97.5%
3876.0 2250.0 -603.5 8807.3

Pooled Average Treatment Effect: Y - EV
  mean     sd   2.5%  97.5%
2610.3 1067.6  652.9 4642.0



> summary(s.out3, subset = 3)


Results for 1

  Model: ls
  Number of simulations: 25

Mean Values of Observed Data (n = 47)
(Intercept)        re74        re75    distance
     1.0000    939.9688   1217.4546      0.6934

Pooled Expected Values: E(Y|X)
 mean    sd  2.5% 97.5%
 3708  4410  -569  9341

Pooled Average Treatment Effect: Y - EV
 mean    sd  2.5% 97.5%
 2186   923   526  3822
```

4 Reference Manual

4.1 Inputs

The main command, `matchit()`, can be used to implements any of the matching procedures:

```
> m.out <- matchit(formula, data, method = "nearest", distance = "logit", distance.options=list(),
discard = "none", reestimate = FALSE, ...)
```

The command takes some inputs that are common to all matching procedures and other inputs specific to particular procedures. The outputs are standard across all procedures. We organize the reference manual in these categories.

4.1.1 All Matching Methods

1. `formula` takes the usual syntax of R formula, `treat ~ x1 + x2`, where `treat` is a binary treatment indicator and `x1` and `x2` are the pre-treatment covariates. Both the treatment indicator and pre-treatment covariates must be contained in the same data frame, which is specified as `data` (see below). All of the usual R syntax for formulas work here. For example, `x1:x2` represents the first order interaction term between `x1` and `x2`, and `I(x1 ^ 2)` represents the square term of `x1`. See `help(formula)` for details.

2. `data` specifies the data frame containing the variables called in `formula`. You may find it helpful for the diagnostics to specify observation names in the data frame (see Section 6.3).

3. `method` specifies a matching method. Currently, `exact` (exact matching), `full` (full matching), `nearest` (nearest neighbor matching), `optimal` (optimal matching), and `subclass` (subclassification) are available. The default is `nearest`. Note that within each of these matching methods, MatchIt offers a variety of options. See Section 3 for more details.

4. `distance` specifies the method used to estimate the distance measure. The default is logistic regression, `logit`. Before using any of these techniques, it is best to understand the theoretical groundings of these techniques and to evaluate the results. Most of these methods (such as logistic or probit regression) are estimating the propensity score, defined as the probability of receiving treatment, conditional on the covariates (Rosenbaum and Rubin (1983)). The distance measures used are the predicted probabilities from the model (the propensity scores). Currently, the following methods are available:

   (a) `mahalanobis` computes the Mahalanobis distance measure (`mahalanobis()` in the `stats` package).

   (b) binomial generalized linear models with various links (`glm()` in the `stats` package); `logit` (logistic link), `linear.logit` (logistic link with linear propensity score)[2], `probit` (probit link), `linear.probit` (probit link with linear propensity score), `cloglog` (complementary log-log link), `linear.cloglog` (complementary log-log link with linear propensity score), `log` (log link), `linear.log` (log link with linear propensity score), `cauchit` (Cauchy CDF link), `linear.cauchit` (Cauchy CDF link with linear propensity score).

   (c) binomial generalized additive model with various links (`gam()` in the `mgcv` package); `GAMlogit` (logistic link), `GAMlinear.logit` (logistic link with linear propensity score), `probit` (probit link), `GAMlinear.probit` (probit link with linear propensity score), `GAMcloglog` (complementary log-log link), `GAMlinear.cloglog` (complementary log-log link with linear propensity score), `GAMlog` (log link), `GAMlinear.log` (log link with linear propensity score), `GAMcauchit` (Cauchy CDF link), `GAMlinear.cauchit` (Cauchy CDF link with linear propensity score). Hastie and Tibshirani (1990); Beck and Jackman (1998) and many others discuss the generalized additive models.

   (d) `nnet`, neural network model (`nnet()` in the `nnet` package). Beck, King and Zeng (2000); Bishop (1995); King and Zeng (2002); White (1992); Zeng (1999) among many others discuss neural networks.

   (e) `rpart`, classification trees (`rpart()` in the `rpart` package). Breiman et al. (1984); Ruger et al. (2003) and many others discuss classification trees.

5. `distance.options` specifies the optional arguments that are passed to the model for estimating the distance measure. The input to this argument should be a list. For example, if the distance measure is estimated with a logistic regression, users can increase the maximum IWLS iterations by `distance.options = list(maxit = 5000)`.

6. `discard` specifies whether to discard units that fall outside some measure of support of the distance score before matching, and not allow them to be used at all in the matching procedure. Note that discarding units may change the quantity of interest being estimated.

   - `none` (default) discards no units before matching. Use this option when the units to be matched are substantially similar, such as in the case of matching treatment and control units from a field experiment that was close to (but not fully) randomized (e.g., Imai 2005), when caliper matching will restrict the donor pool, or when you do not wish to change the quantity of interest and the parametric methods to be used post-matching can be trusted to extrapolate.

[2]The linear propensity scores are obtained by transforming back onto a linear scale

- `both` discards all units (treated and control) that are outside the support of the distance measure. Use this option when the units to be matched are substantially different (when there is a large degree of non-overlapping support on the distance score), such as in the case of measuring the effect of democracy on economic growth.
- `control` discards only control units outside the support of the distance measure of the treated units. Use this option when the average treatment effect on the treated is of most interest and when unwilling to discard non-overlapping treatment units (which would change the quantity of interest), such as possibly in the case of the effect of job training on those individuals that actually participated in a job evaluation program or a drug study where interest is in all patients treated with the drug.
- `treat` discards only treated units outside the support of the distance measure of the control units. Use this option when the average treatment effect on the control units is of most interest and when unwilling to discard control units.
- `convex.hull` discards control units not within the convex hull of the treated units using the method developed in (King and Zeng, 2004).

7. `reestimate` specifies whether the model for distance measure should be re-estimated after units are discarded. The input must be a logical value. The default is `FALSE`. Re-estimation may be desirable for efficiency reasons, especially if many units were discarded and so the post-discard samples are quite different from the original samples.

8. `verbose` specifies whether or not to print out comments indicating the status of the matching. The input must be a logical value. The default is `FALSE`.

4.1.2 Exact Matching

Exact matching is implemented in MatchIt using `method = "exact"`. Exact matching will be done on all covariates included on the right-hand side of the `formula` specified in the MatchIt call. No `distance` option is used for exact matching, and there are no additional options for exact matching.

4.1.3 Subclassification

1. `subclass` is either (1) a scalar, specifying the number of subclasses, or (2) a vector of probabilities bounded between 0 and 1, to create quantiles of the distance measure using the units in the group specified by `sub.by`. The default is `subclass = 6`.

2. `sub.by` specifies by what criteria to subclassify: `"treat"` indicates by the number of treatment units (default), `"control"` indicates by the number of control units, and `"all"` indicates by the total number of units.

4.1.4 Nearest Neighbor Matching

1. `m.order` specifies the order in which to match treatment units with control units:

   - `"largest"` indicates matching from the largest value of the distance measure to the smallest. This is the default.
   - `"smallest"` indicates matching from the smallest value of the distance measure to the largest.
   - `"random"` indicates matching in random order.

2. `replace` specifies whether each control unit can be matched to more than one treated unit. For matching "with replacement", `replace = TRUE`. If each control is to be used as a match at most once ("without replacement"), `replace = FALSE`. The default is `FALSE`.

3. `ratio` specifies the number of control units to match to each treated unit, default=1. If matching is done without replacement and there are fewer control units than ratio times the number of eligible treated units (i.e., there are not enough control units for the specified method), then the higher ratios will have `NA` in place of the matching unit number in `match.matrix`.

4. `exact` specifies variables on which to perform exact matching within the nearest neighbor matching. If `exact` is specified, only matches that exactly match on the covariates in `exact` will be allowed. Within the matches that match on the variables in `exact`, the match with the closest distance measure will be chosen. `exact` should be entered as a vector of variable names (`exact = c("X1", "X2")`) that are names of variables in `data`.

5. `caliper` specifies the number of standard deviations of the distance measure within which to draw control units, default=0. If a caliper is specified, the matches are restricted to being within the caliper and a control unit within the caliper for a treated unit is randomly selected as the match for that treated unit. If `caliper != 0`, there are two additional options:

    - `calclosest` specifies whether to take the nearest available match if no matches are available within the `caliper`. The default is `FALSE`.
    - `mahvars` specifies variables on which to perform Mahalanobis-metric matching within each caliper (default=NULL). Variables should be entered as a vector of variable names (`mahvars=c("X1","X2")`) that are names of variables in `data`. If `mahvars` is specified without `caliper`, the caliper is set to 0.25.

6. `subclass` and `sub.by`. See Section 3.2.2 for more details on these options. If a `subclass` is specified within `method = "nearest"`, the matched units will be placed into subclasses after the nearest neighbor matching is completed.

4.1.5 Full Matching

1. `...` represents additional inputs that can be passed to the `fullmatch()` function in the `optmatch` package. See `help(fullmatch)` for details.

4.1.6 Optimal Matching

The available options are listed below.

1. `ratio` specifies the number of control units to be matched to each treatment unit, the default is `1`.

2. `...` represents additional inputs that can be passed to the `fullmatch()` function in the `optmatch` package. See `help(fullmatch)` for details.

4.2 Outputs

4.2.1 Output Object Contents

Regardless of the type of matching performed, the `matchit` output object contains the following elements:

1. `call` provides the original `matchit()` call.

2. `formula` shows the formula used to specify the model for estimating the distance measure.

3. `model` stores the output of the model used to estimate the distance measure. `summary(m.out$model)` will give the summary of the model where `m.out` is the output object from `matchit()`.

4. `match.matrix` is an n_1 by `ratio` matrix where:

- the row names, which can be obtained through `row.names(match.matrix)`, represent the names of the treatment units, which come from the data frame specified in `data` (to learn how to do this, see Section 6.3).
- each column stores the name(s) of the control unit(s) matched to the treatment unit of that row. For example, when the `ratio` input for nearest neighbor or optimal matching is specified as 3, the three columns of `match.matrix` represent the three control units matched to one treatment unit).
- `NA` indicates that the treatment unit was not matched.

5. `discarded` is a vector of length n that displays whether the units were ineligible for matching due to common support restrictions. It equals `TRUE` if unit i was discarded, and it is set to `FALSE` otherwise.
6. `distance` is a vector of length n with the estimated distance measure for each unit.
7. `weights` is a vector of length n that provides the weights assigned to each unit in the matching process. Unmatched units have weights equal to 0. Matched treated units have weight 1. Each matched control unit has weight proportional to the number of treatment units to which it was matched, and the sum of the control weights is equal to the number of uniquely matched control units. See Section 6.2 for more details.
8. `subclass` contains the subclass index in an ordinal scale from 1 to the total number of subclasses as specified in `subclass` (or the total number of subclasses from full or exact matching). Unmatched units have `NA`.
9. `q.cut` gives the subclass cut-points that classify the distance measure.
10. `treat` stores the treatment indicator from `data` (the left-hand side of `formula`).
11. `X` stores the covariates used for estimating the distance measure (the right-hand side of `formula`). When applicable, `X` is augmented by covariates contained in `mahvars` and `exact`.

4.2.2 `summary()`

The `summary` command returns more information about the MatchIt model. Optional inputs are:

1. `interactions`, which is an option to calculate summary statistics in `sum.all` and `sum.matched` for all covariates, their squares, and two-way interactions when `interactions=TRUE` and only the covariates themselves when `interactions=FALSE` (default).
2. `addlvariables`, which may contain additional variables on which to calculate the diagnostic statistics (in addition to the variables included in the matching procedure). By default, `addlvariables=NULL`. `addlvariables` must be specified as a data frame, with the same number of units and units in the same order as in the data set sent to MatchIt.

The `summary` call returns, when applicable:

1. The original assignment model call.
2. `sum.all` is a data frame that contains variable names and interactions down the row names, and summary statistics on *all observations* in each of the columns. The columns in `sum.all` contain [3]:
   - means of all covariates X for treated and control units, where `Means Treated`$= \mu_{X|T=1} = \frac{1}{n_1}\sum_{T=1} X_i$ and `Means Control`$= \mu_{X|T=0} = \frac{1}{n_0}\sum_{T=0} X_i$,

[3] The output for full matching is slightly different from that described here; see Section 3.2.4 for details.

- standard deviation in the treated group for all covariates X, $\quad s_{x|T=1} = \sqrt{\frac{\sum_{i\in\{i:T_i=1\}} X_i - \mu_{X|T=1}}{n_1 - 1}}$.
- summary statistics from a Q-Q plot, which compares treated and control covariate distributions, where `QQ Med`, `QQ Mean`, and `QQ Max` indicate the median, mean, and maximum orthogonal deviations from the 45 degree line of a Q-Q plot.
- standardized bias statistics,

$$\text{Std.Bias} = \frac{\mu_{X|T=1} - \mu_{X|T=0}}{s_{x|T=1}}.$$

3. `sum.matched` is a data frame which contains variable names down the row names, and summary statistics on only the *matched observations* in each of the columns. Specifically, the columns in `sum.matched` contain the following elements[4]:
   - weighted means for matched treatment units of all covariates X and their interactions, where `Means Treated`$= \mu_{wX|T=1} = \frac{1}{n_1}\sum_{T=1} w_i X_i$ and `Means Control`$= \mu_{wX|T=0} = \frac{1}{n_0}\sum_{T=0} w_i X_i$,
   - weighted standard deviations in the matched treated group for all covariates X, where `SD` $= s_{wX} = \sqrt{\frac{1}{n}\sum_i (w_i X_i - \overline{X}^*)^2}$, where $\overline{X}^*$ is the weighted mean of X in the matched treated group.
   - standardized bias statistics `Std. Bias`$= \frac{\mu_{wX|T=1} - \mu_{wX|T=0}}{s_{x|T=1}}$, and

   where w represents the vector of `weights`.
4. `reduction` shows the percent bias reduction achieved in each of the balance measures in `sum.all` and `sum.matched`, defined as $100(|a| - |b|)/|a|$, where a was the value of the balance measure before matching and b is the value of the balance measure after matching. Because the difference in means and the standardized bias differ only by a constant (the standard deviation in the full treated group), the percent reduction in bias is the same for these two measures, and thus is only printed out once.
5. `nn` gives the sample sizes in the full and matched samples and the number of discarded units, by treatment and control.
6. `q.table` is an array that contains the same information as `sum.matched` by subclass.
7. `qn` gives the sample sizes in the full and matched samples and the number of discarded units, by subclass and by treatment and control.
8. `match.matrix` from the matchit output.

4.2.3 `plot()`

The `plot` command allows you to check the distributions of covariates in the assignment model, squares, and interactions, and within each subclasses if specified. The graphs present:

1. Q-Q plots of each covariate to check balance of marginal distributions (`type="QQ"` (default)). This graph plots covariate values that fall in (approximately) the same quantile of treated and control distributions. Control unit quantile values are plotted on the x-axis, and treated unit quantile values are plotted on the y-axis. If values fall below the 45 degree line, control units generally take lower values of the covariate. Data points that fall exactly on the 45 degree line indicate that the marginal distributions are identical. Discrete covariates that take 5 or fewer values are jittered for visibility. This may be changed by setting the option `discrete.cutoff`.
2. Jitter plots of the propensity score for treated and control units (`type="jitter"`).

[4]The values output for full matching are slightly different from that described here; see Section 3.2.4 for details

4.2.4 `match.data()`

To extract the matched data set for subsequent analyses from the output object (see Section 3.4), we provide the function `match.data()`. This is used as follows:

```
m.data <- match.data(object, group = "all", distance = "distance", weights = "weights",
                     subclass = "subclass")
```

The output of the function `match.data()` is the original data frame where additional information about matching (i.e., distance measure as well as resulting weights and subclasses) is added, restricted to units that were matched.

Inputs `match.data()` takes the following inputs:

1. `object` is the output object from `matchit()`. This is a required input.
2. `group` specifies for which matched group the user wants to extract the data. Available options are `"all"` (all matched units), `"treat"` (matched units in the treatment group), and `"control"` (matched units in the control group). The default is `"all"`.
3. `distance` specifies the variable name used to store the distance measure. The default is `"distance"`.
4. `weights` specifies the variable name used to store the resulting weights from matching. The default is `"weights"`. See Section 6.2 for more details on the weights.
5. `subclass` specifies the variable name used to store the subclass indicator. The default is `"subclass"`.

Examples Here, we present examples for using `match.data()`. Users can run these commands by typing `demo(match.data)` at the R prompt.

```
## load the Lalonde data
data(lalonde)

## perform nearest neighbor matching
m.out1 <- matchit(treat ~ re74 + re75 + age + educ, data = lalonde, method = "nearest",
                  distance = "logit")

## obtain matched data
m.data1 <- match.data(m.out1)

## summarize the resulting matched data
summary(m.data1)

## obtain matched data for the treatment group
m.data2 <- match.data(m.out1, group = "treat")
summary(m.data2)

## obtain matched data for the control group
m.data3 <- match.data(m.out1, group = "control")
summary(m.data3)

## run a subclassification method
m.out2 <- matchit(treat ~ re74 + re75 + age + educ, data=lalonde, method = "subclass")

## specify different names
```

```
m.data4 <- match.data(m.out2, subclass = "block", weights = "w",
                      distance = "pscore")
## print the variable names of the matched data
names(m.data4)
```

5 What's New?

- **2.0-1** (??, 2005): Stable release for R 2.1. Major revisions.
- **1.0-2** (August 10, 2005): Stable release for R 2.1. Minor bug fixes (Thanks to Bart Bonikowski).
- **1.0-1** (January 3, 2005): Stable release for R 2.0. The first official version of MatchIt

6 Frequently Asked Questions

6.1 Can I use a Difference-in-Difference Estimator for Matched Data?

A difference-in-differences (DID) analysis can be easily conducted with MatchIt. If we were interested in the DID matching estimate in the Lalonde data, we could simply include `re75` as a covariate in the preprocessing step. Then the analysis can be performed on the change in income from 1975 to 1978: `re78-re75`. Time-varying covariates (of which none exist in the Lalonde data) should of course also be differenced for the DID estimator.

6.2 How Exactly are the Weights Created?

Each type of matching method can be thought of as creating groups of units with at least one treated unit and at least one control unit in each. In exact matching, subclassification, or full matching, these groups are the subclasses formed, and the number of treated and control units will vary quite a bit across subclasses. In nearest neighbor or optimal matching, the groups are the pairs (or sets) of treated and control units matched. In 1:1 nearest neighbor matching there will be one treated unit and one control unit in each group. In 2:1 nearest neighbor matching there will be one treated unit and two control units in each group. Unmatched units receive a weight of 0. All matched treated units receive a weight of 1.

The weights for matched control units are formed as follows:

1. Within each group, each control unit is given a preliminary weight of n_{ti}/n_{ci}, where n_{ti} and n_{ci} are the number of treated and control units in group i, respectively.
2. If matching is done with replacement, each control unit's weight is added up across the groups in which it was matched.
3. The control group weights are scaled to sum to the number of uniquely matched control units.

With subclassification, when the analysis is done separately within each subclass and then aggregated up across the subclasses, these weights will generally not be used, but they may be used for full matching or nearest neighbor matching if the number of control units matched to each treated unit varies.

6.3 How Do I Create Observation Names?

Since the diagnostics often make use of the observation names of the data frame, you may find it helpful to specify observation names for the data input. Use the `row.names` command to achieve this. For example, to assign the names "Dan", "Kosuke", "Liz" and "Gary" to a data frame with the first four observations in the Lalonde data, type:

```
> test <- lalonde[1:4, ]
> row.names(test) <- c("Dan", "Kosuke", "Liz", "Gary")
> print(test)

       treat age educ black hispan married nodegree re74 re75  re78
Dan        1  37   11     1      0       1        1    0    0  9930
Kosuke     1  22    9     0      1       0        1    0    0  3596
Liz        1  30   12     1      0       0        0    0    0 24909
Gary       1  27   11     1      0       0        1    0    0  7506
```

6.4 How Do I Ensure Replicability As MatchIt Versions Develop?

As the literature on matching techniques is rapidly evolving, MatchIt will strive to incorporate new developments. MatchIt is thereby an evolving program. Users may be concerned that analysis written in a particular version may not be compatible with newer versions of the program. The primary way to ensure that replication archives remain valid is to record the version of MatchIt that was used in the analysis. Our website maintains binaries of all public release versions, so that researchers can replicate results exactly with the appropriate version (for Unix-based platforms, see http://gking.harvard.edu/src/contrib/; for windows, see http://gking.harvard.edu/bin/windows/contrib/).

In addition, users may find it helpful to install packages with version control, using the `installWithVers` command with `install.packages`. So for example, in the windows R console, users may download the appropriate version from our website and install the package with version control by:

```
install.packages(choose.files('',filters=Filters[c('zip','All'),]),
                 .libPaths()[1],installWithVers=T,CRAN=NULL)
```

R CMD INSTALL similarly permits users to specify this version using the `-with-package-versions` option. After having specified version control, different versions of the program may be called as necessary. Similar advice may also be appropriate for version control for R more generally.

6.5 What Do I Do about Missing Data?

MatchIt requires complete data sets, with no missing values (other than potential outcomes of course). If there are missing values in the data set, imputation techniques should be used first to fill in ("impute") the missing values (both covariates and outcomes), or the analysis should be done using only complete cases (which we do not in general recommend). For imputation software, see Amelia at (http://gking.harvard.edu/stats.shtml) or other programs at http://www.multiple-imputation.com. For more information on missing data and imputation methods, see King et al. (2001).

6.6 Why Preprocessing?

The purpose of matching is to approximate an experimental template, where the matching procedure approximates random assignment of treatment in order to balance covariates between treatment and control groups. Separation of the estimation procedure into two steps simulates the research design of an experiment, where no information on outcomes is known at the point of experimental design and randomization. Much like an experimenter cannot easily rerun an experiment if the outcome was not satisfactory, the separation of the balancing process in MatchIt from the analysis process afterwards helps keep clear the goal of balancing control and treatment groups and makes it less likely that the user will inadvertently cook the books in his or her favor.

References

Beck, Nathaniel, Gary King and Langche Zeng. 2000. "Improving Quantitative Studies of International Conflict." *American Political Science Review*. 94(1):21–36. http://gking.harvard.edu/files/abs/improv-abs.shtml.

Beck, Nathaniel and Simon Jackman. 1998. "Beyond Linearity by Default: Generalized Additive Model." *American Journal of Political Science*. 42(2):596–627.

Bishop, Christopher M. 1995. *Neural Networks for Pattern Recognition*. Oxford: Oxford University Press.

Breiman, Leo, Jerome H. Friedman, Richard A. Olshen and Charles J. Stone. 1984. *Classification and Regression Trees*. New York, New York: Chapman & Hall.

Dehejia, Rajeev H. and Sadek Wahba. 1999. "Causal Effects in Nonexperimental Studies: Re-Evaluating the Evaluation of Training Programs." *Journal of the American Statistical Association*. 94:1053–62.

Gu, X.S. and Paul R. Rosenbaum. 1993. "Comparison of multivariate matching methods: structures, distances, and algorithms." *Journal of Computational and Graphical Statistics*. 2:405–420.

Hansen, Ben B. 2004. "Full Matching in an Observational Study of Coaching for the SAT." *Journal of the American Statistical Association*. 99(467):609–618.

Hastie, Trevor J. and Robert Tibshirani. 1990. *Generalized Additive Models*. London: Chapman Hall.

Ho, Daniel, Kosuke Imai, Gary King and Elizabeth Stuart. 2005. "Matching as Nonparametric Preprocessing for Parametric Causal Inference." http://gking.harvard.edu/files/matchp.pdf.

Imai, Kosuke. 2005. "Do Get-Out-The-Vote Calls Reduce Turnout? The Importance of Statistical Methods for Field Experiments." *American Political Science Review*. 99(2):283–300.

Imai, Kosuke and David A. van Dyk. 2004. "Causal Inference with General Treatment Treatment Regimes: Generalizing the Propensity Score." *Journal of the American Statistical Association*. 99(467):854–866.

Imai, Kosuke, Gary King and Olivia Lau. 2004. "Zelig: Everyone's Statistical Software.". http://gking.harvard.edu/zelig.

King, Gary, James Honaker, Anne Joseph and Kenneth Scheve. 2001. "Analyzing Incomplete Political Science Data: An Alternative Algorithm for Multiple Imputation." *American Political Science Review*. 95(1):49–69.

King, Gary and Langche Zeng. 2002. "Improving Forecasts of State Failure." *World Politics*. 53(4):623–658. http://gking.harvard.edu/files/abs/civil-abs.shtml.

King, Gary and Langche Zeng. 2004. "When Can History Be Our Guide? The Pitfalls of Counterfactual Inference." http://gking.harvard.edu/files/counterf.pdf.

Lalonde, Robert. 1986. "Evaluating the Econometric Evaluations of Training Programs." *aer*. 76:604–620.

Rosenbaum, Paul R. 2002. *Observational Studies, 2nd Edition*. New York, NY: Springer Verlag.

Rosenbaum, Paul R. and Donald B. Rubin. 1983. "The Central Role of the Propensity Score in Observational Studies for Causal Effects." *Biometrika*. 70:41–55.

Ruger, Theodore W., Pauline T. Kim, Andrew D. Martin and Kevin M. Quinn. 2003. "The Supreme Court Forecasting Project: Legal and Political Science Approaches to Predicting Supreme Court Decision-Making." Washington University in St. Louis.

White, Halbert H. 1992. *Artificial Neural Networks, Approximation and Learning Theory.* Cambridge, MA: Blackwell.

Zeng, Langche. 1999. "Classification and Prediction with Neural Network Models." *Sociological Methods and Research.* 27(4):499–524.