Syntax

```
> m.out <- matchit(formula, data, method = "nearest", verbose = FALSE, ...)
```

Arguments

Arguments for All Matching Methods

- `formula`: formula used to calculate the distance measure for matching. It takes the usual syntax of R formulas, `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.

- `data`: the data frame containing the variables called in `formula`.

- `method`: the matching method (default = `"nearest"`, nearest neighbor matching). Currently, `"exact"` (exact matching), `"full"` (full matching), `"nearest"` (nearest neighbor matching), `"optimal"` (optimal matching), `"subclass"` (subclassification), and `"genetic"` (genetic matching) are available. Note that within each of these matching methods, MATCHIT offers a variety of options. See below for more details.

- `verbose`: a logical value indicating whether to print the status of the matching algorithm (default = `FALSE`).

Additional Arguments for Specification of Distance Measures The following arguments specify distance measures that are used for matching methods. These arguments apply to all matching methods *except exact matching*.

- `distance`: the method used to estimate the distance measure (default = `"logit"`, logistic regression) or a numerical vector of user's own distance measure. 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) estimate the propensity score, defined as the probability of receiving treatment, conditional on the covariates. Available methods include:

  - `"mahalanobis"`: the Mahalanobis distance measure.
  - binomial generalized linear models with one of the following link functions:
    * `"logit"`: logistic link
    * `"linear.logit"`: logistic link with linear propensity score)[1]
    * `"probit"`: probit link

---

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

* "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

– Choose one of the following generalized additive models (see `help(gam)` for more options).

* "GAMlogit": logistic link
* "GAMlinear.logit": logistic link with linear propensity score
* "GAMprobit": 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

– "nnet": neural network model. See `help(nnet)` for more options.

– "rpart": classification trees. See `help(rpart)` for more options.

- `distance.options`: optional arguments 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)`. Find additional options for general linear models using `help(glm)` or `help(family)`, for general additive models using `help(gam)`, for neutral network models `help(nnet)`, and for classification trees `help(rpart)`.

- `discard`: specifies whether to discard units that fall outside some measure of support of the distance measure (default = "none", discard no units). Discarding units may change the quantity of interest being estimated. Enter a logical vector indicating which unit should be discarded or choose from the following options:

– "none": no units will be discarded 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., **?**), 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.

– `"hull.both"`: all units that are not within the convex hull will be discarded. We recommend that this option be used with observational data sets.

– `"both"`: all units (treated and control) that are outside the support of the distance measure will be discarded.

– `"hull.control"`: only control units that are not within the convex hull of the treated units will be discarded.

– `"control"`: only control units outside the support of the distance measure of the treated units will be discarded. Use this option when the average treatment effect on the treated is of most interest and when you are unwilling to discard non-overlapping treatment units (which would change the quantity of interest).

– `"hull.treat"`: only treated units that are not within the convex hull of the control units will be discarded.

– `"treat"`: only treated units outside the support of the distance measure of the control units will be discarded. Use this option when the average treatment effect on the control units is of most interest and when unwilling to discard control units.

- `reestimate`: If `FALSE` (default), the model for the distance measure will not be re-estimated after units are discarded. The input must be a logical value. 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.

Additional Arguments for Subclassification

- `sub.by`: criteria for subclassification. Choose from: `"treat"` (default), the number of treatment units; `"control"`, the number of control units; or `"all"`, the total number of units.

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

Additional Arguments for Nearest Neighbor Matching

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

  – `"largest"` (default): matches from the largest value of the distance measure to the smallest.

  – `"smallest"`: matches from the smallest value of the distance measure to the largest.

– `"random"`: matches in random order.

- `replace`: logical value indicating whether each control unit can be matched to more than one treated unit (default = `replace = FALSE`, each control unit is used at most once – i.e., sampling without replacement). For matching with replacement, use `replace = TRUE`.

- `ratio`: 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`.

- `exact`: variables on which to perform exact matching within the nearest neighbor matching (default = `NULL`, no exact 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 (e.g., `exact = c("X1", "X2")`).

- `caliper`: the number of standard deviations of the distance measure within which to draw control units (default = 0, no caliper matching). If a caliper is specified, 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`: whether to take the nearest available match if no matches are available within the `caliper` (default = `FALSE`).

  – `mahvars`: variables on which to perform Mahalanobis-metric matching within each caliper (default = `NULL`). Variables should be entered as a vector of variable names (e.g., `mahvars = c("X1", "X2")`). If `mahvars` is specified without `caliper`, the caliper is set to 0.25.

- `subclass` and `sub.by`: See the options for subclassification 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.

Additional Arguments for Optimal Matching

- `ratio`: the number of control units to be matched to each treatment unit (default = 1).

- `...`: additional inputs that can be passed to the `fullmatch()` function in the `optmatch` package. See `help(fullmatch)` or http://www.stat.lsa.umich.edu/~bbh/optmatch.html for details.

Additional Arguments for Full Matching

- `...`: additional inputs that can be passed to the `fullmatch()` function in the `optmatch` package. See `help(fullmatch)` or http://www.stat.lsa.umich.edu/~bbh/optmatch.html for details.

Additional Arguments for Genetic Matching   The available options are listed below.

- `ratio`: the number of control units to be matched to each treatment unit (default = 1).

- `...`: additional minor inputs that can be passed to the `GenMatch()` function in the `Matching` package. See `help(GenMatch)` or http://sekhon.polisci.berkeley.edu/library/Matching/htm for details.

Output Values

Regardless of the type of matching performed, the `matchit` output object contains the following elements:[2]

- `call`: the original `matchit()` call.

- `formula`: the formula used to specify the model for estimating the distance measure.

- `model`: 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()`.

- `match.matrix`: an $n_1 \times$ `ratio` matrix where:
    - the row names represent the names of the treatment units (which match the row names of the data frame specified in `data`).
    - 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.

- `discarded`: 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.

- `distance`: a vector of length n with the estimated distance measure for each unit.

---

[2]When inapplicable or unnecessary, these elements may equal `NULL`. For example, when exact matching, `match.matrix = NULL`.

- **weights**: a vector of length n with 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.

- **subclass**: 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**.

- **q.cut**: the subclass cut-points that classify the distance measure.

- **treat**: the treatment indicator from **data** (the left-hand side of **formula**).

- **X**: 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**.

Contributors

If you use MATCHIT, please cite

and

The **convex.hull** discard option is implemented via the **WhatIf** package. If you use this option, please cite

and either or

Generalized linear distance measures are implemented via the **stats** package. If you use this distance measure, please cite

Generalized additive distance measures are implemented via the **mcgv** package. If you use this distance measure, please cite

The neural network distance measure is implemented via the **nnet** package. If you use this distance measure, please cite

The classification trees distance measure is implemented via the **rpart** package. If you use this distance measure, please cite

Full and optimal matching are implemented via the `optmatch` package by Ben Hansen. If you use either of these methods, please cite

Genetic matching is implemented via the `Matching` package by Jasjeet Sekhon. If you use this method, please cite