Title

stata.com

teffects nnmatch — Nearest-neighbor matching

Description

`teffects nnmatch` estimates the average treatment effect (ATE) and average treatment effect on the treated (ATET) from observational data by nearest-neighbor matching (NNM). NNM estimators impute the missing potential outcome for each subject by using an average of the outcomes of similar subjects that receive the other treatment level. Similarity between subjects is based on a weighted function of the covariates for each observation. The treatment effect is computed by taking the average of the difference between the observed and imputed potential outcomes for each subject. `teffects nnmatch` accepts a continuous, binary, count, fractional, or nonnegative outcome.

See [TE] **teffects intro** or [TE] **teffects intro advanced** for more information about estimating treatment effects from observational data.

Quick start

ATE of `treat` on `y` estimated by NNM on `x1` and indicators for levels of categorical variable `a`

```
teffects nnmatch (y x1 i.a) (treat)
```

As above, but estimate the ATET

```
teffects nnmatch (y x1 i.a) (treat), atet
```

Add continuous covariate `x2` and perform bias correction

```
teffects nnmatch (y x1 x2 i.a) (treat), biasadj(x1 x2)
```

As above, and match exactly on values of `a`

```
teffects nnmatch (y x1 x2 i.a) (treat), biasadj(x1 x2) ematch(i.a)
```

With robust standard errors

```
teffects nnmatch (y x1 x2 i.a) (treat), vce(robust)
```

With four matches per observation

```
teffects nnmatch (y x1 x2 i.a) (treat), nneighbor(4)
```

Menu

Statistics > Treatment effects > Continuous outcomes > Nearest-neighbor matching

Statistics > Treatment effects > Binary outcomes > Nearest-neighbor matching

Statistics > Treatment effects > Count outcomes > Nearest-neighbor matching

Statistics > Treatment effects > Fractional outcomes > Nearest-neighbor matching

Statistics > Treatment effects > Nonnegative outcomes > Nearest-neighbor matching

Syntax

```
teffects nnmatch (ovar omvarlist) (tvar) [if] [in] [weight]
    [, stat options]
```

ovar is a binary, count, continuous, fractional, or nonnegative outcome of interest.

omvarlist specifies the covariates in the outcome model.

tvar must contain integer values representing the treatment levels. Only two treatment levels are allowed.

stat	Description
Stat	
`ate`	estimate average treatment effect in population; the default
`atet`	estimate average treatment effect on the treated

options	Description
Model	
`nneighbor(#)`	specify number of matches per observation; default is `nneighbor(1)`
`biasadj(`*varlist*`)`	correct for large-sample bias using specified variables
`ematch(`*varlist*`)`	match exactly on specified variables
SE/Robust	
`vce(`*vcetype*`)`	*vcetype* may be `vce(robust [, nn(#)])`; use robust Abadie–Imbens standard errors with # matches `vce(iid)`; use independently and identically distributed Abadie–Imbens standard errors
Reporting	
`level(#)`	set confidence level; default is `level(95)`
`dmvariables`	display names of matching variables
display_options	control columns and column formats, row spacing, line width, display of omitted variables and base and empty cells, and factor-variable labeling
Advanced	
`caliper(#)`	specify the maximum distance for which two observations are potential neighbors
`dtolerance(#)`	set maximum distance between individuals considered equal
`osample(`*newvar*`)`	*newvar* identifies observations that violate the overlap assumption
`control(# | `*label*`)`	specify the level of *tvar* that is the control
`tlevel(# | `*label*`)`	specify the level of *tvar* that is the treatment
`generate(`*stub*`)`	generate variables containing the observation numbers of the nearest neighbors
`metric(`*metric*`)`	select distance metric for covariates
`coeflegend`	display legend instead of statistics

metric	Description
mahalanobis	inverse sample covariate covariance; the default
ivariance	inverse diagonal sample covariate covariance
euclidean	identity
matrix *matname*	user-supplied scaling matrix

omvarlist may contain factor variables; see [U] **11.4.3 Factor variables**.

by, collect, and statsby are allowed; see [U] **11.1.10 Prefix commands**.

fweights are allowed; see [U] **11.1.6 weight**.

coeflegend does not appear in the dialog box.

See [U] **20 Estimation and postestimation commands** for more capabilities of estimation commands.

Options

Model

nneighbor(#) specifies the number of matches per observation. The default is nneighbor(1). Each observation is matched with at least the specified number of observations from the other treatment level. nneighbor() must specify an integer greater than or equal to 1 but no larger than the number of observations in the smallest treatment group.

biasadj(*varlist*) specifies that a linear function of the specified covariates be used to correct for a large-sample bias that exists when matching on more than one continuous covariate. By default, no correction is performed.

Abadie and Imbens (2006, 2011) show that nearest-neighbor matching estimators are not consistent when matching on two or more continuous covariates and propose a bias-corrected estimator that is consistent. The correction term uses a linear function of variables specified in biasadj(); see example 3.

ematch(*varlist*) specifies that the variables in *varlist* match exactly. All variables in *varlist* must be numeric and may be specified as factors. teffects nnmatch exits with an error if any observations do not have the requested exact match.

Stat

stat is one of two statistics: ate or atet. ate is the default.

ate specifies that the average treatment effect be estimated.

atet specifies that the average treatment effect on the treated be estimated.

SE/Robust

vce(*vcetype*) specifies the standard errors that are reported. By default, teffects nnmatch uses robust standard errors estimated using two matches.

vce(robust [, nn(#)]) specifies that robust standard errors be reported and that the requested number of matches be used optionally.

vce(iid) specifies that standard errors for independently and identically distributed data be reported.

The standard derivative-based standard-error estimators cannot be used by teffects nnmatch, because these matching estimators are not differentiable. The implemented methods were derived by Abadie and Imbens (2006, 2011, 2012); see *Methods and formulas*.

As discussed in Abadie and Imbens (2008), bootstrap estimators do not provide reliable standard errors for the estimator implemented by `teffects nnmatch`.

Reporting

`level(#)`; see [R] **Estimation options**.

`dmvariables` specifies that the matching variables be displayed.

display_options: `noci`, `nopvalues`, `noomitted`, `vsquish`, `noemptycells`, `baselevels`, `allbaselevels`, `nofvlabel`, `fvwrap(#)`, `fvwrapon(`*style*`)`, `cformat(%`*fmt*`)`, `pformat(%`*fmt*`)`, `sformat(%`*fmt*`)`, and `nolstretch`; see [R] **Estimation options**.

Advanced

`caliper(#)` specifies the maximum distance at which two observations are a potential match. By default, all observations are potential matches regardless of how dissimilar they are.

The distance is based on *omvarlist*. If an observation does not have at least `nneighbor(#)` matches, `teffects nnmatch` exits with an error message. Use option `osample(`*newvar*`)` to identify all observations that are deficient in matches.

`dtolerance(#)` specifies the tolerance used to determine exact matches. The default value is `dtolerance(sqrt(c(epsdouble)))`.

Integer-valued variables are usually used for exact matching. The `dtolerance()` option is useful when continuous variables are used for exact matching.

`osample(`*newvar*`)` specifies that indicator variable *newvar* be created to identify observations that violate the overlap assumption. This variable will identify all observations that do not have at least `nneighbor(#)` matches in the opposite treatment group within `caliper(#)` (for `metric()` distance matching) or `dtolerance(#)` (for `ematch(`*varlist*`)` exact matches).

The `vce(robust, nn(#))` option also requires at least # matches in the same treatment group within the distance specified by `caliper(#)` or within the exact matches specified by `dtolerance(#)`.

The average treatment effect on the treated, option `atet`, using `vce(iid)` requires only `nneighbor(#)` control group matches for the treated group.

`control(#|`*label*`)` specifies the level of *tvar* that is the control. The default is the first treatment level. You may specify the numeric level # (a nonnegative integer) or the label associated with the numeric level. `control()` and `tlevel()` may not specify the same treatment level.

`tlevel(#|`*label*`)` specifies the level of *tvar* that is the treatment for the statistic `atet`. The default is the second treatment level. You may specify the numeric level # (a nonnegative integer) or the label associated with the numeric level. `tlevel()` may only be specified with statistic `atet`. `tlevel()` and `control()` may not specify the same treatment level.

`generate(`*stub*`)` specifies that the observation numbers of the nearest neighbors be stored in the new variables *stub*`1`, *stub*`2`, ... This option is required if you wish to perform postestimation based on the matching results. The number of variables generated may be more than `nneighbor(#)` because of tied distances. These variables may not already exist.

`metric(`*metric*`)` specifies the distance matrix used as the weight matrix in a quadratic form that transforms the multiple distances into a single distance measure; see *Nearest-neighbor matching estimator* in *Methods and formulas* for details.

The following option is available with `teffects nnmatch` but is not shown in the dialog box:

`coeflegend`; see [R] **Estimation options**.

Remarks and examples

stata.com

The NNM method of treatment-effect estimation imputes the missing potential outcome for each individual by using an average of the outcomes of similar subjects that receive the other treatment level. Similarity between subjects is based on a weighted function of the covariates for each observation. The average treatment effect (ATE) is computed by taking the average of the difference between the observed and potential outcomes for each subject.

`teffects nnmatch` determines the "nearest" by using a weighted function of the covariates for each observation. By default, the Mahalanobis distance is used, in which the weights are based on the inverse of the covariates' variance–covariance matrix. `teffects nnmatch` also allows you to request exact matching for categorical covariates. For example, you may want to force all matches to be of the same gender or race.

NNM is nonparametric in that no explicit functional form for either the outcome model or the treatment model is specified. This flexibility comes at a price; the estimator needs more data to get to the true value than an estimator that imposes a functional form. More formally, the NNM estimator converges to the true value at a rate slower than the parametric rate, which is the square root of the sample size, when matching on more than one continuous covariate. `teffects nnmatch` uses bias correction to fix this problem. `teffects psmatch` implements an alternative to bias correction; the method matches on a single continuous covariate, the estimated treatment probabilities. See [TE] **teffects intro** or [TE] **teffects intro advanced** for more information about this estimator.

We will illustrate the use of `teffects nnmatch` by using data from a study of the effect of a mother's smoking status during pregnancy (`mbsmoke`) on infant birthweight (`bweight`) as reported by Cattaneo (2010). This dataset also contains information about each mother's age (`mage`), education level (`medu`), marital status (`mmarried`), whether the first prenatal exam occurred in the first trimester (`prenatal1`), whether this baby was the mother's first birth (`fbaby`), and the father's age (`fage`).

▷ Example 1: Estimating the ATE

We begin by using `teffects nnmatch` to estimate the average treatment effect of `mbsmoke` on `bweight`. Subjects are matched using the Mahalanobis distance defined by covariates `mage`, `prenatal1`, `mmarried`, and `fbaby`.

```
. use https://www.stata-press.com/data/r17/cattaneo3
(Excerpt from Cattaneo (2010) Journal of Econometrics 155: 138-154)
. teffects nnmatch (bweight mage prenatal1 mmarried fbaby) (mbsmoke)
Treatment-effects estimation                   Number of obs      =      4,642
Estimator      : nearest-neighbor matching     Matches: requested =          1
Outcome model  : matching                                  min =          1
Distance metric: Mahalanobis                               max =        139
```

bweight	Coefficient	AI robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
mbsmoke (smoker vs nonsmoker)	-240.3306	28.43006	-8.45	0.000	-296.0525	-184.6087

The average birthweight if all mothers were to smoke would be 240 grams less than the average that would occur if none of the mothers had smoked.

◁

When the model includes indicator and categorical variables, you may want to restrict matches to only those subjects who are in the same category. The `ematch()` option of `teffects nnmatch` allows you to specify such variables that must match exactly.

Example 2: Exact matching

Here we use the `ematch()` option to require exact matches on the binary variables `prenatal1`, `mmarried`, and `fbaby`. We also use Euclidean distance, rather than the default Mahalanobis distance, to match on the continuous variable `mage`.

```
. teffects nnmatch (bweight mage) (mbsmoke),
> ematch(prenatal1 mmarried fbaby) metric(euclidean)
Treatment-effects estimation                   Number of obs      =      4,642
Estimator      : nearest-neighbor matching     Matches: requested =          1
Outcome model  : matching                                  min =          1
Distance metric: Euclidean                                 max =        139
```

bweight	Coefficient	AI robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
mbsmoke (smoker vs nonsmoker)	-240.3306	28.43006	-8.45	0.000	-296.0525	-184.6087

Abadie and Imbens (2006, 2011) have shown that nearest-neighbor matching estimators are not consistent when matching on two or more continuous covariates. A bias-corrected estimator that uses a linear function of variables can be specified with `biasadj()`.

Example 3: Bias adjustment

Here we match on two continuous variables, `mage` and `fage`, and we use the bias-adjusted estimator:

```
. teffects nnmatch (bweight mage fage) (mbsmoke),
> ematch(prenatal1 mmarried fbaby) biasadj(mage fage)
Treatment-effects estimation                   Number of obs      =      4,642
Estimator      : nearest-neighbor matching     Matches: requested =          1
Outcome model  : matching                                  min =          1
Distance metric: Mahalanobis                               max =         25
```

bweight	Coefficient	AI robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
mbsmoke (smoker vs nonsmoker)	-223.8389	26.19973	-8.54	0.000	-275.1894	-172.4883

These results are similar to those reported in example 1.

Example 4: NNM can reduce to RA

NNM reduces to RA when matching exactly and all the covariates are discrete. We begin our illustration of this point by estimating the ATE by NNM using exact matching on `mmarried` and the mother's age-categories `magecat`.

```
. teffects nnmatch (bweight) (mbsmoke), ematch(i.mmarried i.magecat)

Treatment-effects estimation                   Number of obs      =      4,642
Estimator      : nearest-neighbor matching     Matches: requested =          1
Outcome model  : matching                                 min =         11
Distance metric: Mahalanobis                              max =       1310
```

bweight	Coefficient	AI robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
mbsmoke (smoker vs nonsmoker)	-241.5264	24.39661	-9.90	0.000	-289.3429	-193.71

The RA estimator that includes the interactions among the discrete covariates produces the same point estimate.

```
. teffects ra (bweight i.mmarried##i.magecat) (mbsmoke)

Iteration 0:   EE criterion =  1.523e-23
Iteration 1:   EE criterion =  7.899e-26

Treatment-effects estimation                   Number of obs      =      4,642
Estimator      : regression adjustment
Outcome model  : linear
Treatment model: none
```

bweight	Coefficient	Robust std. err.	z	P>|z|	[95% conf.	interval]
ATE						
mbsmoke (smoker vs nonsmoker)	-241.5264	24.26233	-9.95	0.000	-289.0797	-193.9732
POmean						
mbsmoke nonsmoker	3403.651	9.492683	358.56	0.000	3385.046	3422.256

The two estimates of the ATE are the same. The standard errors differ in finite samples because the RA and NNM estimators use different robust estimators of the variance of the estimator.

With exact matching on discrete covariates, the NNM estimator reduces to an average of differences in cell means. With fully interacted discrete covariates, the RA estimator reduces to the same average of difference in cell means.

Video example

Treatment effects in Stata: Nearest-neighbor matching

Stored results

`teffects nnmatch` stores the following in `e()`:

Scalars

`e(N)`	number of observations
`e(n`*j*`)`	number of observations for treatment level *j*
`e(k_levels)`	number of levels in treatment variable
`e(treated)`	level of treatment variable defined as treated
`e(control)`	level of treatment variable defined as control
`e(k_nneighbor)`	requested number of matches
`e(k_nnmin)`	minimum number of matches
`e(k_nnmax)`	maximum number of matches
`e(k_robust)`	matches for robust VCE

Macros

`e(cmd)`	`teffects`
`e(cmdline)`	command as typed
`e(depvar)`	name of outcome variable
`e(tvar)`	name of treatment variable
`e(emvarlist)`	exact match variables
`e(bavarlist)`	variables used in bias adjustment
`e(mvarlist)`	match variables
`e(subcmd)`	`nnmatch`
`e(metric)`	`mahalanobis`, `ivariance`, `euclidean`, or `matrix` *matname*
`e(stat)`	statistic estimated, `ate` or `atet`
`e(wtype)`	weight type
`e(wexp)`	weight expression
`e(title)`	title in estimation output
`e(tlevels)`	levels of treatment variable
`e(vce)`	*vcetype* specified in `vce()`
`e(vcetype)`	title used to label Std. err.
`e(datasignature)`	the checksum
`e(datasignaturevars)`	variables used in calculation of checksum
`e(properties)`	`b V`
`e(estat_cmd)`	program used to implement `estat`
`e(predict)`	program used to implement `predict`
`e(marginsnotok)`	predictions disallowed by `margins`
`e(asbalanced)`	factor variables `fvset` as `asbalanced`
`e(asobserved)`	factor variables `fvset` as `asobserved`

Matrices

`e(b)`	coefficient vector
`e(V)`	variance–covariance matrix of the estimators

Functions

`e(sample)`	marks estimation sample

In addition to the above, the following is stored in `r()`:

Matrices

`r(table)`	matrix containing the coefficients with their standard errors, test statistics, p-values, and confidence intervals

Note that results stored in `r()` are updated when the command is replayed and will be replaced when any r-class command is run after the estimation command.

Methods and formulas

The methods and formulas presented here provide the technical details underlying the estimators implemented in `teffects nnmatch` and `teffects psmatch`. See *Methods and formulas* of [TE] **teffects aipw** for the methods and formulas used by `teffects aipw`, `teffects ipw`, `teffects ipwra`, and `teffects ra`.

Methods and formulas are presented under the following headings:

Nearest-neighbor matching estimator

`teffects nnmatch` implements the nearest-neighbor matching (NNM) estimator for the average treatment effect (ATE) and the average treatment effect on the treated (ATET). This estimator was derived by Abadie and Imbens (2006, 2011) and was previously implemented in Stata as discussed in Abadie et al. (2004).

`teffects psmatch` implements nearest-neighbor matching on an estimated propensity score. A propensity score is a conditional probability of treatment. The standard errors implemented in `teffects psmatch` were derived by Abadie and Imbens (2012).

`teffects nnmatch` and `teffects psmatch` permit two treatment levels: the treatment group with $t = 1$ and a control group with $t = 0$.

Matching estimators are based on the potential-outcome model, in which each individual has a well-defined outcome for each treatment level; see [TE] **teffects intro**. In the binary-treatment potential-outcome model, y_1 is the potential outcome obtained by an individual if given treatment-level 1 and y_0 is the potential outcome obtained by each individual i if given treatment-level 0. The problem posed by the potential-outcome model is that only y_{1i} or y_{0i} is observed, never both. y_{0i} and y_{1i} are realizations of the random variables y_0 and y_1. Throughout this document, i subscripts denote realizations of the corresponding, unsubscripted random variables.

Formally, the ATE is

$$\tau_1 = E(y_1 - y_0)$$

and the ATET is

$$\delta_1 = E(y_1 - y_0|t = 1)$$

These expressions imply that we must have some solution to the missing-data problem that arises because we only observe either y_{1i} or y_{0i}, not both.

For each individual, NNM uses an average of the individuals that are most similar, but get the other treatment level, to predict the unobserved potential outcome. NNM uses the covariates $\{x_1, x_2, \ldots, x_p\}$ to find the most similar individuals that get the other treatment level.

More formally, consider the vector of covariates $\mathbf{x}_i = \{x_{i,1}, x_{i,2}, \ldots, x_{i,p}\}$ and frequency weight w_i for observation i. The distance between $\mathbf{x}_i$ and $\mathbf{x}_j$ is parameterized by the vector norm

$$\|\mathbf{x}_i - \mathbf{x}_j\|_S = \{(\mathbf{x}_i - \mathbf{x}_j)'\mathbf{S}^{-1}(\mathbf{x}_i - \mathbf{x}_j)\}^{1/2}$$

where $\mathbf{S}$ is a given symmetric, positive-definite matrix.

Using this distance definition, we find that the set of nearest-neighbor indices for observation i is

$$\Omega^{\mathbf{x}}_m(i) = \{j_1, j_2, \ldots, j_{m_i} \mid t_{j_k} = 1 - t_i,\ \|\mathbf{x}_i - \mathbf{x}_{j_k}\|_S < \|\mathbf{x}_i - \mathbf{x}_l\|_S,\ t_l = 1 - t_i,\ l \neq j_k\}$$

Here m_i is the smallest number such that the number of elements in each set, $m_i = |\Omega^{\mathbf{x}}_m(i)| = \sum_{j\in\Omega^{\mathbf{x}}_m(i)} w_j$, is at least m, the desired number of matches. You set the size of m using the `nneighbor(#)` option. The number of matches for the ith observation may not equal m because of ties or if there are not enough observations with a distance from observation i within the caliper limit, c, $\|\mathbf{x}_i - \mathbf{x}_j\|_S \leq c$. You may set the caliper limit by using the `caliper(#)` option. For ease of notation, we will use the abbreviation $\Omega(i) = \Omega^{\mathbf{x}}_m(i)$.

With the `metric(`*string*`)` option, you have three choices for the scaling matrix $\mathbf{S}$: Mahalanobis, inverse variance, or Euclidean.

$$\mathbf{S} = \begin{cases} \dfrac{(\mathbf{X} - \overline{\mathbf{x}}'\mathbf{1}_n)'\mathbf{W}(\mathbf{X} - \overline{\mathbf{x}}'\mathbf{1}_n)}{\sum_i^n w_i - 1} & \text{if } \texttt{metric} = \texttt{mahalanobis} \\ \text{diag}\left\{\dfrac{(\mathbf{X} - \overline{\mathbf{x}}'\mathbf{1}_n)'\mathbf{W}(\mathbf{X} - \overline{\mathbf{x}}'\mathbf{1}_n)}{\sum_i^n w_i - 1}\right\} & \text{if } \texttt{metric} = \texttt{ivariance} \\ \mathbf{I}_p & \text{if } \texttt{metric} = \texttt{euclidean} \end{cases}$$

where $\mathbf{1}_n$ is an $n \times 1$ vector of ones, $\mathbf{I}_p$ is the identity matrix of order p, $\overline{\mathbf{x}} = (\sum_i^n w_i\mathbf{x}_i)/(\sum_i^n w_i)$, and $\mathbf{W}$ is an $n \times n$ diagonal matrix containing frequency weights.

The NNM method predicting the potential outcome for the ith observation as a function of the observed y_i is

$$\widehat{y}_{ti} = \begin{cases} y_i & \text{if } t_i = t \\ \dfrac{\sum\limits_{j\in\Omega(i)} w_j y_j}{\sum\limits_{j\in\Omega(i)} w_j} & \text{otherwise} \end{cases}$$

for $t \in \{0, 1\}$.

We are now set to provide formulas for estimates $\widehat{\tau}_1$, the ATE, and $\widehat{\delta}_1$, the ATET,

$$\widehat{\tau}_1 = \frac{\sum_{i=1}^n w_i(\widehat{y}_{1i} - \widehat{y}_{0i})}{\sum_{i=1}^n w_i} = \frac{\sum_{i=1}^n w_i(2t_i - 1)\{1 + K_m(i)\}y_i}{\sum_{i=1}^n w_i}$$

$$\widehat{\delta}_1 = \frac{\sum_{i=1}^n t_i w_i(\widehat{y}_{1i} - \widehat{y}_{0i})}{\sum_{i=1}^n t_i w_i} = \frac{\sum_{i=1}^n \{t_i - (1 - t_i)K_m(i)\}y_i}{\sum_{i=1}^n t_i w_i}$$

where

$$K_m(i) = \sum_{j=1}^n I\{i \in \Omega(j)\}\frac{w_j}{\sum\limits_{k\in\Omega(j)} w_k}$$

The estimated variance of $\widehat{\tau}_1$ and $\widehat{\delta}_1$ are computed as

$$\widehat{\sigma}^2_\tau = \frac{\sum_{i=1}^n w_i\left[(\widehat{y}_{1i} - \widehat{y}_{0i} - \widehat{\tau}_1)^2 + \widehat{\xi}_i^2\{K_m^2(i) + 2K_m(i) - K'_m(i)\}\right]}{\left(\sum_{i=1}^n w_i\right)^2}$$

$$\widehat{\sigma}^2_\delta = \frac{\sum_{i=1}^n t_i w_i\left[(\widehat{y}_{1i} - \widehat{y}_{0i} - \widehat{\delta}_1)^2 + \widehat{\xi}_i^2\{K_m^2(i) - K'_m(i)\}\right]}{\left(\sum_{i=1}^n t_i w_i\right)^2}$$

where

$$K'_m(i) = \sum_{j=1}^{n} I\{i \in \Omega(j)\} \frac{w_j}{\left(\sum_{k\in\Omega(j)} w_k\right)^2}$$

and $\xi_i^2 = \text{var}(y_{ti}|\mathbf{x}_i)$ is the conditional outcome variance. If we can assume that ξ_i^2 does not vary with the covariates or treatment (homoskedastic), then we can compute an ATE estimate of ξ_τ^2 as

$$\widehat{\xi}_\tau^2 = \frac{1}{2\sum_i^n w_i} \sum_{i=1}^{n} w_i \left[\frac{\sum_{j\in\Omega(i)} w_j\{y_i - y_j(1-t_i) - \widehat{\tau}_1\}^2}{\sum_{j\in\Omega(i)} w_j} \right]$$

and an ATET estimate of ξ_δ^2 as

$$\widehat{\xi}_\delta^2 = \frac{1}{2\sum_i^n t_i w_i} \sum_{i=1}^{n} t_i w_i \left[\frac{\sum_{j\in\Omega(i)} t_j w_j\{y_i - y_j(1-t_i) - \widehat{\delta}_1\}^2}{\sum_{j\in\Omega(i)} t_j w_j} \right]$$

If the conditional outcome variance is dependent on the covariates or treatment, we require an estimate for ξ_i^2 at each observation. In this case, we require a second matching procedure, where we match on observations within the same treatment group.

Define the within-treatment matching set

$$\Psi_h^{\mathbf{x}}(i) = \{j_1, j_2, \ldots, j_{h_i} \mid t_{j_k} = t_i,\ \|\mathbf{x}_i - \mathbf{x}_{j_k}\|_S < \|\mathbf{x}_i - \mathbf{x}_l\|_S,\ t_l = t_i,\ l \neq j_k\}$$

where h is the desired set size. As before, the number of elements in each set, $h_i = |\Psi_h^{\mathbf{x}}(i)|$, may vary depending on ties and the value of the caliper. You set h using the `vce(robust, nn(#))` option. As before, we will use the abbreviation $\Psi(i) = \Psi_h^{\mathbf{x}}(i)$ where convenient.

We estimate ξ_i^2 by

$$\widehat{\xi}_{t_i}^2(\mathbf{x}_i) = \frac{\sum_{j\in\Psi(i)} w_j(y_j - \overline{y}_{\Psi i})^2}{\sum_{j\in\Psi(i)} w_j - 1} \qquad \text{where} \qquad \overline{y}_{\Psi i} = \frac{\sum_{j\in\Psi(i)} w_j y_j}{\sum_{j\in\Psi(i)} w_j - 1}$$

Bias-corrected matching estimator

When matching on more than one continuous covariate, the matching estimator described above is biased, even in infinitely large samples; in other words, it is not $\sqrt{n}$-consistent; see Abadie and Imbens (2006, 2011). Following Abadie and Imbens (2011) and Abadie et al. (2004), `teffects nnmatch` makes an adjustment based on the regression functions $\mu_t(\widetilde{\mathbf{x}}_i) = E(y_t \mid \widetilde{\mathbf{x}} = \widetilde{\mathbf{x}}_i)$, for $t = 0, 1$ and the set of covariates $\widetilde{\mathbf{x}}_i = (\widetilde{x}_{i,1}, \ldots, \widetilde{x}_{i,q})$. The bias-correction covariates may be the same as the NNM covariates $\mathbf{x}_i$. We denote the least-squares estimates as $\widehat{\mu}_t(\widetilde{\mathbf{x}}_i) = \widehat{\nu}_t + \widehat{\boldsymbol{\beta}}_t'\widetilde{\mathbf{x}}_i$, where we regress $\{y_i \mid t_i = t\}$ onto $\{\widetilde{\mathbf{x}}_i \mid t_i = t\}$ with weights $w_i K_m(i)$, for $t = 0, 1$.

Given the estimated regression functions, the bias-corrected predictions for the potential outcomes are computed as

$$\widehat{y}_{ti} = \begin{cases} y_i & \text{if } t_i = t \\ \dfrac{\displaystyle\sum_{j \in \Omega^{\mathbf{x}}_m(i)} w_j\{y_j + \widehat{\mu}_t(\widetilde{\mathbf{x}}_i) - \widehat{\mu}_t(\widetilde{\mathbf{x}}_j)\}}{\displaystyle\sum_{j \in \Omega^{\mathbf{x}}_m(i)} w_j} & \text{otherwise} \end{cases}$$

The `biasadj(`*varlist*`)` option specifies the bias-adjustment covariates $\widetilde{\mathbf{x}}_i$.

Propensity-score matching estimator

The propensity-score matching (PSM) estimator uses a treatment model (TM), $p(\mathbf{z}_i, t, \boldsymbol{\gamma})$, to model the conditional probability that observation i receives treatment t given covariates $\mathbf{z}_i$. The literature calls $p(\mathbf{z}_i, t, \boldsymbol{\gamma})$ a propensity score, and PSM matches on the estimated propensity scores.

When matching on the estimated propensity score, the set of nearest-neighbor indices for observation i, $i = 1, \ldots, n$, is

$$\Omega^{\mathrm{P}}_m(i) = \{j_1, j_2, \ldots, j_{m_i} \mid t_{j_k} = 1 - t_i,\ |\widehat{p}_i(t) - \widehat{p}_{j_k}(t)| < |\widehat{p}_i(t) - \widehat{p}_l(t)|,\ t_l = 1 - t_i, l \neq j_k\}$$

where $\widehat{p}_i(t) = p(\mathbf{z}_i, t, \widehat{\boldsymbol{\gamma}})$. As was the case with the NNM estimator, m_i is the smallest number such that the number of elements in each set, $m_i = |\Omega^{\mathrm{P}}_m(i)| = \sum_{j \in \Omega^{\mathrm{P}}_m(i)} w_j$, is at least m, the desired number of matches, set by the `nneighbor(#)` option.

We define the within-treatment matching set analogously,

$$\Psi^{\mathrm{P}}_h(i) = \{j_1, j_2, \ldots, j_{h_i} \mid t_{j_k} = t_i,\ |\widehat{p}_i(t) - \widehat{p}_{j_k}(t)| < |\widehat{p}_i(t) - \widehat{p}_l(t)|,\ t_l = t_i,\ l \neq j_k\}$$

where h is the desired number of within-treatment matches, and $h_i = |\Psi^{\mathrm{P}}_h(i)|$, for $i = 1, \ldots, n$, may vary depending on ties and the value of the caliper. The sets $\Psi^{\mathrm{P}}_h(i)$ are required to compute standard errors for $\widehat{\tau}_1$ and $\widehat{\delta}_1$.

Once a matching set is computed for each observation, the potential-outcome mean, ATE, and ATET computations are identical to those of NNM. The ATE and ATET standard errors, however, must be adjusted because the TM parameters were estimated; see Abadie and Imbens (2012).

PSM, ATE, and ATET variance adjustment

The variances for $\widehat{\tau}_1$ and $\widehat{\delta}_1$ must be adjusted because we use $\widehat{\boldsymbol{\gamma}}$ instead of $\boldsymbol{\gamma}$. The adjusted variances for $\widehat{\tau}_1$ and $\widehat{\delta}_1$ have the following forms, respectively:

$$\widehat{\sigma}^2_{\tau,\mathrm{adj}} = \widehat{\sigma}^2_{\tau} + \widehat{\mathbf{c}}'_{\tau}\widehat{\mathbf{V}}_{\gamma}\widehat{\mathbf{c}}_{\tau}$$

$$\widehat{\sigma}^2_{\delta,\mathrm{adj}} = \widehat{\sigma}^2_{\delta} - \widehat{\mathbf{c}}'_{\delta}\widehat{\mathbf{V}}_{\gamma}\widehat{\mathbf{c}}_{\delta} + \widehat{\frac{\partial\delta_1}{\partial\boldsymbol{\gamma}'}}\widehat{\mathbf{V}}_{\gamma}\widehat{\frac{\partial\delta_1}{\partial\boldsymbol{\gamma}}}$$

In both equations, the matrix $\widehat{\mathbf{V}}_{\gamma}$ is the TM coefficient variance–covariance matrix.

The adjustment term for ATE can be expressed as

$$\widehat{\mathbf{c}}_\tau = \frac{1}{\sum_{i=1}^n w_i} \sum_{i=1}^n w_i f(\mathbf{z}_i'\widehat{\boldsymbol{\gamma}}) \left(\frac{\widehat{\text{cov}}\left(\mathbf{z}_i, \widehat{y}_{i1}\right)}{\widehat{p}_i(1)} + \frac{\widehat{\text{cov}}\left(\mathbf{z}_i, \widehat{y}_{i0}\right)}{\widehat{p}_i(0)} \right)$$

where

$$f(\mathbf{z}_i'\widehat{\boldsymbol{\gamma}}) = \frac{d\, p(\mathbf{z}_i, 1, \widehat{\boldsymbol{\gamma}})}{d(\mathbf{z}_i'\widehat{\boldsymbol{\gamma}})}$$

is the derivative of $p(\mathbf{z}_i, 1, \widehat{\boldsymbol{\gamma}})$ with respect to $\mathbf{z}_i'\widehat{\boldsymbol{\gamma}}$, and

$$\widehat{\text{cov}}\left(\mathbf{z}_i, \widehat{y}_{ti}\right) = \begin{cases} \dfrac{\sum\limits_{j \in \Psi_h(i)} w_j (\mathbf{z}_j - \overline{\mathbf{z}}_{\Psi i})(y_j - \overline{y}_{\Psi i})}{\sum\limits_{j \in \Psi_h(i)} w_j - 1} & \text{if } t_i = t \\ \dfrac{\sum\limits_{j \in \Omega_h(i)} w_j (\mathbf{z}_j - \overline{\mathbf{z}}_{\Omega i})(y_j - \overline{y}_{\Omega i})}{\sum\limits_{j \in \Omega_h(i)} w_j - 1} & \text{otherwise} \end{cases}$$

is a $p \times 1$ vector with

$$\overline{\mathbf{z}}_{\Psi i} = \frac{\sum\limits_{j \in \Psi_h(i)} w_j \mathbf{z}_j}{\sum\limits_{j \in \Psi_h(i)} w_j} \quad \overline{\mathbf{z}}_{\Omega i} = \frac{\sum\limits_{j \in \Omega_h(i)} w_j \mathbf{z}_j}{\sum\limits_{j \in \Omega_h(i)} w_j} \quad \text{and} \quad \overline{y}_{\Omega i} = \frac{\sum\limits_{j \in \Omega_h(i)} w_j y_j}{\sum\limits_{j \in \Omega_h(i)} w_j}$$

Here we have used the notation $\Psi_h(i) = \Psi_h^{\text{p}}(i)$ and $\Omega_h(i) = \Omega_h^{\text{p}}(i)$ to stress that the within-treatment and opposite-treatment clusters used in computing $\widehat{\sigma}^2_{\tau,\text{adj}}$ and $\widehat{\delta}^2_{\tau,\text{adj}}$ are based on h instead of the cluster $\Omega_m^{\text{p}}(i)$ based on m used to compute $\widehat{\tau}_1$ and $\widehat{\delta}_1$, although you may desire to have $h = m$.

The adjustment term $\mathbf{c}_\delta$ for the ATET estimate has two components, $\mathbf{c}_\delta = \mathbf{c}_{\delta,1} + \mathbf{c}_{\delta,2}$, defined as

$$\mathbf{c}_{\delta,1} = \frac{1}{\sum_{i=1}^n t_i w_i} \sum_{i=1}^n w_i \mathbf{z}_i f(\mathbf{z}_i'\widehat{\boldsymbol{\gamma}}) \left(\widetilde{y}_{1i} - \widetilde{y}_{0i} - \widehat{\delta}_1 \right)$$

$$\mathbf{c}_{\delta,2} = \frac{1}{\sum_{i=1}^n t_i w_i} \sum_{i=1}^n w_i f(\mathbf{z}_i'\widehat{\boldsymbol{\gamma}}) \left\{ \widehat{\text{cov}}\left(\mathbf{z}_i, \widehat{y}_{1i}\right) + \frac{\widehat{p}_i(1)}{\widehat{p}_i(0)} \widehat{\text{cov}}\left(\mathbf{z}_i, \widehat{y}_{0i}\right) \right\}$$

where

$$\widetilde{y}_{ti} = \begin{cases} \dfrac{\sum\limits_{j \in \Psi_h(-i)} w_j y_j}{\sum\limits_{j \in \Psi_h(-i)} w_j} & \text{if } t = t_i \\ \dfrac{\sum\limits_{j \in \Omega_h} w_j y_j}{\sum\limits_{j \in \Omega_h} w_j} & \text{otherwise} \end{cases}$$

and the within-treatment matching sets $\Psi_h(-i) = \Psi_h^{\text{p}}(-i)$ are similar to $\Psi_h^{\text{p}}(i)$ but exclude observation i:

$$\Psi_h^{\text{p}}(-i) = \{j_1, j_2, \ldots, j_{h_i} \mid j_k \neq i,\ t_{j_k} = t_i,\ |\widehat{p}_i - \widehat{p}_{j_k}| < |\widehat{p}_i - \widehat{p}_l|,\ t_l = t_i,\ l \notin \{i, j_k\}\}$$

Finally, we cover the computation of $\widehat{\frac{\partial\delta_1}{\partial\boldsymbol{\gamma}'}}$ in the third term on the right-hand side of $\widehat{\sigma}^2_{\delta,\mathrm{adj}}$. Here we require yet another clustering, but we match on the opposite treatment by using the covariates $\mathbf{z}_i = (z_{i,1}, \ldots, z_{i,p})'$. We will denote these cluster sets as $\Omega^{\mathbf{z}}_m(i)$, for $i = 1, \ldots, n$.

The estimator of the $p \times 1$ vector $(\partial\delta_1)/(\partial\boldsymbol{\gamma}')$ is computed as

$$\widehat{\frac{\partial\delta_1}{\partial\boldsymbol{\gamma}'}} = \frac{1}{\sum_i^n t_i w_i} \sum_{i=1}^{n} \mathbf{z}_i f(\mathbf{z}'\widehat{\boldsymbol{\gamma}}) \left\{ (2t_i - 1)(y_i - \overline{y}_{\Omega^{\mathbf{z}}_m i}) - \widehat{\delta}_1 \right\}$$

where

$$\overline{y}_{\Omega^{\mathbf{z}}_m i} = \frac{\sum_{j \in \Omega^{\mathbf{z}}_m(i)} w_j y_j}{\sum_{j \in \Omega^{\mathbf{z}}_m(i)} w_j}$$

References

Abadie, A., D. M. Drukker, J. L. Herr, and G. W. Imbens. 2004. Implementing matching estimators for average treatment effects in Stata. *Stata Journal* 4: 290–311.

Abadie, A., and G. W. Imbens. 2006. Large sample properties of matching estimators for average treatment effects. *Econometrica* 74: 235–267. https://doi.org/10.1111/j.1468-0262.2006.00655.x.

——. 2008. On the failure of the bootstrap for matching estimators. *Econometrica* 76: 1537–1557. https://doi.org/10.3982/ECTA6474.

——. 2011. Bias-corrected matching estimators for average treatment effects. *Journal of Business & Economic Statistics* 29: 1–11. https://doi.org/10.1198/jbes.2009.07333.

——. 2012. Matching on the estimated propensity score. Harvard University and National Bureau of Economic Research. http://www.nber.org/papers/w15301.

Cattaneo, M. D. 2010. Efficient semiparametric estimation of multi-valued treatment effects under ignorability. *Journal of Econometrics* 155: 138–154. https://doi.org/10.1016/j.jeconom.2009.09.023.

Díaz, J. D., I. Gutiérrez, and J. Rivera. 2021. Implementing blopmatching in Stata. *Stata Journal* 21: 180–194.

Drukker, D. M. 2016. Exact matching on discrete covariates is the same as regression adjustment. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2016/08/16/exact-matching-on-discrete-covariates-is-the-same-as-regression-adjustment/.

Huber, C. 2015. Introduction to treatment effects in Stata: Part 2. *The Stata Blog: Not Elsewhere Classified.* http://blog.stata.com/2015/08/24/introduction-to-treatment-effects-in-stata-part-2/.

Also see

[TE] **teffects postestimation** — Postestimation tools for teffects

[TE] **teffects** — Treatment-effects estimation for observational data

[TE] **teffects psmatch** — Propensity-score matching

[U] **20 Estimation and postestimation commands**