Stata Commands for Matching and Weighting Estimators of Treatment Effects

`teffects` is a suite of commands that estimate ATEs, ATTs, and potential outcome means using different methods for constructing counterfactuals. `psmatch2` is an older command used for propensity score matching that for some purposes is preferable to `teffects`. Both commands are described below.

1. **Exact matching** using `teffects nnmatch`:

   ```
   teffects nnmatch (y) (treat), ematch(x1 x2) atet
   ```

   This command will perform exact matching for treated and untreated cases based on the pre-treatment covariates *x1* and *x2*. It will then calculate the ATT for the outcome *y*. Use the option `ate` for the ATE. *treat* is the treatment indicator. The default number of "nearest neighbor" matches in `nnmatch` is 1, but in the case of exact matching, `teffects` will use all available exact matches. For exact matching to work for ATT, there must be at least one exact match for *every* treated observation. For the default standard error calculation to work, at least two exact matches are required.

   Include the option `gen(stubname)` to have Stata create new variables beginning with *stubname* that contain the observation number(s) of every nearest neighbor match (exact match, in this case). Since the sort order of a dataset can change—which would change the observation number of the nearest neighbor—you may want to insert code like the following before any matching commands; this will capture the current sort order as the variable *obsno*.

   ```
   gen obsno=_n
   ```

2. **Nearest neighbor matching** using `teffects nnmatch`:

   ```
   teffects nnmatch (y x1 x2 x3) (treat), atet
   ```

   This command will perform nearest neighbor matching for treated and untreated cases based on the pre-treatment covariates *x1, x2* and *x3*. It will then calculate the ATT for the outcome *y*. Use the option `ate` for the ATE. The default number of matches is 1, but this can be adjusted using the option `nneighbor(#)`. Nearest neighbor matching can be combined with exact matching on a subset of covariates (use the `ematch` option as above).

   The default distance metric is Mahalanobis, however Euclidean distance is possible using the option `metric(euclidean)`. You can set the maximum distance allowed for two observations to be considered nearest neighbors with the option `caliper(#)`. Again, the option `gen(stubname)` will create new variables that contain the observation numbers of nearest neighbor matches.

   The option `biasadj(varlist)` will perform a large-sample bias correction based on Abadie & Imbens (2006, 2011). This involves using OLS to estimate the relationship between *y* and the variables in *varlist* and then using the predicted *y*s in the adjustment. This adjusts for the fact that nearest neighbor matches are not exact matches. It is recommended that you include your continuous covariates in *varlist*.

3. **Propensity score matching** using `teffects psmatch`:

   ```
   teffects psmatch (y) (treat x1 x2 x3, tmodel), atet
   ```

   This command will perform nearest neighbor matching for treated and untreated cases using estimated propensity scores. The propensity score model is ***tmodel*** (e.g., `logit` or `probit`, where logit is the default) with *treat* the binary outcome and *x1, x2,* etc., the predictors of treatment. The default number of matches is 1, but this can be adjusted using the option `nneighbor(#)`. You can set the maximum distance allowed for two observations to be considered nearest neighbors with the option `caliper(#)`. Again, the option `gen(stubname)` will create new variables that contain the observation numbers of nearest neighbor matches.

   It is good practice to *not* see the ATT (or ATE) before settling on a propensity score model. To suppress the reporting of the treatment effect, you can precede the `teffects` command with `quietly:` and then conduct diagnostics (like `tebalance`) as described below.

4. **Inverse probability weighting** using `teffects ipw`:

   ```
   teffects ipw (y) (treat x1 x2 x3, tmodel), atet
   ```

   This command will compute average treatment effects using inverse probability weighting (where the estimated propensity scores are used in the weights). The syntax is very similar to `teffects psmatch`.

5. Checking balance in covariates using `tebalance summarize`:

   ```
   tebalance summarize x1 x2 x3
   ```

   After `teffects nnmatch` or `teffects psmatch`, this command will report standardized differences in means and variance ratios between the treated and matched untreated cases. *x1, x2,* etc., represent covariates that may or may not have been used in the original matching. Including the option `baseline` will report the means and variances of the variables *x1, x2, x3*, separately for the treated and untreated group at "baseline" (not conditional on the matched sample).

6. Checking balance using `tebalance box`:

   ```
   tebalance box x1
   ```

   After `teffects nnmatch` or `teffects psmatch`, this command will generate boxplots for the covariate *x1*, separately for the treated and untreated cases. Both the raw (baseline) distributions and matched sample distributions are shown. To generate boxplots for estimated propensity scores, just use `tebalance box` alone after `teffects psmatch`.

7. Checking overlap in treatment probabilities using `teffects overlap`:

   ```
   teffects overlap, ptlevel(#)
   ```

   After `teffects psmatch` or `teffects ipw`, this command will show densities for the estimated propensity scores, separately for treated and untreated cases on the same graph. Note this command uses all observations, not just those in a matched sample. In the `ptlevel` option, specify which value of your *treat* variable corresponds to treatment (usually 1). You can also produce densities of *ps1* and *ps0* (the estimated probabilities of being treated and untreated, respectively) for the treated or untreated group alone using the option `tlevels(#)` (where in most cases # corresponds to 0 for the untreated group and 1 for the treated group).

8. Creating new variables from `teffects`:

   There are many new variables that can be created after `teffects` estimation. The options vary depending on the type of estimation (e.g., `nnmatch, psmatch, ipw`). The generic syntax is:

   ```
   predict newvar, statistic
   ```

   For example, after `nnmatch`, you can create new variables that contain `po` (the potential outcome), `te` (the estimated treatment effect), or `distance` (the distance to the nearest neighbor). `po, te,` and `distance` are the *statistic* specified in the `predict` command. The potential outcome (*po*) is the imputed counterfactual from the nearest neighbor match(es). The treatment effect (*te*) is the difference between the actual and counterfactual outcome.

   After `psmatch` you can also create a new variable that contains the estimated propensity score *ps*. For example, the 2nd line below creates variables for both $1-\widehat{P(X)}$ and $\widehat{P(X)}$:

   ```
   teffects psmatch (y) (treat x1 x2 x3), atet
   predict ps0 ps1, ps /* for propensity scores 1-p(x) and p(x) */
   predict po0 po1, po /* for potential outcomes y0 and y1 */
   predict te /* for treatment effect */
   ```

9. **Propensity score matching** using psmatch2:

   psmatch2 without requesting a treatment effect estimate will perform nearest neighbor matching using estimated propensity scores:

   ```
   psmatch2 treat x1 x2 x3
   ```

treat is the treatment indicator and $x_1 - x_3$ are pre-treatment covariates used for matching. Unlike `teffects psmatch`, the default propensity score model is probit rather than logit, but the option `logit` can change this. Other useful options for refining your matches:

- `neighbor(#)` sets the number of nearest neighbors desired
- `noreplacement` uses nearest neighbor matching *without* replacement
- `ties`: if there are ties, uses all tied cases as matches (otherwise will depend on the sort order). Note this is the default in `teffects`.
- `caliper(#)` sets the maximum distance allowed for matches
- `radius` will perform radius matching, using the caliper set above
- `odds`: matches based on the log odds of the propensity score, $ln(p(x)/(1-p(x)))$

10. Useful variables created by `psmatch2`:

    - `_pscore`: the estimated propensity score
    - `_treated`: = 1 for treated cases and = 0 otherwise (including unmatched cases)
    - `_support`: = 1 for cases on the common support (often every observation)
    - `_weight`: = 1 for treated cases and a weight for matched untreated cases. With nearest neighbor matching and no untreated case used more than once, the matched untreated cases will all have a weight of 1. If untreated matched cases are used more than once, their weight will exceed 1. If *ties* are permitted, weights can be fractional, since multiple matched cases provide the counterfactual.
    - `_id`: ID number assigned to a case for identifying matches
    - `_n1`: ID number for nearest neighbor (populated for treated cases only). If more than one nearest neighbor is requested, there will be as many of these variables created as the number of neighbors k, e.g. `_n1, _n2, ..., _nk`
    - `_nn`: number of matched neighbors. Will be non-zero for treated cases only, and equal to the number of nearest neighbors requested.
    - `_pdif`: absolute value of the difference in propensity score vs. nearest neighbor (populated for treated cases only). If more than one nearest neighbor is requested, only the difference with the closest neighbor is provided.

11. Checking overlap and balance after `psmatch2` using `psgraph` and `pstest`:

    `psgraph` will provide a histogram of propensity scores for the treated and untreated cases, using all data (not just the matched cases). You can adjust the number of bins with the `bin(#)` option. `pstest` will compare the means and variances of covariates that you choose:

```
pstest x1 x2 x3 x4
```

`pstest` is comparable to `tebalance summarize`. `psgraph` is comparable to `teffects overlap`. You can use the *_weight* and *_pscore* variables created by `psmatch2` to generate your own densities/histograms of estimated propensity scores for the *matched* sample.

12. Treatment effect estimation using `psmatch2`:

    Once you have finalized your propensity score model, you can request the treatment effect estimate using the `outcome( )` option:

    ```
    psmatch2 treat x1 x2 x3, outcome(y)
    ```

    The default is to compute the ATT. The option `ate` will tell Stata to also compute the overall ATE and the ATU. With nearest neighbor matching, the ATT will be a simple difference in means, where the mean for the untreated cases is weighted. For example, the following will report the same group means, after `psmatch2`:

    ```
    tabstat y [weight=_weight], by(_treat) stat(n mean)
    ```

    Important note: the standard error formula used for the treatment effect estimate in `psmatch2` is incorrect, since it does not account for the fact that the propensity score is estimated (Abadie & Imbens, 2012). This suggests that one should use `teffects` to do the final estimation.

13. **Nearest neighbor (Mahalanobis) matching** using `psmatch2`:

    Rather than propensity scores, one can match using the Mahalanobis measure d. d is a measure of the distance between two vectors x and x' in multivariate space:

    $$d = ||x, x'|| = (x - x')'\hat{\Omega}_X^{-1}(x - x')$$

    where $\hat{\Omega}_X^{-1}$ is the sample covariance matrix of the covariates. Intuitively, the role of the covariance matrix is to scale the distance between x values (kind of like when we divide by the standard deviation when standardizing a univariate x), and to take correlations between the x into account. To perform Mahalanobis matching in `psmatch2`:

    ```
    psmatch2 treat, mahalanobis(x1 x2 x3)
    ```

    Note in this statement the matching covariates have been moved to the `mahalanobis` option. In the same way as for propensity score matching, you can use other options to select the number of nearest neighbors, request the treatment effect estimate, etc.

14. **Regression** after `psmatch2` on the matched sample:

    Some analysts wish to fit a regression model using your matched sample. After `psmatch2` simply use the created `_weight` variable in your regression command to account for the fact that some untreated observations are matched more than once:

    ```
    psmatch2 treat x1 x2 x3
    reg y treat x1 x2 x2 [fweight=_weight]
    ```

    Unfortunately, `teffects` does not create such a weight variable, so you must create one yourself. The code below provides an example of how to do this:

```
**example with 5 nearest neighbors stored as nn*
teffects psmatch (y) (treat x1 x2 x2), atet gen(nn) nn(5)
gen obs=_n

**create a tempfile containing the neighbor obs# and count of matches
preserve
tempfile treated
keep if treat==1
keep nn* obs
reshape long nn, i(obs) j(mnum)
bysort nn: gen weight=_N
by nn: keep if _n==1
drop obs mnum
rename nn obs
save ‘treated’
restore
merge m:1 obs using ‘treated’
replace weight=1 if treat==1

**the resulting "weight" represents the count of times an observation was used
**as a nearest neighbor (and is equal to one for treat=1)
table treat weight, col
```