## Title

**scdatamulti** — Data Preparation for Synthetic Control Methods with Staggered Adoption.

## Syntax

**scdatamulti** *features* [if] [in] , **id(***idvar***) time(***timevar***) outcome(***outcomevar***) treatment(***treatmentvar***) dfname(***string***)** [**covadj(***string***) cointegrated(***string***) constant(***string***) anticipation(***string***) effect(***string***) post_est(***string***) units_est(***string***) donors_est(***string***) pypinocheck**]

## Description

**scdatamulti** prepares the data to be used by scest or scpi to implement estimation and inference procedures for Synthetic Control (SC) methods in the general case of multiple treated units and staggered adoption. It allows the user to specify for each treated unit the features to be matched, covariate-adjustment feature by feature, anticipation effects, and presence of cointegration. The command follows the terminology proposed in Cattaneo, Feng, and Titiunik (2021) for a single treated unit and Cattaneo, Feng, Palomba, and Titiunik (2023) for multiple treated units and staggered adoption. The command is a wrapper of the companion Python package. As such, the user needs to have a running version of Python with the package installed. A tutorial on how to install Python and link it to Stata can be found here.

Companion R and Python packages are described in Cattaneo, Feng, Palomba and Titiunik (2022).

Companion commands are: scdata for data preparation in the single treated unit case, scest for point estimation, scpi for inference procedures, and scplot for SC plots.

Related Stata, R, and Python packages useful for inference in SC designs are described in the following website:

https://nppackages.github.io/scpi/

For an introduction to synthetic control methods, see Abadie (2021) and references therein.

In case of unbalanced panel datasets, the preferred data structure should be a balanced panel with missing values. See tsfill, full for a useful command to create balanced structures.

## Options

### Variables

**id(***idvar***)** specifies the variable containing the identifier for each unit.

**time(***timevar***)** specifies the variable containing the time period of each observation.

**outcome(***outcomevar***)** specifies the outcome variable of interest. Note that *outcomevar* may not be among the *features* specified.

**treatment(***treatmentvar***)** specifies the treatment indicator.

### Estimator

**covadj(***string***)** specifies the variable to be used for adjustment for each features for each treated unit. If the user wants to specify the same set of covariates for all features, a string should be provided according to the following format: **covadj("***cov1, cov2***")**. If instead a different set of covariates per feature has to be specified, then the following format should be used **covadj("***cov1, cov2; cov1, cov3***")**. Note that in this latter case the number of sub-lists delimited by ";" must be equal to the number of *features*. Moreover, the order of the sub-lists matters, in the sense that the first sub-list is interpreted as the set of covariates used for adjustment for the first feature, and so on. Finally, the user can specify 'constant' and 'trend' as covariates even if they are not present in the loaded dataset, the former includes a constant, whilst the latter a linear deterministic trend. See Details section for more.

**cointegrated(***string***)** a logical value (the input should be either True or False) that specifies the presence of a cointegrating relationship between the features of the treated unit(s) and the the features of the donors. Default is **cointegrated("False")**. It can be specified for each treated unit. See Details section for more.

**constant(***string***)** a logical value (the input should be either True or False) that includes a common constant term across features. Default is **constant("False"}**. It can be specified for each treated unit. See Details section for more.

**anticipation(***string***)** specifies the number of periods of potential anticipation effects. Default is no anticipation. Note that it has to be a string, e.g. **anticipation("1")**. It can be specified for each treated unit. See Details section for more.

**effect(***string***)** a string indicating the type of treatment effect to be estimated. Options are: 'unit-time', which estimates treatment effects for each treated unit-time combination; 'unit', which estimates the treatment effect for each unit by averaging post-treatment features over time; 'time', which estimates the average treatment effect on the treated at various horizons.

**post_est(***string***)** a string specifying the number of post-treatment periods for which treatment effects have to be estimated for each treated unit. It is only effective when effect = "unit-time". Note that it has to be a string, e.g. **post_est("1")**.

**units_est(***string***)** a string specifying the treated units for which treatment effects have to be estimated. Treated units must be separated by commas, e.g. **units_est("unit1, unit2, unit3")**.

**donors_est(***string***)** a string specifying the donors units to be used. Note that all treated units share the same potential donors. If this is not desired, the donor pool can be separately specified for each treated unit. See Details section for more.

### Others

**dfname(***string***)** specifies the name of the Python object that is saved and that will be passed to scest or scpi.

**pypinocheck)** if specified avoids to check that the version of scpi_pkg in Python is the one required by **scdata** in Stata. When not specified performs the check and stores a macro called to avoid checking it multiple times.{p_end

## Details

This section describes how to use **scdatamulti** in two cases: first, when the user wants a common specification across treated units; second, when the user wants to tailor her specification for each treated unit.

### Common Specification

Let's start first with the simple case of common specification across treated units. Suppose, for the sake of the example, that there are just two treated units and two features to be matched on. The command would simply be

```
scdatamulti feature1 feature2, id(idvar) outcome(feature1) treatment(trvar) time(timevar)
```

If covariate adjustment, cointegration, anticipation effects, and a global constant need to be specified for each treated unit, then

```
scdatamulti feature1 feature2, id(idvar) outcome(feature1) treatment(trvar) time(timevar) ///
constant(True) cointegrated(True) anticipation(1) covadj("constant, trend")
```

### Heterogeneous Specification

Again, suppose there are two treated units and an individual specification is desired. In particular, we would like to match one feature of unit one and two features of the second unit. Then

```
scdatamulti (unit1: feature1) (unit2: feature1 feature2), id(idvar) outcome(feature1) treatment(trvar) time(timevar) ///
constant($cons_spec) cointegrated($coint_spec) anticipation($ant_spec) covadj($cov_spec)
```

Where the globals are defined as follows

First, we specify covariate adjustment just for the first feature of both treated units adding a linear trend for the first unit and a constant term for the second unit.

```
global cov_spec = "(unit1: trend) (unit2: constant; None)"
```

Second, we add a global constant for both treated units. There are two equivalent ways to do it:

```
global cons_spec = "True"
global cons_spec = "(unit1: True) (unit2: True)"
```

Similarly,

```
global coint_spec = "(unit1: True) (unit2: True)"
global ant_spec = "(unit1: 0) (unit2: 1)"
global donors_spec = "(unit1: donor1 donor2) (unit2: donor2 donor3)"
```

## Example: Germany Data

Setup

```
. use scpi_germany.dta
```

Prepare data

```
. scdata gdp, dfname("python_scdata") id(country) outcome(gdp) time(year) treatment(status) cointegrated
```

## Stored results

**scdata** stores the following in **e()**:

Scalars

| | |
|---|---|
| **e(I)** | number of treated units |
| **e(KMI)** | total number of covariates used for adjustment |

Macros

| | |
|---|---|
| **e(features)** | name of features |
| **e(outcomevar)** | name of outcome variable |
| **e(constant)** | logical indicating the presence of a common constant across features |
| **e(cointegrated)** | logical indicating cointegration |

Matrices

| | |
|---|---|
| **e(A)** | pre-treatment features of the treated unit |
| **e(B)** | pre-treatment features of the control units |
| **e(C)** | covariates used for adjustment |
| **e(P)** | predictor matrix |
| **e(J)** | number of donors for each treated unit |
| **e(KM)** | total number of covariates used for adjustment for each treated unit |

## References

Abadie, A. 2021. Using synthetic controls: Feasibility, data requirements, and methodological aspects. *Journal of Economic Literature*, 59(2), 391-425.

Cattaneo, M. D., Feng, Y., and Titiunik, R. 2021. Prediction intervals for synthetic control methods. *Journal of the American Statistical Association*, 116(536), 1865-1880.

Cattaneo, M. D., Feng, Y., Palomba F., and Titiunik, R. 2022. scpi: Uncertainty Quantification for Synthetic Control Estimators, *arXiv*:2202.05984.

Cattaneo, M. D., Feng, Y., Palomba F., and Titiunik, R. 2023. Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption, *arXiv*:2210.05026.

## Authors

Matias D. Cattaneo, Princeton University, Princeton, NJ. cattaneo@princeton.edu.

Yingjie Feng, Tsinghua University, Beijing, China. fengyj@sem.tsinghua.edu.cn.

Filippo Palomba, Princeton University, Princeton, NJ. fpalomba@princeton.edu.

Rocio Titiunik, Princeton University, Princeton, NJ.