Package 'scpi'

June 10, 2026

Title Synthetic Control Methods

Version 4.0.1

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

BugReports https://github.com/nppackages/scpi/issues

Description Implementation of prediction and inference procedures for Synthetic Control methods using least square, lasso, ridge, or simplex-type constraints. Uncertainty is quantified with prediction intervals as developed in Cattaneo, Feng, and Titiunik (2021) <doi:10.1080/01621459.2021.1979561> for a single treated unit and in Cattaneo, Feng, Palomba, and Titiunik (2027) <doi:10.1162/rest_a_01588> for multiple treated units and staggered adoption. More details about the software implementation can be found in Cattaneo, Feng, Palomba, and Titiunik (2025) <doi:10.18637/jss.v113.i01>.

Depends R (>= 4.1.0)

Imports abind (>= 1.4.5), CVXR (> 1.9), doSNOW (>= 1.0.19), dplyr (>= 1.0.7), ECOSolveR (>= 0.5.4), fastDummies (>= 1.6.3), foreach (>= 1.5.1), ggplot2 (>= 3.3.3), magrittr (>= 2.0.1), MASS (>= 7.3), Matrix (>= 1.3.3), methods (>= 4.1.0), parallel (>= 4.1.0), purrr (>= 0.3.4), Qtools (>= 1.5.6), reshape2 (>= 1.4.4), Rdpack (>= 2.4), rlang (>= 0.4.11), stats (>= 4.1.0), stringr (>= 1.4.0), tibble (>= 3.1.2), tidyr (>= 1.1.3), utils (>= 4.1.1)

Suggests testthat (>= 3.0.0)

LazyData true

NeedsCompilation no

RdMacros Rdpack

License GPL-3

Encoding UTF-8

RoxygenNote 7.3.2

Config/testthat/edition 3

Author Matias D. Cattaneo [aut, cre],
Yingjie Feng [aut],
Filippo Palomba [aut],
Rocio Titiunik [aut]

Maintainer Matias D. Cattaneo <matias.d.cattaneo@gmail.com>

Repository CRAN

Date/Publication 2026-06-10 09:40:02 UTC

Contents

`scpi-package` `scpi`*: A Package to Compute Synthetic Control Prediction Intervals With Multiple Treated Units and Staggered Adoption*

Description

The package implements estimation, inference procedures, and produces plots for Synthetic Control (SC) methods using least squares, lasso, ridge, or simplex-type constraints. Uncertainty is quantified using prediction intervals according to Cattaneo et al. (2021) and Cattaneo et al. (2027).

Included functions are: scdata and scdataMulti for data preparation, scest for point estimation, scpi for inference procedures, and scplot and scplotMulti for plots.

`print()` and `summary()` methods are available for `scest` and `scpi`.

Companion Stata and Python packages are described in Cattaneo et al. (2025).

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.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

Useful links:

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

`coef.scest` *Coef Method for Synthetic Control Methods*

Description

The coef method for synthetic control prediction fitted objects.

Usage

```
## S3 method for class 'scest'
coef(object, ...)
```

Arguments

`object`	Class "scest" object, obtained by calling `scest`.
`...`	Other arguments (eg. `ncols`).

Value

No return value, called to show scest constructed weights.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.
Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.
Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.
Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scest for synthetic control prediction.
Supported methods: print.scest, summary.scest, coef.scest.

---

coef.scpi *Coef Method for Synthetic Control Methods*

Description

The coef method for synthetic control prediction fitted objects.

Usage

```
## S3 method for class 'scpi'
coef(object, ...)
```

Arguments

object	Class "scpi" object, obtained by calling scpi.
...	Other arguments (eg. ncols).

Value

No return value, called to show scpi constructed weights.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.
Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.
Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.
Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scpi for synthetic control prediction.
Supported methods: print.scpi, summary.scpi, coef.scpi.

print.scdata *Summary Method for Synthetic Control*

Description

The print method for synthetic control data objects.

Usage

```
## S3 method for class 'scdata'
print(x, ...)
```

Arguments

x	Class "scdata" object, obtained by calling scdata.
...	Other arguments.

Value

No return value, called to print scdata results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scdata for synthetic control data preparation.

Supported methods: print.scdata, summary.scdata.

print.scdataMulti *Summary Method for Synthetic Control*

Description

The print method for synthetic control data objects.

Usage

```
## S3 method for class 'scdataMulti'
print(x, ...)
```

Arguments

x	Class "scdataMulti" object, obtained by calling scdataMulti.
...	Other arguments.

Value

No return value, called to print scdataMulti results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scdataMulti for synthetic control data preparation.

Supported methods: print.scdataMulti, summary.scdataMulti.

---

print.scest *Print Method for Synthetic Control Methods*

Description

The print method for synthetic control prediction fitted objects.

Usage

```
## S3 method for class 'scest'
print(x, ...)
```

Arguments

x	Class "scest" object, obtained by calling scest.
...	Other arguments.

Value

No return value, called to print scest results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scest for synthetic control prediction.

Supported methods: print.scest, summary.scest, coef.scest.

---

print.scpi *Print Method for Synthetic Control Inference*

Description

The print method for synthetic control inference objects.

Usage

```
## S3 method for class 'scpi'
print(x, ...)
```

Arguments

x	Class "scpi" object, obtained by calling scpi.
...	Other arguments.

Value

No return value, called to print scpi results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scpi for synthetic control inference

Supported methods: print.scpi, summary.scpi.

scdata *Data Preparation for* scest *or* scpi *for Point Estimation and Inference Procedures Using Synthetic Control Methods.*

Description

The command prepares the data to be used by scest or scpi to implement estimation and inference procedures for Synthetic Control (SC) methods. It allows the user to specify the outcome variable, the features of the treated unit to be matched, and covariate-adjustment feature by feature. The names of the output matrices follow the terminology proposed in Cattaneo et al. (2021) and Cattaneo et al. (2027).

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdataMulti for data preparation in the multiple treated units case with staggered adoption, scest for point estimation, scpi for inference procedures, scplot and scplotMulti for plots in the single and multiple treated unit(s) cases, respectively.

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.

Usage

```
scdata(
  df,
  id.var,
  time.var,
  outcome.var,
  period.pre,
  period.post,
  unit.tr,
  unit.co,
  features = NULL,
  cov.adj = NULL,
  cointegrated.data = FALSE,
  anticipation = 0,
  constant = FALSE,
  verbose = TRUE
)
```

Arguments

df	a dataframe object.
id.var	a character or numeric scalar with the name of the variable containing units' IDs. The ID variable can be numeric or character.

`time.var` a character with the name of the time variable. The time variable has to be numeric, integer, or Date. In case `time.var` is Date it should be the output of `as.Date()` function. An integer or numeric time variable is suggested when working with yearly data, whereas for all other formats a Date type time variable is preferred.

`outcome.var` a character with the name of the outcome variable. The outcome variable has to be numeric.

`period.pre` a numeric vector that identifies the pre-treatment period in time.var.

`period.post` a numeric vector that identifies the post-treatment period in time.var.

`unit.tr` a character or numeric scalar that identifies the treated unit in `id.var`.

`unit.co` a character or numeric vector that identifies the donor pool in `id.var`.

`features` a character vector containing the name of the feature variables used for estimation. If this option is not specified the default is `features = outcome.var`.

`cov.adj` a list specifying the names of the covariates to be used for adjustment for each feature. If `outcome.var` is not in the variables specified in `features`, we force `cov.adj<-NULL`. See the **Details** section for more.

`cointegrated.data` a logical that indicates if there is a belief that the data is cointegrated or not. The default value is `FALSE`. See the **Details** section for more.

`anticipation` a scalar that indicates the number of periods of potential anticipation effects. Default is 0.

`constant` a logical which controls the inclusion of a constant term across features. The default value is `FALSE`.

`verbose` if `TRUE` prints additional information in the console.

Details

- `cov.adj` can be used in two ways. First, if only one feature is specified through the option `features`, `cov.adj` has to be a list with one (even unnamed) element (eg. `cov.adj = list(c("constant","trend"))`). Alternatively, if multiple features are specified, then the user has two possibilities:
  - provide a list with one element, then the same covariates are used for adjustment for each feature. For example, if there are two features specified and the user inputs `cov.adj = list(c("constant","trend"))`, then a constant term and a linear trend are for adjustment for both features.
  - provide a list with as many elements as the number of features specified, then feature-specific covariate adjustment is implemented. For example, `cov.adj = list('f1' = c("constant","trend"), 'f2' = c("trend"))`. In this case the name of each element of the list should be one (and only one) of the features specified. Note that if two (or more) features are specified and covariates adjustment has to be specified just for one of them, the user must still provide a list of the same length of the number of features, e.g., `cov.adj = list('f1' = c("constant","trend"), 'f2' = NULL`.

  This option allows the user to include feature-specific constant terms or time trends by simply including "constant" or "trend" in the corresponding element of the list.

When `outcome.var` is not included in `features`, we automatically set $\mathcal{R} = \emptyset$, that is we do not perform covariate adjustment. This is because, in this setting it is natural to create the out-of-sample prediction matrix $\mathbf{P}$ using the post-treatment outcomes of the donor units only.

- `cointegrated.data` allows the user to model the belief that $\mathbf{A}$ and $\mathbf{B}$ form a cointegrated system. In practice, this implies that when dealing with the pseudo-true residuals $\mathbf{u}$, the first-difference of $\mathbf{B}$ are used rather than the levels.

Value

The command returns an object of class 'scdata' containing the following

`A`	a matrix containing pre-treatment features of the treated unit.
`B`	a matrix containing pre-treatment features of the control units.
`C`	a matrix containing covariates for adjustment.
`P`	a matrix whose rows are the vectors used to predict the out-of-sample series for the synthetic unit.
`Y.pre`	a matrix containing the pre-treatment outcome of the treated unit.
`Y.post`	a matrix containing the post-treatment outcome of the treated unit.
`Y.donors`	a matrix containing the pre-treatment outcome of the control units.
`specs`	a list containing some specifics of the data:

- `J`, the number of control units
- `K`, a numeric vector with the number of covariates used for adjustment for each feature
- `KM`, the total number of covariates used for adjustment
- `M`, number of features
- `period.pre`, a numeric vector with the pre-treatment period
- `period.post`, a numeric vector with the post-treatment period
- `T0.features`, a numeric vector with the number of periods used in estimation for each feature
- `T1.outcome`, the number of post-treatment periods
- `outcome.var`, a character with the name of the outcome variable
- `features`, a character vector with the name of the features
- `constant`, for internal use only
- `out.in.features`, for internal use only
- `effect`, for internal use only
- `sparse.matrices`, for internal use only
- `treated.units`, list containing the IDs of all treated units

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

scdataMulti, scest, scpi, scplot, scplotMulti

Examples

```
data <- scpi_germany

df <- scdata(df = data, id.var = "country", time.var = "year",
             outcome.var = "gdp", period.pre = (1960:1990),
             period.post = (1991:2003), unit.tr = "West Germany",
             unit.co = setdiff(unique(data$country), "West Germany"),
             constant = TRUE, cointegrated.data = TRUE)
```

---

scdataMulti *Data Preparation for* scest *or* scpi *for Point Estimation and Inference Procedures Using Synthetic Control Methods.*

Description

The command 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 is a generalization of scdata, since this latter prepares the data in the particular case of a single treated unit.

The names of the output matrices follow the terminology proposed in Cattaneo et al. (2021) and Cattaneo et al. (2027).

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdataMulti for data preparation in the multiple treated units case with staggered adoption, scest for point estimation, scpi for inference procedures, scplot and scplotMulti for plots in the single and multiple treated unit(s) cases, respectively.

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.

Variable Naming Convention: due to how `scpi` handles object internally, we kindly ask the users of the `R` version of the package to avoid including dots in the variable names. For example, "y.var" would generate issues with some parts of the code, whereas "yvar" or "y_var" would not.

Usage

```
scdataMulti(
  df,
  id.var,
  time.var,
  outcome.var,
  treatment.var,
  features = NULL,
  cov.adj = NULL,
  cointegrated.data = FALSE,
  post.est = NULL,
  units.est = NULL,
  donors.est = NULL,
  anticipation = 0,
  effect = "unit-time",
  constant = FALSE,
  verbose = TRUE,
  sparse.matrices = FALSE
)
```

Arguments

`df`	a dataframe object.
`id.var`	a character with the name of the variable containing units' IDs. The ID variable can be numeric or character.
`time.var`	a character with the name of the time variable. The time variable has to be numeric, integer, or Date. In case `time.var` is Date it should be the output of `as.Date()` function. An integer or numeric time variable is suggested when working with yearly data, whereas for all other formats a Date type time variable is preferred.
`outcome.var`	a character with the name of the outcome variable. The outcome variable has to be numeric.
`treatment.var`	a character with the name of the variable containing the treatment assignment of each unit. The referenced variable has to take value 1 if the unit is treated in that period and value 0 otherwise. Please notice that, as common in the SC literature, we presume that once a unit is treated it remains treated forever. If treatment.var does not comply with this requirement the command would not work as expected!

`features` a list containing the names of the feature variables used for estimation. If this option is not specified the default is `features = outcome.var`. For multiple treated units, this can be a one-element list shared by all treated units or a named list with one element for each selected treated unit.

`cov.adj` a list specifying the names of the covariates to be used for adjustment for each feature. For multiple treated units, this can be a shared `scdata`-style list or a named list with one element for each selected treated unit, where each element is a `scdata`-style list. Unnamed feature-specific entries are matched to that unit's features in order. If `outcome.var` is not in the variables specified in `features`, we force `cov.adj<-NULL`. See the **Details** section for more.

`cointegrated.data` a logical that indicates if there is a belief that the data is cointegrated or not, or a named list of logicals with one element for each selected treated unit. The default value is `FALSE`.

`post.est` a scalar specifying the number of post-treatment periods or a list specifying the periods for which treatment effects have to be computed for each treated unit. It is only effective when effect = "unit-time".

`units.est` a list specifying the treated units for which treatment effects have to be computed.

`donors.est` a list specifying the donors units to be used. If the list has length 1, then all treated units share the same potential donors. Otherwise, if the user requires different donor pools for different treated units, the list must be of the same length of the number of treated units and each element has to be named with one treated unit's name as specified in id.var.

`anticipation` a scalar that indicates the number of periods of potential anticipation effects. Default is 0.

`effect` a string indicating the type of treatment effect to be computed. Options are: 'unit-time', which estimates treatment effects for each treated unit- post treatment period 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.

`constant` a logical which controls the inclusion of a constant term across features, or a named list of logicals with one element for each selected treated unit. The default value is `FALSE`.

`verbose` if `TRUE` prints additional information in the console.

`sparse.matrices` if `TRUE` all block diagonal matrices ($\mathbf{B}$, $\mathbf{C}$, and $\mathbf{P}$) are sparse matrices. This is suggested if the dimension of the dataset is large as it will likely reduce the execution time. The sparse matrices will be objects of class 'dgCMatrix' or 'lgCMatrix', thus to visualize them they need to be transformed in matrices, e.g. `View(as.matrix(B))`.

Details

- **Covariate-adjustment.** See the **Details** section in `scdata` for further information on how to specify covariate-adjustment feature-by-feature.

- **Cointegration.** `cointegrated.data` allows the user to model the belief that $\mathbf{A}$ and $\mathbf{B}$ form a cointegrated system. In practice, this implies that when dealing with the pseudo-true residuals $\mathbf{u}$, the first-difference of $\mathbf{B}$ are used rather than the levels.
- **Effect.** `effect` allows the user to select between two causal quantities. The default option, `effect = "unit-time"`, prepares the data for estimation of

$$\tau_{ik}, \quad k \geq, i = 1, \ldots, N_1,$$

whereas the option `effect = "unit"` prepares the data for estimation of

$$\tau_{\cdot k} = \frac{1}{N_1} \sum_{i=1}^{N_1} \tau_{ik}$$

which is the average effect on the treated unit across multiple post-treatment periods.

Value

The command returns an object of class 'scdataMulti' containing the following

`A` a matrix containing pre-treatment features of the treated units.

`B` a matrix containing pre-treatment features of the control units.

`C` a matrix containing covariates for adjustment.

`P` a matrix whose rows are the vectors used to predict the out-of-sample series for the synthetic units.

`P.diff` for internal use only.

`Y.df` a dataframe containing the outcome variable for all units.

`Y.pre` a matrix containing the pre-treatment outcome of the treated units.

`Y.post` a matrix containing the post-treatment outcome of the treated units.

`Y.donors` a matrix containing the pre-treatment outcome of the control units.

`specs` a list containing some specifics of the data:

- `J`, a list containing the number of donors for each treated unit
- `K`, a list containing the number of covariates used for adjustment for each feature for each treated unit
- `KM`, a list containing the total number of covariates used for adjustment for each treated unit
- `M`, a list containing number of features used for each treated unit
- `I`, number of treated units
- `KMI`, overall number of covariates used for adjustment
- `period.pre`, a list containing a numeric vector with the pre-treatment period for each treated unit
- `period.post`, a list containing a numeric vector with the post-treatment period for each treated unit
- `T0.features`, a list containing a numeric vector with the number of periods used in estimation for each feature for each treated unit

- `T1.outcome`, a list containing the number of post-treatment periods for each treated unit
- `features.list`, a list containing the name of the features for each treated unit
- `outcome.var`, a character containing the name of the outcome variable
- `constant`, for internal use only
- `effect`, for internal use only
- `anticipation`, number of periods of potential anticipation effects
- `out.in.features`, for internal use only
- `sparse.matrices`, for internal use only
- `treated.units`, list containing the IDs of all treated units
- `donors.list`, list containing the IDs of the donors of each treated unit

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

`scdata`, `scest`, `scpi`, `scplot`, `scplotMulti`

Examples

```
datager <- scpi_germany

datager$tr_id <- 0
datager$tr_id[(datager$country == "West Germany" & datager$year > 1990)] <- 1
```

```
datager$tr_id[(datager$country == "Italy" & datager$year > 1992)] <- 0

outcome.var <- "gdp"
id.var <- "country"
treatment.var <- "tr_id"
time.var <- "year"
df.unit <- scdataMulti(datager, id.var = id.var, outcome.var = outcome.var,
                       treatment.var = treatment.var,
                       time.var = time.var, features = list(c("gdp", "trade")),
                     cointegrated.data = TRUE, constant = TRUE)
```

`scest` *Prediction of Synthetic Control*

Description

The command implements estimation procedures for Synthetic Control (SC) methods using least squares, lasso, ridge, or simplex-type constraints. For more information see Cattaneo et al. (2021) and Cattaneo et al. (2027).

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdata and scdataMulti for data preparation in the single and multiple treated unit(s) cases, respectively, scpi for inference procedures, scplot and scplotMulti for plots in the single and multiple treated unit(s) cases, respectively.

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.

Usage

```
scest(
  data,
  w.constr = NULL,
  V = "separate",
  V.mat = NULL,
  solver = "CLARABEL",
  plot = FALSE,
  plot.name = NULL,
  plot.path = NULL,
  save.data = NULL
)
```

Arguments

data	a class 'scdata' object, obtained by calling scdata, or class 'scdataMulti' obtained via scdataMulti.
w.constr	a list specifying the constraint set the estimated weights of the donors must belong to. w.constr can contain up to four objects: • 'p', a string indicating the norm to be constrained (p should be one of "no norm", "L1", and "L2") • 'dir', a string indicating whether the constraint on the norm is an equality ("==") or inequality ("<=") • 'Q', a scalar defining the value of the constraint on the norm • 'lb', a scalar defining the lower bound on the weights. It can be either 0 or -Inf. • 'name', a character selecting one of the default proposals. See the **Details** section for more.
V	specifies the type of weighting matrix to be used when minimizing the sum of squared residuals $$(\mathbf{A} - \mathbf{Bw} - \mathbf{Cr})'\mathbf{V}(\mathbf{A} - \mathbf{Bw} - \mathbf{Cr})$$ The default is the identity matrix, so equal weight is given to all observations. In the case of multiple treated observations (you used scdataMulti to prepare the data), the user can specify V as a string equal to either "separate" or "pooled". If scdata() was used to prepare the data, V is automatically set to "separate" as the two options are equivalent. See the **Details** section for more.
V.mat	A conformable weighting matrix $\mathbf{V}$ to be used in the minimization of the sum of squared residuals $$(\mathbf{A} - \mathbf{Bw} - \mathbf{Cr})'\mathbf{V}(\mathbf{A} - \mathbf{Bw} - \mathbf{Cr}).$$ See the **Details** section for more information on how to prepare this matrix.
solver	a string containing the name of the solver used by CVXR when computing the weights. You can check which solvers are available on your machine by running CVXR::installed_solvers(). More information on what different solvers do can be found at the following link https://cvxr.rbind.io/cvxr_examples/cvxr_using-other-solvers/. "CLARABEL" is the default in all cases.
plot	a logical specifying whether scplot should be called and a plot saved in the current working directory. For more options see scplot.
plot.name	a string containing the name of the plot (the format is by default .png). For more options see scplot.
plot.path	a string containing the path at which the plot should be saved (default is output of getwd().)
save.data	a character specifying the name and the path of the saved dataframe containing the processed data used to produce the plot.

Details

Information is provided for the simple case in which $N_1 = 1$ if not specified otherwise.

- **Estimation of Weights.** `w.constr` specifies the constraint set on the weights. First, the element `p` allows the user to choose between imposing a constraint on either the L1 (`p = "L1"`) or the L2 (`p = "L2"`) norm of the weights and imposing no constraint on the norm (`p = "no norm"`). Second, `Q` specifies the value of the constraint on the norm of the weights. Third, `lb` sets the lower bound of each component of the vector of weights. Fourth, `dir` sets the direction of the constraint on the norm in case `p = "L1"` or `p = "L2"`. If `dir = "=="`, then

$$||\mathbf{w}||_p = Q, \quad w_j \geq lb, \ \ j = 1, \ldots, J$$

  If instead `dir = "<="`, then

$$||\mathbf{w}||_p \leq Q, \quad w_j \geq lb, \ \ j = 1, \ldots, J$$

  If instead `dir = "NULL"` no constraint on the norm of the weights is imposed.

  An alternative to specifying an ad-hoc constraint set on the weights would be choosing among some popular types of constraints. This can be done by including the element 'name' in the list `w.constr`. The following are available options:

  - If `name == "simplex"` (the default), then

$$||\mathbf{w}||_1 = 1, \quad w_j \geq 0, \ \ j = 1, \ldots, J.$$

  - If `name == "lasso"`, then

$$||\mathbf{w}||_1 \leq Q,$$

    where `Q` is by default equal to 1 but it can be provided as an element of the list (eg. `w.constr = list(name = "lasso", Q = 2)`).
  - If `name == "ridge"`, then

$$||\mathbf{w}||_2 \leq Q,$$

    where `Q` is a tuning parameter that is by default computed as

$$(J + KM)\widehat{\sigma}_u^2 / ||\widehat{\mathbf{w}}_{OLS}||_2^2$$

    where J is the number of donors and KM is the total number of covariates used for adjustment. The user can provide `Q` as an element of the list (eg. `w.constr = list(name = "ridge", Q = 1)`).
  - If `name == "ols"`, then the problem is unconstrained and the vector of weights is estimated via ordinary least squares.
  - If `name == "L1-L2"`, then

$$||\mathbf{w}||_1 = 1, \quad ||\mathbf{w}||_2 \leq Q, \quad w_j \geq 0, \ \ j = 1, \ldots, J.$$

    where Q is a tuning parameter computed as in the "ridge" case.

- **Weighting Matrix.**
  - if `V <- "separate"`, then $\mathbf{V} = \mathbf{I}$ and the minimized objective function is

$$\sum_{i=1}^{N_1} \sum_{l=1}^{M} \sum_{t=1}^{T_0} \left(a_{t,l}^i - \mathbf{b}_{t,l}^{i\prime} \mathbf{w}^i - \mathbf{c}_{t,l}^{i\prime} \mathbf{r}_l^i \right)^2,$$

    which optimizes the separate fit for each treated unit.

– if `V <- "pooled"`, then $\mathbf{V} = \frac{1}{I}\mathbf{1}\mathbf{1}' \otimes \mathbf{I}$ and the minimized objective function is

$$\sum_{l=1}^{M}\sum_{t=1}^{T_0}\left(\frac{1}{N_1^2}\sum_{i=1}^{N_1}\left(a_{t,l}^{i} - \mathbf{b}_{t,l}^{i\prime}\mathbf{w}^{i} - \mathbf{c}_{t,l}^{i\prime}\mathbf{r}_{l}^{i}\right)\right)^2,$$

which optimizes the pooled fit for the average of the treated units.

– if the user wants to provide their own weighting matrix, then it must use the option `V.mat` to input a $v \times v$ positive-definite matrix, where v is the number of rows of $\mathbf{B}$ (or $\mathbf{C}$) after potential missing values have been removed. In case the user wants to provide their own `V`, we suggest to check the appropriate dimension v by inspecting the output of either `scdata` or `scdataMulti` and check the dimensions of $\mathbf{B}$ (and $\mathbf{C}$). Note that the weighting matrix could cause problems to the optimizer if not properly scaled. For example, if $\mathbf{V}$ is diagonal we suggest to divide each of its entries by $\|\mathrm{diag}(\mathbf{V})\|_1$.

Value

The function returns an object of class 'scest' containing two lists. The first list is labeled 'data' and contains used data as returned by `scdata` and some other values.

`A`	a matrix containing pre-treatment features of the treated unit(s).
`B`	a matrix containing pre-treatment features of the control units.
`C`	a matrix containing covariates for adjustment.
`P`	a matrix whose rows are the vectors used to predict the out-of-sample series for the synthetic unit(s).
`P.diff`	for internal use only.
`Y.pre`	a matrix containing the (raw) pre-treatment outcome of the treated unit(s).
`Y.post`	a matrix containing the (raw) post-treatment outcome of the treated unit(s).
`Y.pre.agg`	a matrix containing the aggregate pre-treatment outcome of the treated unit(s). This differs from Y.pre only in the case 'effect' in `scdataMulti()` is set to either 'unit' or 'time'.
`Y.post.agg`	a matrix containing the aggregate post-treatment outcome of the treated unit(s). This differs from Y.post only in the case 'effect' in `scdataMulti()` is set to either 'unit' or 'time'.
`Y.donors`	a matrix containing the pre-treatment outcome of the control units.
`specs`	a list containing some specifics of the data: • `J`, the number of control units • `K`, a numeric vector with the number of covariates used for adjustment for each feature • `M`, number of features • `KM`, the total number of covariates used for adjustment • `KMI`, the total number of covariates used for adjustment • `I`, number of treated units • `period.pre`, a numeric vector with the pre-treatment period • `period.post`, a numeric vector with the post-treatment period

- `T0.features`, a numeric vector with the number of periods used in estimation for each feature
- `T1.outcome`, the number of post-treatment periods
- `constant`, for internal use only
- `effect`, for internal use only
- `anticipation`, number of periods of potential anticipation effects
- `out.in.features`, for internal use only
- `treated.units`, list containing the IDs of all treated units
- `donors.list`, list containing the IDs of the donors of each treated unit
- `class.type`, for internal use only

The second list is labeled 'est.results' and contains estimation results.

`w`	a matrix containing the estimated weights of the donors.
`r`	a matrix containing the values of the covariates used for adjustment.
`b`	a matrix containing $\mathbf{w}$ and $\mathbf{r}$.
`Y.pre.fit`	a matrix containing the estimated pre-treatment outcome of the SC unit(s).
`Y.post.fit`	a matrix containing the estimated post-treatment outcome of the SC unit(s).
`A.hat`	a matrix containing the predicted values of the features of the treated unit(s).
`res`	a matrix containing the residuals $\mathbf{A} - \widehat{\mathbf{A}}$.
`V`	a matrix containing the weighting matrix used in estimation.
`w.constr`	a list containing the specifics of the constraint set used on the weights.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

scdataMulti, scdata, scpi, scplot, scplotMulti

Examples

```
data <- scpi_germany

df <- scdata(df = data, id.var = "country", time.var = "year",
             outcome.var = "gdp", period.pre = (1960:1990),
             period.post = (1991:2003), unit.tr = "West Germany",
             unit.co = setdiff(unique(data$country), "West Germany"),
             constant = TRUE, cointegrated.data = TRUE)

result <- scest(df, w.constr = list(name = "simplex", Q = 1))
result <- scest(df, w.constr = list(lb = 0, dir = "==", p = "L1", Q = 1))
```

---

scpi *Prediction Intervals for Synthetic Control Methods*

Description

The command implements estimation and inference procedures for Synthetic Control (SC) methods using least squares, lasso, ridge, or simplex-type constraints. Uncertainty is quantified using prediction intervals according to Cattaneo et al. (2021) and Cattaneo et al. (2027). scpi returns the estimated post-treatment series for the synthetic unit through the command scest and quantifies in-sample and out-of-sample uncertainty to provide confidence intervals for each point estimate.

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdata and scdataMulti for data preparation in the single and multiple treated unit(s) cases, respectively, scest for point estimation, scplot and scplotMulti for plots in the single and multiple treated unit(s) cases, respectively.

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.

Usage

```
scpi(
  data,
  w.constr = NULL,
  V = "separate",
  V.mat = NULL,
  solver = "CLARABEL",
  P = NULL,
  u.missp = TRUE,
```

```
  u.sigma = "HC1",
  u.order = 1,
  u.lags = 0,
  u.design = NULL,
  u.alpha = 0.05,
  e.method = "all",
  e.order = 1,
  e.lags = 0,
  e.design = NULL,
  e.alpha = 0.05,
  sims = 200,
  rho = NULL,
  rho.max = 0.2,
  cores = 1,
  plot = FALSE,
  plot.name = NULL,
  w.bounds = NULL,
  e.bounds = NULL,
  force.joint.PI.optim = FALSE,
  save.data = NULL,
  verbose = TRUE
)
```

Arguments

`data` a class 'scdata' object, obtained by calling `scdata`, a class 'scdataMulti' obtained via `scdataMulti`, or a class 'scest' object obtained by calling `scest`. When a class 'scest' object is provided, `scpi` reuses its point estimates and skips the internal `scest` call.

`w.constr` a list specifying the constraint set the estimated weights of the donors must belong to. `w.constr` can contain up to five elements:

- '`p`', a scalar indicating the norm to be used (`p` should be one of "no norm", "L1", and "L2")
- '`dir`', a string indicating whether the constraint on the norm is an equality ("==") or inequality ("<=")
- '`Q`', a scalar defining the value of the constraint on the norm
- '`lb`', a scalar defining the lower bound on the weights. It can be either 0 or `-Inf`.
- '`name`', a character selecting one of the default proposals See the **Details** section for more.

`V` specifies the type of weighting matrix to be used when minimizing the sum of squared residuals

$$(\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r})'\mathbf{V}(\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r})$$

The default is the identity matrix, so equal weight is given to all observations. In the case of multiple treated observations (you used `scdataMulti` to prepare the data), the user can specify `V` as a string equal to either "separate" or "pooled". If

scdata() was used to prepare the data, V is automatically set to "separate" as the two options are equivalent. See the **Details** section for more.

V.mat — A conformable weighting matrix $\mathbf{V}$ to be used in the minimization of the sum of squared residuals

$$(\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r})'\mathbf{V}(\mathbf{A} - \mathbf{B}\mathbf{w} - \mathbf{C}\mathbf{r}).$$

See the **Details** section for more information on how to prepare this matrix.

solver — a string containing the name of the solver used by CVXR when computing the weights. You can check which solvers are available on your machine by running CVXR::installed_solvers(). More information on what different solvers do can be found at the following link https://cvxr.rbind.io/cvxr_examples/cvxr_using-other-solvers/. "CLARABEL" is the default in all cases.

P — a $I \cdot T_1 \times I \cdot (J + KM)$ matrix containing the design matrix to be used to obtain the predicted. post-intervention outcome of the synthetic control unit. T_1 is the number of post-treatment periods, J is the size of the donor pool, and K_1 is the number of covariates used for adjustment in the outcome equation.

u.missp — a logical indicating if misspecification should be taken into account when dealing with $\mathbf{u}$.

u.sigma — a string specifying the type of variance-covariance estimator to be used when estimating the conditional variance of $\mathbf{u}$.

u.order — a scalar that sets the order of the polynomial in $\mathbf{B}$ when predicting moments of $\mathbf{u}$. The default is u.order = 1, however if there is risk of over-fitting, the command automatically sets it to u.order = 0. See the **Details** section for more information.

u.lags — a scalar that sets the number of lags of $\mathbf{B}$ when predicting moments of $\mathbf{u}$. The default is u.lags = 0, however if there is risk of over-fitting, the command automatically sets it to u.lags = 0. See the **Details** section for more information.

u.design — a matrix with the same number of rows of $\mathbf{A}$ and $\mathbf{B}$ and whose columns specify the design matrix to be used when modeling the estimated pseudo-true residuals $\mathbf{u}$.

u.alpha — a scalar specifying the confidence level for in-sample uncertainty, i.e. 1 - u.alpha is the confidence level.

e.method — a string selecting the method to be used in quantifying out-of-sample uncertainty among: "gaussian" which uses conditional subgaussian bounds; "ls" which specifies a location-scale model for $\mathbf{u}$; "qreg" which employs a quantile regressions to get the conditional bounds; "all" uses each one of the previous methods.

e.order — a scalar that sets the order of the polynomial in $\mathbf{B}$ when predicting moments of $\mathbf{e}$. The default is e.order = 1, however if there is risk of over-fitting, the command automatically sets it to e.order = 0. See the **Details** section for more information.

e.lags — a scalar that sets the number of lags of $\mathbf{B}$ when predicting moments of $\mathbf{e}$. The default is e.order = 1, however if there is risk of over-fitting, the command automatically sets it to e.order = 0. See the **Details** section for more information.

`e.design` a matrix with the same number of rows of $\mathbf{A}$ and $\mathbf{B}$ and whose columns specify the design matrix to be used when modeling the estimated out-of-sample residuals $\mathbf{e}$.

`e.alpha` a scalar specifying the confidence level for out-of-sample uncertainty, i.e. 1 - `e.alpha` is the confidence level.

`sims` a scalar providing the number of simulations to be used in quantifying in-sample uncertainty.

`rho` a string specifying the formula used for the regularizing parameter that imposes sparsity on the estimated vector of weights. Users can provide a scalar with their own value for `rho`. Other options are described in the **Details** section.

`rho.max` a scalar indicating the maximum value attainable by the tuning parameter `rho`.

`cores` number of cores to be used by the command. The default is one. When the weighting matrix $\mathbf{V}$ is diagonal this option has no effect.

`plot` a logical specifying whether `scplot` should be called and a plot saved in the current working directory. For more options see `scplot`.

`plot.name` a string containing the name of the plot (the format is by default .png). For more options see `scplot`.

`w.bounds` a $N_1 \cdot T_1 \times 2$ matrix with the user-provided bounds on β. If `w.bounds` is provided, then the quantification of in-sample uncertainty is skipped. It is possible to provide only the lower bound or the upper bound by filling the other column with `NA`s.

`e.bounds` a $N_1 \cdot T_1 \times 2$ matrix with the user-provided bounds on $(\widehat{\mathbf{w}}, \widehat{\mathbf{r}})'$. If `e.bounds` is provided, then the quantification of out-of-sample uncertainty is skipped. It is possible to provide only the lower bound or the upper bound by filling the other column with `NA`s.

`force.joint.PI.optim`
this option is here mostly for backward-compatibility. If FALSE (the default) it solves a separate optimization problem for each treated unit when it comes to quantify in-sample uncertainty as long as the weighting matrix $\mathbf{V}$ is diagonal. If TRUE it solves a joint optimization problem for all treated units to quantify in-sample uncertainty. Both are valid approaches as we detail in the main paper (Cattaneo, Feng, Palomba, and Titiunik (2027)). The former is faster and less conservative.

`save.data` a character specifying the name and the path of the saved dataframe containing the processed data used to produce the plot.

`verbose` if `TRUE` prints additional information in the console.

Details

Information is provided for the simple case in which $N_1 = 1$ if not specified otherwise.

- **Estimation of Weights.** `w.constr` specifies the constraint set on the weights. First, the element `p` allows the user to choose between imposing a constraint on either the L1 (`p = "L1"`) or the L2 (`p = "L2"`) norm of the weights and imposing no constraint on the norm (`p = "no norm"`). Second, `Q` specifies the value of the constraint on the norm of the weights. Third,

`lb` sets the lower bound of each component of the vector of weights. Fourth, `dir` sets the direction of the constraint on the norm in case `p = "L1"` or `p = "L2"`. If `dir = "=="`, then

$$||\mathbf{w}||_p = Q, \quad w_j \geq lb, \ \ j = 1, \ldots, J$$

If instead `dir = "<="`, then

$$||\mathbf{w}||_p \leq Q, \quad w_j \geq lb, \ \ j = 1, \ldots, J$$

If instead `dir = "NULL"` no constraint on the norm of the weights is imposed.

An alternative to specifying an ad-hoc constraint set on the weights would be choosing among some popular types of constraints. This can be done by including the element 'name' in the list `w.constr`. The following are available options:

- If `name == "simplex"` (the default), then

$$||\mathbf{w}||_1 = 1, \quad w_j \geq 0, \ \ j = 1, \ldots, J.$$

- If `name == "lasso"`, then

$$||\mathbf{w}||_1 \leq Q,$$

  where `Q` is by default equal to 1 but it can be provided as an element of the list (eg. `w.constr = list(name = "lasso", Q = 2)`).
- If `name == "ridge"`, then

$$||\mathbf{w}||_2 \leq Q,$$

  where Q is a tuning parameter that is by default computed as

$$(J + KM)\widehat{\sigma}_u^2 / ||\widehat{\mathbf{w}}_{OLS}||_2^2$$

  where J is the number of donors and KM is the total number of covariates used for adjustment. The user can provide `Q` as an element of the list (eg. `w.constr = list(name = "ridge", Q = 1)`).
- If `name == "ols"`, then the problem is unconstrained and the vector of weights is estimated via ordinary least squares.
- If `name == "L1-L2"`, then

$$||\mathbf{w}||_1 = 1, \quad ||\mathbf{w}||_2 \leq Q, \quad w_j \geq 0, \ \ j = 1, \ldots, J.$$

  where Q is a tuning parameter computed as in the "ridge" case.

- **Weighting Matrix.**
  - if `V <- "separate"`, then $\mathbf{V} = \mathbf{I}$ and the minimized objective function is

$$\sum_{i=1}^{N_1} \sum_{l=1}^{M} \sum_{t=1}^{T_0} \left(a_{t,l}^i - \mathbf{b}_{t,l}^{i\prime} \mathbf{w}^i - \mathbf{c}_{t,l}^{i\prime} \mathbf{r}_l^i \right)^2,$$

    which optimizes the separate fit for each treated unit.
  - if `V <- "pooled"`, then $\mathbf{V} = \mathbf{1}\mathbf{1}' \otimes \mathbf{I}$ and the minimized objective function is

$$\sum_{l=1}^{M} \sum_{t=1}^{T_0} \left(\frac{1}{N_1^2} \sum_{i=1}^{N_1} \left(a_{t,l}^i - \mathbf{b}_{t,l}^{i\prime} \mathbf{w}^i - \mathbf{c}_{t,l}^{i\prime} \mathbf{r}_l^i \right) \right)^2,$$

    which optimizes the pooled fit for the average of the treated units.

  - if the user wants to provide their own weighting matrix, then it must use the option `V.mat` to input a $v \times v$ positive-definite matrix, where v is the number of rows of $\mathbf{B}$ (or $\mathbf{C}$) after potential missing values have been removed. In case the user wants to provide their own `V`, we suggest to check the appropriate dimension v by inspecting the output of either `scdata` or `scdataMulti` and check the dimensions of $\mathbf{B}$ (and $\mathbf{C}$). Note that the weighting matrix could cause problems to the optimizer if not properly scaled. For example, if $\mathbf{V}$ is diagonal we suggest to divide each of its entries by $\|\mathrm{diag}(\mathbf{V})\|_1$.

- **Regularization.** `rho` is estimated through the formula

$$\varrho = \sqrt{d_0 \log(d) \log(T_0)} \mathcal{C} T_0^{-1/2}$$

where d is the dimension of $\widehat{\boldsymbol{\beta}}$ and d_0 denote the number of nonzeros in $\widehat{\boldsymbol{\beta}}$ $\mathcal{C} = \widehat{\sigma}_u / \min_j \widehat{\sigma}_{b_j}$ if `rho = 'type-1'` and $\mathcal{C} = \max_j \widehat{\sigma}_{b_j} \widehat{\sigma}_u / \min_j \widehat{\sigma}^2_{b_j}$ if `rho = 'type-2'`, `rho = 'type-2'` is the default option from version 3.0.0 onwards, while previously 'type-1' was the default option. `rho` defines a new sparse weight vector as

$$\widehat{w}^\star_j = \mathbf{1}(\widehat{w}_j \geq \varrho)$$

- **In-sample uncertainty.** To quantify in-sample uncertainty it is necessary to model the pseudo-residuals $\mathbf{u}$. First of all, estimation of the first moment of $\mathbf{u}$ can be controlled through the option `u.missp`. When `u.missp = FALSE`, then $\mathbf{E}[u \mid \mathbf{D}_u] = 0$. If instead `u.missp = TRUE`, then $\mathbf{E}[\mathbf{u} \mid \mathbf{D}_u]$ is estimated using a linear regression of $\widehat{\mathbf{u}}$ on $\mathbf{D}_u$. The default set of variables in $\mathbf{D}_u$ is composed of $\mathbf{B}$, $\mathbf{C}$ and, if required, it is augmented with lags (`u.lags`) and polynomials (`u.order`) of $\mathbf{B}$. The option `u.design` allows the user to provide an ad-hoc set of variables to form $\mathbf{D}_u$. Regarding the second moment of $\mathbf{u}$, different estimators can be chosen: HC0, HC1, HC2, HC3, and HC4 using the option `u.sigma`.

- **Out-of-sample uncertainty.** To quantify out-of-sample uncertainty it is necessary to model the out-of-sample residuals $\mathbf{e}$ and estimate relevant moments. By default, the design matrix used during estimation $\mathbf{D}_e$ is composed of the blocks in $\mathbf{B}$ and $\mathbf{C}$ corresponding to the outcome variable. Moreover, if required by the user, $\mathbf{D}_e$ is augmented with lags (`e.lags`) and polynomials (`e.order`) of $\mathbf{B}$. The option `e.design` allows the user to provide an ad-hoc set of variables to form $\mathbf{D}_e$. Finally, the option `e.method` allows the user to select one of three estimation methods: "gaussian" relies on conditional sub-Gaussian bounds; "ls" estimates conditional bounds using a location-scale model; "qreg" uses conditional quantile regression of the residuals $\mathbf{e}$ on $\mathbf{D}_e$.

- **Residual Estimation Over-fitting.** To estimate conditional moments of $\mathbf{u}$ and e_t we rely on two design matrices, $\mathbf{D}_u$ and $\mathbf{D}_e$ (see above). Let d_u and d_e be the number of columns in $\mathbf{D}_u$ and $\mathbf{D}_e$, respectively. Assuming no missing values and balanced features, the number of observation used to estimate moments of $\mathbf{u}$ is $N_1 \cdot T_0 \cdot M$, whilst for moments of e_t is T_0. Our rule of thumb to avoid over-fitting is to check if $N_1 \cdot T_0 \cdot M \geq d_u + 10$ or $T_0 \geq d_e + 10$. If the former condition is not satisfied we automatically set `u.order = u.lags = 0`, if instead the latter is not met we automatically set `e.order = e.lags = 0`.

Value

The function returns an object of class 'scpi' containing three lists. The first list is labeled 'data' and contains used data as returned by `scdata` and some other values.

`A` a matrix containing pre-treatment features of the treated unit(s).

`B` a matrix containing pre-treatment features of the control units.

`C` a matrix containing covariates for adjustment.

`P` a matrix whose rows are the vectors used to predict the out-of-sample series for the synthetic unit(s).

`Y.pre` a matrix containing the pre-treatment outcome of the treated unit(s).

`Y.post` a matrix containing the post-treatment outcome of the treated unit(s).

`Y.pre.agg` a matrix containing the aggregate pre-treatment outcome of the treated unit(s). This differs from Y.pre only in the case 'effect' in `scdataMulti()` is set to either 'unit' or 'time'.

`Y.post.agg` a matrix containing the aggregate post-treatment outcome of the treated unit(s). This differs from Y.post only in the case 'effect' in `scdataMulti()` is set to either 'unit' or 'time'.

`Y.donors` a matrix containing the pre-treatment outcome of the control units.

`specs` a list containing some specifics of the data:

- `J`, the number of control units
- `K`, a numeric vector with the number of covariates used for adjustment for each feature
- `M`, number of features
- `KM`, the total number of covariates used for adjustment
- `KMI`, the total number of covariates used for adjustment
- `I`, number of treated unit(s)
- `period.pre`, a numeric vector with the pre-treatment period
- `period.post`, a numeric vector with the post-treatment period
- `T0.features`, a numeric vector with the number of periods used in estimation for each feature
- `T1.outcome`, the number of post-treatment periods
- `constant`, for internal use only
- `effect`, for internal use only
- `anticipation`, number of periods of potential anticipation effects
- `out.in.features`, for internal use only
- `treated.units`, list containing the IDs of all treated units
- `donors.list`, list containing the IDs of the donors of each treated unit

The second list is labeled 'est.results' containing all the results from `scest`.

`w` a matrix containing the estimated weights of the donors.

`r` a matrix containing the values of the covariates used for adjustment.

`b` a matrix containing $\mathbf{w}$ and $\mathbf{r}$.

`Y.pre.fit` a matrix containing the estimated pre-treatment outcome of the SC unit(s).

`Y.post.fit` a matrix containing the estimated post-treatment outcome of the SC unit(s).

`A.hat` a matrix containing the predicted values of the features of the treated unit(s).

`res` a matrix containing the residuals $\mathbf{A} - \widehat{\mathbf{A}}$.

V	a matrix containing the weighting matrix used in estimation.
w.constr	a list containing the specifics of the constraint set used on the weights.

The third list is labeled 'inference.results' and contains all the inference-related results.

CI.in.sample	a matrix containing the prediction intervals taking only in-sample uncertainty in to account.
CI.all.gaussian	a matrix containing the prediction intervals estimating out-of-sample uncertainty with sub-Gaussian bounds.
CI.all.ls	a matrix containing the prediction intervals estimating out-of-sample uncertainty with a location-scale model.
CI.all.qreg	a matrix containing the prediction intervals estimating out-of-sample uncertainty with quantile regressions.
bounds	a list containing the estimated bounds (in-sample and out-of-sample uncertainty).
Sigma	a matrix containing the estimated (conditional) variance-covariance $\mathbf{\Sigma}$.
u.mean	a matrix containing the estimated (conditional) mean of the pseudo-residuals $\mathbf{u}$.
u.var	a matrix containing the estimated (conditional) variance-covariance of the pseudo-residuals $\mathbf{u}$.
e.mean	a matrix containing the estimated (conditional) mean of the out-of-sample error e.
e.var	a matrix containing the estimated (conditional) variance of the out-of-sample error e.
u.missp	a logical indicating whether the model has been treated as misspecified or not.
u.lags	an integer containing the number of lags in B used in predicting moments of the pseudo-residuals $\mathbf{u}$.
u.order	an integer containing the order of the polynomial in B used in predicting moments of the pseudo-residuals $\mathbf{u}$.
u.sigma	a string indicating the estimator used for Sigma.
u.user	a logical indicating whether the design matrix to predict moments of $\mathbf{u}$ was user-provided.
u.T	a scalar indicating the number of observations used to predict moments of $\mathbf{u}$.
u.params	a scalar indicating the number of parameters used to predict moments of $\mathbf{u}$.
u.D	the design matrix used to predict moments of $\mathbf{u}$,
u.alpha	a scalar determining the confidence level used for in-sample uncertainty, i.e. 1-u.alpha is the confidence level.
e.method	a string indicating the specification used to predict moments of the out-of-sample error e.
e.lags	an integer containing the number of lags in B used in predicting moments of the out-of-sample error e.
e.order	an integer containing the order of the polynomial in B used in predicting moments of the out-of-sample error e.

`e.user`	a logical indicating whether the design matrix to predict moments of e was user-provided.
`e.T`	a scalar indicating the number of observations used to predict moments of $\mathbf{u}$.
`e.params`	a scalar indicating the number of parameters used to predict moments of $\mathbf{u}$.
`e.alpha`	a scalar determining the confidence level used for out-of-sample uncertainty, i.e. 1-`e.alpha` is the confidence level.
`e.D`	the design matrix used to predict moments of $\mathbf{u}$,
`rho`	an integer specifying the estimated regularizing parameter that imposes sparsity on the estimated vector of weights.
`Q.star`	a list containing the regularized constraint on the norm.
`sims`	an integer indicating the number of simulations used in quantifying in-sample uncertainty.
`failed.sims`	a matrix containing the percentage of failed simulations per post-treatment period to estimate lower and upper bounds.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

`scdata`, `scdataMulti`, `scest`, `scplot`, `scplotMulti`

Examples

```
data <- scpi_germany

df <- scdata(df = data, id.var = "country", time.var = "year",
             outcome.var = "gdp", period.pre = (1960:1990),
             period.post = (1991:2003), unit.tr = "West Germany",
             unit.co = setdiff(unique(data$country), "West Germany"),
             constant = TRUE, cointegrated.data = TRUE)

result <- scpi(df, w.constr = list(name = "simplex", Q = 1), cores = 1, sims = 10)
result <- scpi(df, w.constr = list(lb = 0, dir = "==", p = "L1", Q = 1),
               cores = 1, sims = 10)
```

---

scpi_germany *Replication Dataset for Estimating the Economic Impact of German Reunification*

Description

A dataset containing some economic indicators of 17 OECD countries from 1960 to 2003.

Usage

```
scpi_germany
```

Format

A data frame with 748 rows and 11 variables:

index country index.

country name of the country.

year time index, in years.

gdp GDP per Capita (PPP, 2002 USD).

infrate annual percentage change in consumer prices (base year 1995).

trade trade openness measured as export plus imports as percentage of GDP.

schooling percentage of secondary school attained in the total population aged 25 and older.

industry industry share of value added.

Source

Harvard Dataverse (doi:10.7910/DVN/24714)

scplot *Plot Synthetic Control Point Estimates and Prediction Interval*

Description

The command plots the actual pre-treatment and post-treatment series of the treated unit and the estimated counterfactual synthetic control unit with corresponding prediction intervals. Prediction intervals can take into account either in-sample uncertainty only or in-sample and out-of-sample uncertainty using the techniques developed in to Cattaneo et al. (2021) and Cattaneo et al. (2027). The input object should come from the command `scest` or from the command `scpi`.

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdata and scdataMulti for data preparation in the single and multiple treated unit(s) cases, respectively, scest for point estimation, scpi for inference procedures, and scplotMulti for plots with multiple treated units.

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.

Usage

```
scplot(
  result,
  fig.path = NULL,
  fig.name = NULL,
  fig.format = "png",
  e.out = TRUE,
  joint = FALSE,
  col.treated = "black",
  col.synth = "mediumblue",
  label.xy = NULL,
  plot.range = NULL,
  x.ticks = NULL,
  event.label = NULL,
  plot.specs = NULL,
  save.data = NULL
)
```

Arguments

`result`	a class 'scest' object, obtained by calling `scest`, or a class 'scpi' object, obtained by calling `scpi`.
`fig.path`	a string indicating the path where the plot(s) should be saved.
`fig.name`	a string indicating the name of the plot(s). If multiple plots will be saved the command automatically generates a numeric suffix to avoid overwriting them.

fig.format	a string indicating the format in which the plot(s) should be saved.
e.out	a logical specifying whether out-of-sample uncertainty should be included in the plot(s).
joint	a logical specifying whether simultaneous prediction intervals should be included in the plot(s). It requires e.out = TRUE.
col.treated	a string specifying the color for the treated unit series. Find the full list at http://sape.inf.usi.ch/quick-reference/ggplot2/colour.
col.synth	a string specifying the color for the synthetic unit series. Find the full list at http://sape.inf.usi.ch/quick-reference/ggplot2/colour.
label.xy	a character list with two elements indicating the name of the axes (eg. label.xy = list(x.lab = "Year", y.lab = "GDP growth (%)")).
plot.range	a numeric array indicating the time range of the plot(s).
x.ticks	a numeric list containing the location of the ticks on the x axis.
event.label	a list containing a character object ('lab') indicating the label of the event and a numeric object indicating the height of the label in the plot.
plot.specs	a list containing some specifics to be passed to ggsave (eg. img.width, img.height, dpi)
save.data	a character specifying the name and the path of the saved dataframe containing the processed data used to produce the plot.

Value

plots	a list containing standard ggplot object(s) that can be used for further customization.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

scdata, scdataMulti, scest, scpi, scplotMulti

Examples

```
data <- scpi_germany

df <- scdata(df = data, id.var = "country", time.var = "year",
             outcome.var = "gdp", period.pre = (1960:1990),
             period.post = (1991:2003), unit.tr = "West Germany",
             unit.co = setdiff(unique(data$country), "West Germany"),
             constant = TRUE, cointegrated.data = TRUE)

result <- scest(df, w.constr = list(name = "simplex", Q = 1))

scplot(result)
```

---

scplotMulti *Plot Synthetic Control Point Estimates and Prediction Interval With Multiple Treated units and Staggered Adoption*

Description

The command produces a wide range of plots of Synthetic Control estimates and corresponding prediction intervals. The command allows form multiple treated units and staggered adoption. Prediction intervals can take into account either in-sample uncertainty only or in-sample and out-of-sample uncertainty using the techniques developed in to Cattaneo et al. (2021) and Cattaneo et al. (2027). The input object should come from the command scest or from the command scpi.

Companion Stata and Python packages are described in Cattaneo et al. (2025).

Companion commands are: scdata and scdataMulti for data preparation in the single and multiple treated unit(s) cases, respectively, scest for point estimation, scpi for inference procedures, and scplotMulti for plots with multiple treated units.

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.

Usage

```
scplotMulti(
  result,
  type = "series",
  e.out = TRUE,
  joint = FALSE,
  col.treated = "black",
  col.synth = "mediumblue",
  scales = "fixed",
  point.size = 1.5,
  ncols = 3,
  save.data = NULL,
  verbose = TRUE
)
```

Arguments

result	a class 'scest' object, obtained by calling scest, or a class 'scpi' object, obtained by calling scpi. The data object given as input to this command has to be processed with scdataMulti.
type	a character that specifies the type of plot to be produced. If set to 'treatment' then treatment effects are plotted. If set to 'series' (default), the actual and synthetic time series are reported.
e.out	a logical specifying whether out-of-sample uncertainty should be included in the plot(s).
joint	a logical specifying whether simultaneous prediction intervals should be included in the plot(s). It requires e.out = TRUE.
col.treated	a string specifying the color for the treated unit series. Find the full list at http://sape.inf.usi.ch/quick-reference/ggplot2/colour.
col.synth	a string specifying the color for the synthetic unit series. Find the full list at http://sape.inf.usi.ch/quick-reference/ggplot2/colour.
scales	should axes scales be fixed ("fixed", the default), free ("free"), or free in one dimension ("free_x", "free_y")?
point.size	a scalar controlling the size of points in the scatter plot. Default is 1.5.
ncols	an integer controlling the number of columns in the plot.
save.data	a character specifying the name and the path of the saved dataframe containing the processed data used to produce the plot.
verbose	if TRUE prints additional information in the console.

Value

plots	a list containing standard ggplot object(s) that can be used for further customization.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

References

Abadie A (2021). "Using Synthetic Controls: Feasibility, Data Requirements, and Methodological Aspects." *Journal of Economic Literature*, **59**(2), 391–425. ISSN 0022-0515. doi:10.1257/jel.20191450.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2025). "scpi: Uncertainty Quantification for Synthetic Control Methods." *Journal of Statistical Software*, **113**(1), 1–38. doi:10.18637/jss.v113.i01.

Cattaneo MD, Feng Y, Palomba F, Titiunik R (2027). "Uncertainty Quantification in Synthetic Controls with Staggered Treatment Adoption." *Review of Economics and Statistics*. doi:10.1162/rest_a_01588.

Cattaneo MD, Feng Y, Titiunik R (2021). "Prediction Intervals for Synthetic Control Methods." *Journal of the American Statistical Association*, **116**(536), 1865–1880. ISSN 0162-1459. doi:10.1080/01621459.2021.1979561.

See Also

scdata, scdataMulti, scest, scpi, scplotMulti

Examples

```
datager <- scpi_germany

datager$tr_id <- 0
datager$tr_id[(datager$country == "West Germany" & datager$year > 1990)] <- 1
datager$tr_id[(datager$country == "Italy" & datager$year > 1992)] <- 0

outcome.var <- "gdp"
id.var <- "country"
treatment.var <- "tr_id"
time.var <- "year"
df.unit <- scdataMulti(datager, id.var = id.var, outcome.var = outcome.var,
                       treatment.var = treatment.var,
                       time.var = time.var, features = list(c("gdp", "trade")),
                    cointegrated.data = TRUE, constant = TRUE)

res.unit <- scpi(df.unit, sims = 10, cores = 1)
scplotMulti(res.unit, joint = TRUE)
```

summary.scdata *Summary Method for Synthetic Control Prediction*

Description

The summary method for synthetic control prediction objects.

Usage

```
## S3 method for class 'scdata'
summary(object, ...)
```

Arguments

object	Class "scest" object, obtained by calling scdata.
...	Additional arguments

Value

No return value, called to summarize scdata results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scdata

Supported methods: print.scdata, summary.scdata.

summary.scdataMulti *Summary Method for Synthetic Control Prediction*

Description

The summary method for synthetic control prediction objects.

Usage

```
## S3 method for class 'scdataMulti'
summary(object, ...)
```

Arguments

object	Class "scdataMulti" object, obtained by calling scdataMulti.
...	Additional arguments

Value

No return value, called to summarize scdataMulti results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scdataMulti

Supported methods: print.scdataMulti, summary.scdataMulti.

---

summary.scest *Summary Method for Synthetic Control Prediction*

Description

The summary method for synthetic control prediction fitted objects.

Usage

```
## S3 method for class 'scest'
summary(object, ...)
```

Arguments

object	Class "scest" object, obtained by calling scest.
...	Additional arguments

Value

No return value, called to summarize scest results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scest

Supported methods: print.scest, summary.scest, coef.scest.

---

summary.scpi *Summary Method for Synthetic Control Inference*

Description

The summary method for synthetic control inference objects.

Usage

```
## S3 method for class 'scpi'
summary(object, ...)
```

Arguments

object	Class "scpi" object, obtained by calling scpi.
...	Additional arguments

Value

No return value, called to summarize scpi results.

Author(s)

Matias D. Cattaneo, Princeton University. <matias.d.cattaneo@gmail.com>.

Yingjie Feng, Tsinghua University. <fengyingjiepku@gmail.com>.

Filippo Palomba, Princeton University. <filippo.palomba19@gmail.com>.

Rocio Titiunik, Princeton University. <rocio.titiunik@gmail.com>.

See Also

scpi

Supported methods: print.scpi, summary.scpi.

Index