

Package: Synth (via r-universe)

October 4, 2026

Version 1.2-0

Date 2026-10-04

Title Synthetic Control Group Method for Comparative Case Studies

Description Implements the synthetic control group method for comparative case studies as described in Abadie and Gardeazabal (2003) and Abadie, Diamond, and Hainmueller (2010, 2011, 2014). The synthetic control method allows for effect estimation in settings where a single unit (a state, country, firm, etc.) is exposed to an event or intervention. It provides a data-driven procedure to construct synthetic control units based on a weighted combination of comparison units that approximates the characteristics of the unit that is exposed to the intervention. A combination of comparison units often provides a better comparison for the unit exposed to the intervention than any comparison unit alone.

Imports kernlab, optimx, rgenoud

Suggests testthat (>= 3.0.0), knitr, rmarkdown, CVXR (>= 1.8.0), torch, ggplot2

Config/testthat/edition 3

VignetteBuilder knitr

License GPL (>= 2)

Depends R (>= 3.6.0)

URL <https://web.stanford.edu/~jhain/>

Encoding UTF-8

Config/pak/sysreqs cmake

Repository <https://j-hai.r-universe.dev>

Date/Publication 2026-10-04 15:28:51 UTC

RemoteUrl <https://github.com/j-hai/synth>

RemoteRef HEAD

RemoteSha b3e2eaa6ebda879ca2e79d34e2f810d9e6b747a0

Contents

basque	2
collect.optimx	4
dataprep	5
fn.V	11
gaps.plot	13
path.plot	14
plot.synth_inference	16
plot.synth_placebos	17
smoking	18
spec.pred.func	19
synth	21
synth.data	26
synth.tab	26
synth_data	27
synth_inference	29
synth_mspe_plot	32
synth_mspe_test	32
synth_placebos	33

Index	37
--------------	-----------

basque	<i>Panel Data from Spanish Regions to demonstrate the use of the Synthetic Control Method</i>
--------	---

Description

The dataset contains information from 1955–1997 on 17 Spanish regions. It was used by Abadie and Gardeazabal (2003), which studied the economic effects of conflict, using the terrorist conflict in the Basque Country as a case study. This paper used a combination of other Spanish regions to construct a synthetic control region resembling many relevant economic characteristics of the Basque Country before the onset of political terrorism in the 1970s. The data contains per-capita GDP (the outcome variable), as well as population density, sectoral production, investment, and human capital (the predictor variables) for the relevant years, and is used here to demonstrate the implementation of the synthetic control method with the synth library.

Usage

```
data(basque)
```

Format

A panel dataframe made up of 18 units: 1 treated (no 17; the Basque country) and 16 control regions (no. 2-16,18). Region no. 1 is the average for the whole country of Spain. 1 outcome variable (gdpcap). 13 predictor variables (6 sectoral production shares, 5 highest educational attainment categories, population density, and the investment rate). Region names and numbers are stored

in `regionno` and `regionname`. 43 time periods (1955 - 1997). All columns have self-explanatory column names. For reference the variables are:

- `regionno`
: Region Number.
- `regionname`
: Region Name.
- `year`
: Year.
- `gdpcap`
: real GDP per capita (in 1986 USD, thousands).
- `sec.agriculture`
: production in agriculture, forestry, and fishing sector as a percentage of total production.
- `sec.energy`
: production in energy and water sector as a percentage of total production.
- `sec.industry`
: production in industrial sector as a percentage of total production.
- `sec.construction`
: production in construction and engineering sector as a percentage of total production.
- `sec.services.venta`
: production in marketable services sector as a percentage of total production.
- `sec.services.nonventa`
: production in Nonmarketable services sector as a percentage of total production.
- `school.illit`
: number of illiterate persons.
- `school.prim`
: number of persons with primary education or without studies.
- `school.med`
: number of persons with some high school education.
- `school.high`
: number of persons with high school degree.
- `school.post.high`
: number of persons with tertiary education.
- `popdens`
: population density (persons per square kilometer).
- `invest`
: gross total investment as a share of GDP.

Source

Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.

collect.optimx	<i>Collect results from optimx optimization methods</i>
----------------	---

Description

An internal function that collects the results from the different optimization methods run by optimx. It stores the parameter and function values and extracts the results for the best performing method (minimum or maximum).

Usage

```
collect.optimx(res, opt = "min")
```

Arguments

res	Output from a call to optimx().
opt	Either "min" or "max" to extract results for the methods that obtained the minimum or maximum function value across the methods.

Value

out.list	Dataframe with results from the different methods.
par	Parameter values from method that attained minimum/maximum across the methods.
value	Function value from method that attained minimum/maximum across the methods.

Author(s)

Jens Hainmueller

See Also

Also see [optimx](#).

dataprep	<i>Constructs a list of matrices from panel dataset to be loaded into synth()</i>
----------	---

Description

The `dataprep` function takes a standard panel dataset and produces a list of data objects necessary for running `synth` and other Synth package functions to construct synthetic control groups according to the methods outlined in Abadie and Gardeazabal (2003) and Abadie, Diamond, Hainmueller (2010, 2011, 2014) (see references and example).

User supplies a dataframe ("foo"), chooses predictors, special predictors (explained below), the operators that act upon these predictors, the dependent variable, identifies the columns associated with unit numbers, time periods (and unit names, when available), as well as the treated unit, the control units, the time-period over which to select the predictors, the time-period over which to optimize, and the time-period over which outcome data should be plotted.

The output of `dataprep` contains a list of matrices. This list object can be directly loaded into `synth`.

Usage

```
dataprep(foo = NULL, predictors = NULL,
         predictors.op = "mean", special.predictors = NULL,
         dependent = NULL, unit.variable = NULL,
         time.variable = NULL, treatment.identifier = NULL,
         controls.identifier = NULL, time.predictors.prior = NULL,
         time.optimize.ssr = NULL, time.plot = time.optimize.ssr,
         unit.names.variable = NA)
```

Arguments

<code>foo</code>	The dataframe with the panel data.
<code>predictors</code>	A vector of column numbers or column-name character strings that identifies the predictors' columns. All predictors have to be numeric.
<code>predictors.op</code>	A character string identifying the method (operator) to be used on the predictors, for the treated and the control units alike. Default is "mean". <code>na.rm = T</code> is hardwired into the code. See <i>*Details*</i> .
<code>special.predictors</code>	A list object identifying additional numeric predictors and their associated pre-treatment years and operators (analogous to "predictors.op" above). See <i>*Details*</i> .
<code>dependent</code>	A scalar identifying the column number or column-name character string that corresponds to the numeric dependent (outcome) variable.
<code>unit.variable</code>	A scalar identifying the column number or column-name character string associated with unit numbers. The <code>unit.variable</code> has to be numeric.

<code>time.variable</code>	A scalar identifying column number or column-name character string associated with period (time) data. The time variable has to be numeric.
<code>treatment.identifier</code>	A scalar identifying the “unit.variable” number or a character string giving the “unit.name” of the treated unit. If a character is supplied, a <code>unit.names.variable</code> also has to be supplied to identify the treated unit.
<code>controls.identifier</code>	A vector identifying the “unit.variable” numbers or a vector of character strings giving the “unit.name”s of control units. If a character is supplied, a <code>unit.names.variable</code> also has to be supplied to identify the control units. The order in which the control units are listed does not matter: the output always holds them in ascending order of their “unit.variable” number.
<code>time.predictors.prior</code>	A numeric vector identifying the pretreatment periods over which the values of the predictors are aggregated with <code>predictors.op</code> (averaged, by default).
<code>time.optimize.ssr</code>	A numeric vector identifying the periods of the dependent variable over which the loss function should be minimized (i.e. the periods over which mean squared prediction error (MSPE), that is the sum of squared residuals between treated and the synthetic control unit, are minimized).
<code>time.plot</code>	A vector identifying the periods over which results are to be plotted with gaps.plot and path.plot .
<code>unit.names.variable</code>	A scalar or column-name character string identifying the column with the names of the units. This variable has to be of mode character.

Details

The `predictors.op` argument is a character string that provides a function (eg., “mean”, “median”, etc.) identifying the name of the operator to be applied to the predictors over the given time period. The function is called as `f(x, na.rm = TRUE)` on one unit’s values of one predictor and has to return a single number. A function of your own has to be defined at top level (or come from an attached package), under a name that base R, Synth, and the packages Synth imports do not already use. `dataprep()` stops with an explanatory message if the name is not a single character string, no function of that name is found, the function cannot be called with `na.rm = TRUE`, or it returns anything other than one number (as “range” does, for example).

The `special.predictors` argument is a list object that contains one or more lists of length = 3. The required components of each of these lists are:

- (a) scalar column number or column-name character string associated with that predictor
- (b) vector of time-period number(s) desired (eg., 1998:2003)
- (c) character-string identifying the name of the operation to be applied (ie., “mean”, “median”, etc.). It is used, and checked in the same way as `predictors.op`, only when (b) holds more than one time period.

eg., `special.predictors <- list(list(x1, 1985:1989, "mean"), list(x2, 1980:1983, "median"), list(x3, 1980, "mean"))`

indicates that predictor x1 should be used with its values averaged over periods 1985:1989; predictor x2 should be used with its median values over periods 1980:1983; x3 should be used with the values from period 1980 only.

Value

X1	matrix of treated predictor data. nrow = number of predictors and (possibly) special predictors. ncol = one.
X0	matrix of controls' predictor data. nrow = number of predictors and (possibly) special predictors. ncol = number of control units.
Z1	matrix of treated outcome data for the pre-treatment periods over which MSPE is to be minimized. nrow = number of pre-treatment periods. ncol = one.
Z0	matrix of controls' outcome data for the pre-treatment periods over which MSPE is to be minimized. nrow = number of pre-treatment periods. ncol = number of control units.
Y1plot	matrix of outcome data for treated unit to be used for results plotting. nrow = number of periods. ncol = one.
Y0plot	matrix of outcome data for control units to be used for results plotting. nrow = number of periods. ncol = number of control units.
names.and.numbers	dataframe with two columns showing all unit numbers and corresponding unit names.
tag	a list of all arguments in initial function call. controls.identifier (the column order of X0, Z0 and Y0plot), time.optimize.ssr (the row order of Z0 and Z1), time.plot (the row order of Y0plot and Y1plot) and time.predictors.prior are stored in ascending order.

Author(s)

Jens Hainmueller and Alexis Diamond

References

- Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.
- Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.
- Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.
- Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[synth](#), [gaps.plot](#), [path.plot](#), [synth.tab](#)

Examples

```
## The usual sequence of commands is:
## 1. dataprep() for matrix-extraction
## 2. synth() for the construction of the synthetic control group
## 3. synth.tab(), gaps.plot(), and path.plot() to summarize the results
## Below we provide two examples.

## First Example: Toy panel dataset

# load data
data(synth.data)

# create matrices from panel data that provide inputs for synth()
dataprep.out<-
  dataprep(
    foo = synth.data,
    predictors = c("X1", "X2", "X3"),
    predictors.op = "mean",
    dependent = "Y",
    unit.variable = "unit.num",
    time.variable = "year",
    special.predictors = list(
      list("Y", 1990, "mean"),
      list("Y", 1985, "mean"),
      list("Y", 1980, "mean")
    ),
    treatment.identifier = 7,
    controls.identifier = c(29, 2, 13, 17, 32, 38),
    time.predictors.prior = c(1984:1989),
    time.optimize.ssr = c(1984:1990),
    unit.names.variable = "name",
    time.plot = 1984:1996
  )

## run the synth command to identify the weights
## that create the best possible synthetic
## control unit for the treated.
synth.out <- synth(dataprep.out)

## there are two ways to summarize the results
## we can either access the output from synth.out directly
round(synth.out$solution.w,2)
# contains the unit weights or
synth.out$solution.v
## contains the predictor weights.

## the output from synth opt
## can be flexibly combined with
## the output from dataprep to
## compute other quantities of interest
## for example, the period by period
## discrepancies between the
```

```

## treated unit and its synthetic control unit
## can be computed by typing
gaps<- dataprep.out$Y1plot-(
    dataprep.out$Y0plot*%synth.out$solution.w
) ; gaps

## also there are three convenience functions to summarize results.
## to get summary tables for all information
## (V and W weights plus balance btw.
## treated and synthetic control) use the
## synth.tab() command
synth.tables <- synth.tab(
    dataprep.res = dataprep.out,
    synth.res = synth.out)
print(synth.tables)

## to get summary plots for outcome trajectories
## of the treated and the synthetic control unit use the
## path.plot() and the gaps.plot() commands

## plot in levels (treated and synthetic)
path.plot(dataprep.res = dataprep.out,synth.res = synth.out)

## plot the gaps (treated - synthetic)
gaps.plot(dataprep.res = dataprep.out,synth.res = synth.out)


## Second example: The economic impact of terrorism in the
## Basque country using data from Abadie and Gardeazabal (2003)
## see JSS paper in the references details

data(basque)

# dataprep: prepare data for synth
dataprep.out <-
  dataprep(
    foo = basque
    ,predictors= c("school.illit",
                  "school.prim",
                  "school.med",
                  "school.high",
                  "school.post.high"
                  ,"invest"
                  )
    ,predictors.op = c("mean")
    ,dependent      = c("gdpcap")
    ,unit.variable  = c("regionno")
    ,time.variable  = c("year")
    ,special.predictors = list(
      list("gdpcap",1960:1969,c("mean")),
      list("sec.agriculture",seq(1961,1969,2),c("mean")),
      list("sec.energy",seq(1961,1969,2),c("mean")),
    )
  )

```

```

    list("sec.industry",seq(1961,1969,2),c("mean")),
    list("sec.construction",seq(1961,1969,2),c("mean")),
    list("sec.services.venta",seq(1961,1969,2),c("mean")),
    list("sec.services.nonventa",seq(1961,1969,2),c("mean")),
    list("popdens",1969,c("mean")))
  ,treatment.identifier = 17
  ,controls.identifier = c(2:16,18)
  ,time.predictors.prior = c(1964:1969)
  ,time.optimize.ssr = c(1960:1969)
  ,unit.names.variable = c("regionname")
  ,time.plot = c(1955:1997)
)

# 1. combine highest and second highest
# schooling category and eliminate highest category
dataprep.out$X1["school.high",] <-
  dataprep.out$X1["school.high",] +
  dataprep.out$X1["school.post.high",]
dataprep.out$X1 <-
  as.matrix(dataprep.out$X1[
    -which(rownames(dataprep.out$X1)=="school.post.high"),])
dataprep.out$X0["school.high",] <-
  dataprep.out$X0["school.high",] +
  dataprep.out$X0["school.post.high",]
dataprep.out$X0 <-
  dataprep.out$X0[
    -which(rownames(dataprep.out$X0)=="school.post.high"),]

# 2. make total and compute shares for the schooling categories
lowest <- which(rownames(dataprep.out$X0)=="school.illit")
highest <- which(rownames(dataprep.out$X0)=="school.high")

dataprep.out$X1[lowest:highest,] <-
  (100 * dataprep.out$X1[lowest:highest,]) /
  sum(dataprep.out$X1[lowest:highest,])
dataprep.out$X0[lowest:highest,] <-
  100 * scale(dataprep.out$X0[lowest:highest,],
    center=FALSE,
    scale=colSums(dataprep.out$X0[lowest:highest,])
  )

# run synth
synth.out <- synth(data.prep.obj = dataprep.out)

# Get result tables
synth.tables <- synth.tab(
  dataprep.res = dataprep.out,
  synth.res = synth.out
)

# results tables:
print(synth.tables)

```

```

# plot results:
# path
path.plot(synth.res = synth.out,
          dataprep.res = dataprep.out,
          Ylab = c("real per-capita GDP (1986 USD, thousand)"),
          Xlab = c("year"),
          Ylim = c(0,13),
          Legend = c("Basque country", "synthetic Basque country"),
          )

## gaps
gaps.plot(synth.res = synth.out,
          dataprep.res = dataprep.out,
          Ylab = c("gap in real per-capita GDP (1986 USD, thousand)"),
          Xlab = c("year"),
          Ylim = c(-1.5,1.5),
          )

## To create the placebo studies simply reassign
## the intervention to other units or times (see references for details).
## synth_placebos() runs the placebo fits for all control units.

```

fn.V

Loss Function for nested optimization of W and V weights

Description

Loss function for the nested optimization of W and V weights used for constructing synthetic control groups according to the methods outlined in Abadie and Gardeazabal (2003) and Abadie, Diamond, Hainmueller (2010, 2011, 2014) (see references). This function is called by `synth` internally, and should not be called manually by a normal user.

Usage

```

fn.V(variables.v = stop("variables.v missing"),
      X0.scaled = stop("X0.scaled missing"),
      X1.scaled = stop("X1.scaled missing"),
      Z0 = stop("Z0 missing"),
      Z1 = stop("Z1 missing"),
      margin.ipop = 5e-04,
      sigf.ipop = 5,
      bound.ipop = 10,
      quadopt = "ipop",
      cvxr_pars = list(),
      torch_pars = list())

```

Arguments

<code>variables.v</code>	a 1 by k vector of v weights.
<code>X0.scaled</code>	matrix of controls' predictor data. nrow = number of predictors and (possibly) special predictors. ncol = number of control units.
<code>X1.scaled</code>	matrix of treated predictor data. nrow = number of predictors and (possibly) special predictors. ncol = one.
<code>Z0</code>	matrix of controls' outcome data for the pre-treatment periods over which MSPE is to be minimized. nrow = number of pre-treatment periods. ncol = number of control units.
<code>Z1</code>	matrix of treated outcome data for the pre-treatment periods over which MSPE is to be minimized. nrow = number of pre-treatment periods. ncol = one.
<code>margin.ipop</code>	setting for ipop optimization routine: how close we get to the constraints (see ipop for details)
<code>sigf.ipop</code>	setting for ipop optimization routine: Precision (default: 5 significant figures) (see ipop for details)
<code>bound.ipop</code>	setting for ipop optimization routine: Clipping bound for the variables (see ipop for details)
<code>quadopt</code>	string vector that specifies the routine for quadratic optimization over w weights. One of "ipop" (default; see ipop), "cvxr" (CVXR with CLARABEL as the default solver), or "torch" (Frank-Wolfe simplex least squares via the torch package).
<code>cvxr_pars</code>	Optional named list of CVXR backend tuning. Recognized fields: solver (default "CLARABEL"; "OSQP", "SCS" and "ECOS" also supported, the latter requiring ECOSolveR), eps (default 1e-8), max_iter (default 5000). Ignored unless quadopt = "cvxr".
<code>torch_pars</code>	Optional named list of torch backend tuning. Recognized fields: max_iter (default 500), tol (default 1e-8), device (default "cpu"; "cuda" or "mps" for GPU), dtype (default "float64"). Ignored unless quadopt = "torch". Both cvxr_pars and torch_pars are forwarded by synth to every fn.V invocation during V-search, so by default the inner solves use the same backend configuration as the final W solve (the *_inner and *_outer arguments of synth override this per stage).

Value

A scalar that contains the function value.

Author(s)

Jens Hainmueller and Alexis Diamond

References

Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.

Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.

Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California’s Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.

Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[synth](#), [dataprep](#), [gaps.plot](#), [path.plot](#), [synth.tab](#)

gaps.plot	<i>Plots Gap in Outcome Trajectories between the Treated and its Synthetic Control Unit</i>
-----------	---

Description

This function plots the gaps in the trajectories of the outcome variable for the treated unit and the synthetic control group constructed by [synth](#) and [dataprep](#). The user can specify whether the whole time period or only the pre-treatment period should be plotted.

Usage

```
gaps.plot(synth.res = NA,
          dataprep.res = NA,
          Ylab = c("Title"),
          Xlab = c("Time"),
          Main = c("Gaps: Treated - Synthetic"),
          tr.intake = NA,
          Ylim = NA,
          Z.plot = FALSE)
```

Arguments

synth.res	Output list created by synth .
dataprep.res	Output list created by dataprep .
tr.intake	Optional scalar to indicate the time of treatment intake with a vertical line.
Ylab	Optional label for Y axis.
Xlab	Optional label for X axis.
Ylim	Optional Ylim.
Main	Optional main title.
Z.plot	Flag. If true, only pretreatment period is plotted.

Details

The trajectory of the outcome for the synthetic control group is calculated as: `dataprep.res$Y0plot %*% synth.res$solution.w`. You can use this calculation to construct custom made plots.

Value

The plot of trajectories.

Author(s)

Jens Hainmueller and Alexis Diamond

References

- Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.
- Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.
- Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.
- Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[dataprep](#), [synth](#), [path.plot](#), [synth.tab](#)

path.plot	<i>Plots Outcome Trajectories for Treated Unit and its Synthetic Control Unit</i>
-----------	---

Description

This function plots the trajectories of the outcome variable for the treated unit and the synthetic control group constructed by [synth](#) and [dataprep](#). The user can specify whether the whole time period or only the pretreatment period should be plotted.

Usage

```
path.plot(synth.res = NA,
          dataprep.res = NA,
          tr.intake = NA,
          Ylab = c("Y Axis"),
          Xlab = c("Time"),
          Ylim = NA,
```

```

Legend=c("Treated", "Synthetic"),
Legend.position=c("topright"),
Main = NA,
Z.plot = FALSE)

```

Arguments

synth.res	Output list created by synth .
dataprep.res	Output list created by dataprep .
tr.intake	Optional scalar to indicate the time of treatment intake with a vertical line.
Ylab	Optional label for Y axis.
Xlab	Optional label for X axis.
Ylim	Optional Ylim.
Main	Optional main title.
Legend	Optional legend text (e.g. c("Treated", "Synthetic")); see ?legend for details.
Legend.position	Optional legend position (e.g. "bottomright"); see ?legend for details.
Z.plot	Flag. If true, only pretreatment period is plotted.

Details

The trajectory of the outcome for the synthetic control group is calculated as: `dataprep.res$Y0plot%%synth.res$solution.w`. You can use this calculation to construct custom made plots.

Value

The plot of trajectories.

Author(s)

Jens Hainmueller and Alexis Diamond

References

Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.

Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.

Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.

Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[dataprep](#), [gaps.plot](#), [synth](#), [synth.tab](#)

plot.synth_inference *Plot a synthetic control prediction band*

Description

Plots the treated and synthetic series with a shaded prediction band on the same axes. The band comes from [synth_inference](#).

Usage

```
## S3 method for class 'synth_inference'
plot(x,
      Ylab = "Y",
      Xlab = "Time",
      Main = NA,
      Ylim = NA,
      Legend = c("Treated", "Synthetic", paste0(100 * (1 - x$alpha), "% band")),
      Legend.position = "topright",
      tr.intake = NA,
      band.col = grDevices::rgb(0, 0, 0, 0.15),
      ...)
```

Arguments

x	Object of class synth_inference.
Ylab	Y axis label.
Xlab	X axis label.
Main	Optional main title.
Ylim	Optional Ylim. If NA, padded automatically.
Legend	Legend entries. Set to NA to suppress.
Legend.position	Legend position; passed to legend .
tr.intake	Optional scalar locating the treatment date with a vertical dashed line. If NA, defaults to the first post-treatment period in x\$time.
band.col	Fill color for the prediction band; default a translucent grey.
...	Currently unused.

Value

Invisibly returns x.

See Also

[synth_inference](#)

plot.synth_placebos *Overlay placebo gap series with the treated gap*

Description

Overlay plot of placebo gap series (in grey) with the treated gap highlighted in black. Operates on the output of `synth_placebos`.

Usage

```
## S3 method for class 'synth_placebos'
plot(x,
      mspe_threshold = NULL,
      Ylab = "Gap",
      Xlab = "Time",
      Main = "Placebo Gaps",
      Ylim = NA,
      tr.intake = NA,
      treated_col = "black",
      placebo_col = "grey60",
      ...)
```

Arguments

<code>x</code>	Output of <code>synth_placebos</code> .
<code>mspe_threshold</code>	Optional scalar; if supplied, placebos whose pre-period MSPE exceeds <code>mspe_threshold * treated_pre_mspe</code> are dropped from the plot. The standard idiom is <code>mspe_threshold = 5</code> (drop placebos whose own pre-period MSPE is more than five times the treated unit's).
<code>Ylab</code>	Y axis label.
<code>Xlab</code>	X axis label.
<code>Main</code>	Optional main title.
<code>Ylim</code>	Optional Ylim. If NA, padded automatically.
<code>tr.intake</code>	Optional scalar locating the treatment date with a vertical dotted line. If NA, defaults to <code>x\$treatment_time</code> , the first post-treatment period.
<code>treated_col</code>	Color for the treated gap line.
<code>placebo_col</code>	Color for placebo gap lines.
<code>...</code>	Not used; a warning is given if anything is passed. Base graphics arguments such as <code>main</code> , <code>xlab</code> or <code>ylim</code> are not passed on: use <code>Main</code> , <code>Xlab</code> , <code>Ylab</code> and <code>Ylim</code> .

Value

Invisibly returns NULL.

References

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies. *Journal of the American Statistical Association* 105 (490) 493–505.

See Also

[synth_placebos](#), [synth_mspe_test](#), [synth_mspe_plot](#).

smoking

Annual State-Level Cigarette Sales in the United States, 1970-2000

Description

A panel of per-capita cigarette sales for 39 US states from 1970 through 2000, plus four covariates (per-capita log income, beer consumption, share of population aged 15-24, and the retail price of cigarettes). This is the dataset used in the canonical synthetic control application of Abadie, Diamond, and Hainmueller (2010), which estimated the effect of California's 1988 Proposition 99 on cigarette consumption.

Usage

```
data(smoking)
```

Format

A `data.frame` with 1209 rows and 8 columns:

`state_id` Numeric state identifier (3 = California).

`state_name` State name as a character string.

`year` Calendar year (1970-2000).

`cigsale` Per-capita cigarette sales in packs.

`lnincome` Log per-capita state income.

`beer` Per-capita beer consumption.

`age15to24` Share of population aged 15-24.

`retprice` Retail price of cigarettes.

`lnincome` and `beer` are missing for some early years; the canonical Proposition 99 analysis matches on covariate averages over windows where the data are observed.

Source

Compiled from the data files distributed with Abadie, Diamond, and Hainmueller (2010) and shipped with the Stata `synth` package.

References

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California's Tobacco Control Program. *Journal of the American Statistical Association*, 105(490), 493-505.

See Also

[basque](#), [synth](#), [synth_inference](#).

Examples

```
## Not run:
data(smoking)

# California is state 3; treatment is Proposition 99 in late 1988,
# so 1989 is the first post-treatment year.
ca_id <- unique(smoking$state_id[smoking$state_name == "California"])

dp <- synth_data(
  panel          = smoking,
  outcome        = "cigsale",
  unit_col       = "state_id",
  time_col       = "year",
  treated        = ca_id,
  treatment_time = 1989,
  predictors      = c("lnincome", "age15to24", "retprice", "beer"),
  special_predictors = list(
    list("cigsale", 1988, "mean"),
    list("cigsale", 1980, "mean"),
    list("cigsale", 1975, "mean")
  ),
  unit_names_col = "state_name"
)

fit <- synth(dp)
inf <- synth_inference(fit, dp, method = "conformal", alpha = 0.10)
plot(inf)

## End(Not run)
```

spec.pred.func

Special Predictor Function for Dataprep

Description

This function is called by [dataprep](#) to handle special predictors in the process of setting up the dataset to be loaded into [synth](#). It should not be called manually by the normal user.

Usage

```
spec.pred.func(list.object = NULL,
               tr.numb = NULL,
               co.numb = NULL,
               unit.var = NULL,
               time.var = NULL,
               foo.object = NULL,
               X0.inner = NULL,
               X1.inner = NULL)
```

Arguments

list.object	NA
tr.numb	NA
co.numb	NA
unit.var	NA
time.var	NA
foo.object	NA
X0.inner	NA
X1.inner	NA

Details

The third element of `list.object` names the operator. It is used only when the second element holds more than one period; it is then looked up and checked in the same way as `predictors.op` in [dataprep](#), and applied to the treated and the control units alike. With a single period the values of that period are used as they are.

Value

NA

Author(s)

Jens Hainmueller and Alexis Diamond

References

Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.

Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.

Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California’s Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.

Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[synth](#), [dataprep](#), [gaps.plot](#), [path.plot](#), [synth.tab](#)

synth

Constructs synthetic control units for comparative case studies

Description

Implements the synthetic control method for causal inference in comparative case studies as developed in Abadie and Gardeazabal (2003) and Abadie, Diamond, and Hainmueller (2010, 2011, 2014). [synth](#) estimates the effect of an intervention by comparing the evolution of an aggregate outcome for a unit affected by the intervention to the evolution of the same aggregate outcome for a synthetic control group.

[synth](#) constructs this synthetic control group by searching for a weighted combination of control units chosen to approximate the unit affected by the intervention in terms of characteristics that are predictive of the outcome. The evolution of the outcome for the resulting synthetic control group is an estimate of the counterfactual of what would have been observed for the affected unit in the absence of the intervention.

[synth](#) can also be used to conduct a variety of placebo and permutation tests that produce informative inference regardless of the number of available comparison units and the number of available time periods. See Abadie and Gardeazabal (2003), Abadie, Diamond, and Hainmueller (2010, 2011, 2014) for details.

[synth](#) requires the user to supply four matrices as its main arguments. These matrices are named `X0`, `X1`, `Z1`, and `Z0` accordingly. `X1` and `X0` contain the predictor values for the treated unit and the control units respectively. `Z1` and `Z0` contain the outcome variable for the pre-intervention period for the treated unit and the control units respectively. The pre-intervention period refers to the time period prior to the intervention, over which the mean squared prediction error (MSPE) should be minimized. The MSPE refers to the squared deviations between the outcome for the treated unit and the synthetic control unit summed over all pre-intervention periods specified in `Z1` and `Z0`.

Creating the matrices `X1`, `X0`, `Z1`, and `Z0` from a (panel) dataset can be tedious. Therefore, the **Synth** package offers a preparatory function called [dataprep](#) that allows the user to easily create all inputs required for [synth](#). By first calling [dataprep](#), the user creates a single list object called `data.prep.obj` that contains all essential data elements to run [synth](#).

A usual sequence of commands to implement the synthetic control method is to first call [dataprep](#) to prepare the data, then call [synth](#) to construct the synthetic control group, and finally summarize results using the functions [synth.tab](#), [path.plot](#), or [gaps.plot](#).

An example of this sequence is provided in the documentation to [dataprep](#). This procedure is strongly recommended. Alternatively, the user may provide their own preprocessed data matrices and load them into [synth](#) via the `X0`, `X1`, `Z1`, and `Z0` arguments. In this case, no `data.prep.obj` should be specified.

The output from [synth](#) is a list object that contains the weights on predictors (`solution.v`) and weights on control units (`solution.w`) that define contributions to the synthetic control unit.

Usage

```
synth(data.prep.obj = NULL,
      X1 = NULL, X0 = NULL,
      Z0 = NULL, Z1 = NULL,
      custom.v = NULL,
      optimxmethod = c("Nelder-Mead", "BFGS"),
      genoud = FALSE, quadopt = "ipop",
      Margin.ipop = 5e-04,
      Sigf.ipop = 5,
      Bound.ipop = 10,
      verbose = FALSE, ...,
      quadopt_inner = NULL,
      quadopt_outer = NULL,
      cvxr_pars = list(),
      cvxr_pars_inner = NULL,
      cvxr_pars_outer = NULL,
      torch_pars = list(),
      torch_pars_inner = NULL,
      torch_pars_outer = NULL)
```

Arguments

<code>data.prep.obj</code>	The object produced by dataprep . This object contains all information about X0, X1, Z1, and Z0. If <code>data.prep.obj</code> is supplied, none of X0, X1, Z1, and Z0 should be manually specified.
<code>X1</code>	Matrix of treated predictor data. Rows correspond to predictors, columns to a single treated unit.
<code>X0</code>	Matrix of control units' predictor data. Rows correspond to predictors, columns to control units (≥ 2).
<code>Z1</code>	Matrix of treated outcome data for the pre-treatment periods over which MSPE is minimized.
<code>Z0</code>	Matrix of control units' outcome data for the pre-treatment periods over which MSPE is minimized.
<code>custom.v</code>	Vector of weights for predictors supplied by the user. Uses <code>synth</code> to bypass optimization for <code>solution.v</code> . See Details.
<code>optimxmethod</code>	Character vector specifying optimization algorithms to be used. Permissible values are all optimization algorithms currently implemented in the function optimx (see that function for details). If multiple algorithms are specified, synth will run all chosen algorithms and return the best-performing result. If "All" is specified, all algorithms in optimx are tried.
<code>genoud</code>	Logical flag. If TRUE, synth performs a two-step optimization. In the first step, genoud is used to obtain a global solution. In the second step, the results are passed to the algorithm(s) chosen in <code>optimxmethod</code> for local optimization. This two-step procedure is slower but may yield lower loss for irregular search spaces.

quadopt	Character specifying the routine for quadratic optimization over W weights. One of "ipop" (default; ipop from kernlab, behavior unchanged from earlier versions of Synth), "cvxr" (CVXR with CLARABEL as the default solver; requires the CVXR package, in <code>Suggests:</code>), or "torch" (Frank-Wolfe simplex least squares via the torch package, also in <code>Suggests:</code>; supports CPU, CUDA, and Apple MPS via <code>torch_pars</code>). The two new backends are opt-in for users with larger panels or who prefer modern convex-optimization or autodiff/GPU machinery; on small panels they reach the same objective value as ipop to within about $1e-3$, the bound the package's tests check. The weights themselves can still differ when several weight vectors fit almost equally well, as in the toy <code>synth.data</code> example below: with the same V (for example with <code>quadopt_outer = "torch"</code>) the torch weights differ from the ipop weights by up to about 0.05, and with <code>quadopt = "torch"</code> the search over V can end at a different V, so the weights can differ by more. When <code>quadopt</code> alone is specified it applies to both the inner and outer QP solves. To use different backends for the two stages — e.g. fast ipop for the V-search loop and a modern solver only for the final W solve — use <code>quadopt_inner</code> and <code>quadopt_outer</code> below.
quadopt_inner	Optional character. If non-NULL, overrides <code>quadopt</code> for the QP solves inside fn.V during the V -search. Defaults to NULL, which means "inherit <code>quadopt</code> ".
quadopt_outer	Optional character. If non-NULL, overrides <code>quadopt</code> for the final W solve once V is fixed. Defaults to NULL, which means "inherit <code>quadopt</code> ". The common pattern is <code>quadopt_outer = "cvxr"</code> (or "torch") with <code>quadopt</code> left at "ipop": V -search runs at ipop's speed and only the final W is solved with the modern backend.
cvxr_pars	Optional named list of tuning parameters forwarded to the CVXR backend when <code>quadopt</code> or <code>quadopt_outer</code> is "cvxr". Recognized fields: <code>solver</code> (default "CLARABEL", an interior-point solver that is a hard CVXR dependency and so ships everywhere CVXR does; other CVXR-supported solvers like "OSQP", "SCS", "ECOS" (requires ECOSolveR), or "MOSEK" (requires a MOSEK license) can be passed), <code>eps</code> (default $1e-8$), and <code>max_iter</code> (default 5000). The generic <code>eps</code> and <code>max_iter</code> are translated to whichever parameter names the chosen solver expects.
torch_pars	Optional named list of tuning parameters forwarded to the torch backend when <code>quadopt</code> or <code>quadopt_outer</code> is "torch". Recognized fields: <code>max_iter</code> (default 500), <code>tol</code> (default $1e-8$), <code>device</code> (default "cpu"; pass "cuda" for NVIDIA GPU or "mps" for Apple Silicon), and <code>dtype</code> ("float64" default or "float32" for speed).
cvxr_pars_inner, cvxr_pars_outer, torch_pars_inner, torch_pars_outer	Optional per-stage overrides for backend tuning. Each defaults to NULL, in which case the matching master list (<code>cvxr_pars</code> or <code>torch_pars</code>) is used. The common pattern is to pass a tighter <code>*_outer</code> configuration (e.g. smaller <code>eps</code> or larger <code>max_iter</code>) for the single final W solve while leaving the V -search at lighter defaults.
Margin.ipop	Setting for the ipop optimization routine: how close we get to the constraints (see ipop for details).

Sigf.ipop	Setting for the ipop optimization routine: precision in significant figures (default: 5; see ipop for details).
Bound.ipop	Setting for the ipop optimization routine: clipping bound for variables (see ipop for details).
verbose	Logical flag. If TRUE, intermediate results are displayed.
...	Further arguments. Currently ignored: they are not passed on to optimx or genoud . The arguments listed after ... in the usage (quadopt_inner to torch_pars_outer) were added in 1.2-0 and have to be named in full; the arguments before it keep their 1.1-10 order.

Details

As proposed in Abadie and Gardeazabal (2003) and Abadie, Diamond, and Hainmueller (2010), [synth](#) searches for the set of weights that generate the best-fitting convex combination of control units. The predictor weight matrix V is chosen among positive definite diagonal matrices such that MSPE is minimized for the pre-intervention period.

Alternatively, the user may supply a vector of V weights based on a subjective assessment of the predictive power of the variables in $X1$ and $X0$. In this case, specify `custom.v` in [synth](#), and the optimization over V matrices is bypassed.

Value

<code>solution.v</code>	Vector of predictor weights.
<code>solution.w</code>	Vector of weights across control units.
<code>loss.v</code>	MSPE from optimization over V and W weights.
<code>loss.w</code>	Loss from optimization over W weights.
<code>custom.v</code>	If specified, returns the user-supplied weight vector.
<code>rgV.optim</code>	Results from <code>optimx()</code> minimization, useful for diagnostics.

Author(s)

Jens Hainmueller and Alexis Diamond

References

- Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science*.
- Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software*, 42(13), 1–17.
- Abadie, A., Diamond, A., Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California’s Tobacco Control Program. *Journal of the American Statistical Association*, 105(490), 493–505.
- Abadie, A., and Gardeazabal, J. (2003). Economic Costs of Conflict: A Case Study of the Basque Country. *American Economic Review*, 93(1), 113–132.

See Also

[dataprep](#), [synth_data](#), [gaps.plot](#), [path.plot](#), [synth.tab](#), [synth_inference](#), [synth_placebos](#)

Examples

```
data(synth.data)
dataprep.out <- dataprep(foo = synth.data,
  predictors = c("X1", "X2", "X3"),
  predictors.op = "mean",
  dependent = "Y",
  unit.variable = "unit.num",
  time.variable = "year",
  special.predictors = list(
    list("Y", 1990, "mean"),
    list("Y", 1985, "mean"),
    list("Y", 1980, "mean")),
  treatment.identifier = 7,
  controls.identifier = c(29, 2, 13, 17, 32, 38),
  time.predictors.prior = 1984:1989,
  time.optimize.ssr = 1984:1990,
  unit.names.variable = "name",
  time.plot = 1984:1996)
synth.out <- synth(dataprep.out)
path.plot(dataprep.res = dataprep.out, synth.res = synth.out)
gaps.plot(dataprep.res = dataprep.out, synth.res = synth.out)

## Not run:
## Alternative quadopt backends (Synth 1.2-0+; both in Suggests:).
## Defaults to ipop. CVXR uses CLARABEL by default (a CVXR Imports,
## so always available); pass cvxr_pars = list(solver = "OSQP"),
## "SCS" or "ECOS" (the latter needs ECOSolveR). Torch uses
## Frank-Wolfe simplex least squares with exact line search and can
## run on GPU/MPS via torch_pars = list(device = "cuda") or "mps".
##
## The calls below reuse the toy dataprep.out from above. On the
## Basque example (see ?dataprep) ipop and torch produce essentially
## the same synthetic control: the same pre-MSPE to four decimals
## (0.0089), with post-MSPE and ATT estimates within 0.001 of each
## other. The CVXR and torch backends are slower because they are
## invoked on every fn.V() evaluation inside optimx's V-search.

# install.packages("CVXR")
synth.cvxr <- synth(dataprep.out, quadopt = "cvxr")

# install.packages("torch"); torch::install_torch()
synth.torch <- synth(dataprep.out, quadopt = "torch")

## To keep V-search fast (ipop) and use a modern solver only for
## the final W solve, set quadopt_outer alone:
synth.fast.cvxr <- synth(dataprep.out, quadopt_outer = "cvxr")

## End(Not run)
```

synth.data	<i>Panel Data to demonstrate the use of the Synthetic Control Method</i>
------------	--

Description

This artificial panel data set is used to demonstrate the use of the Synthetic Control Method.

Usage

```
data(synth.data)
```

Format

A dataframe made up of 8 units: 1 treated (no 7) and 7 control (no. 2,13,17,29,32,36,38) , 3 predictors (X1, X2, X3), 21 time periods (1980 - 2000), a unit.names.variable column ("name") and an outcome variable column (Y). All columns have column names.

synth.tab	<i>Creates Tables that Summarize Results of Synthetic Control Group Method</i>
-----------	--

Description

This function is called after [dataprep](#) and [synth](#) in order to create tables summarizing the results of the run of the synthetic control method. The result tables can be latexed directly.

Usage

```
synth.tab(synth.res      = NA,  
          dataprep.res   = NA,  
          round.digit    = 3)
```

Arguments

- synth.res The list resulting from the call to [synth](#).
- dataprep.res The list resulting from the call to [dataprep](#).
- round.digit Integer for rounding in tables.

Details

NA

Value

tab.pred	The matrix that compares the predictor values of the treated unit, the synthetic control unit, and the sample mean of the control units.
tab.v	The matrix that contains the table of V-weights and respective variable names.
tab.w	The data frame that contains the table of W-weights and respective unit numbers and possibly names.
tab.loss	The matrix that contains the table of W-loss and V-loss

Author(s)

Jens Hainmueller and Alexis Diamond

References

- Abadie, A., Diamond, A., Hainmueller, J. (2014). Comparative Politics and the Synthetic Control Method. *American Journal of Political Science* 59(2): 495-510.
- Abadie, A., Diamond, A., Hainmueller, J. (2011). Synth: An R Package for Synthetic Control Methods in Comparative Case Studies. *Journal of Statistical Software* 42 (13) 1–17.
- Abadie A, Diamond A, Hainmueller J (2010). Synthetic Control Methods for Comparative Case Studies: Estimating the Effect of California’s Tobacco Control Program. *Journal of the American Statistical Association* 105 (490) 493–505.
- Abadie, A. and Gardeazabal, J. (2003) Economic Costs of Conflict: A Case Study of the Basque Country *American Economic Review* 93 (1) 113–132.

See Also

[synth](#), [dataprep](#), [gaps.plot](#), [path.plot](#)

synth_data

Build inputs for synth() from a panel data frame

Description

A friendlier wrapper around [dataprep](#). Takes a long-format panel data frame plus the names of the unit, time, and outcome columns, and returns a dataprep-shaped list ready to pass to [synth](#) and the inference functions.

Defaults:

- controls = NULL uses every panel unit other than the treated one.
- predictors = NULL relies entirely on special_predictors, which must then be non-empty.
- plot_periods = NULL uses the full panel range.
- pre_periods = NULL uses every panel time strictly before treatment_time.

Usage

```
synth_data(panel,
            outcome,
            treated,
            controls = NULL,
            unit_col,
            time_col,
            treatment_time,
            predictors = NULL,
            predictors.op = "mean",
            special_predictors = list(),
            pre_periods = NULL,
            plot_periods = NULL,
            unit_names_col = NULL)
```

Arguments

panel	Long-format data.frame with one row per (unit, time).
outcome	Name of the outcome column (character).
treated	The treated unit. Either a numeric id (matched against unit_col) or a character name (requires unit_names_col).
controls	Optional vector of control units. Same id-or-name convention as treated. Defaults to "all other units in the panel".
unit_col	Name of the unit-id column (character). Must be numeric.
time_col	Name of the time column (character). Must be numeric.
treatment_time	Single numeric value: the first post-treatment period. Used as the SSR-window cutoff and as the default treatment date for downstream inference.
predictors	Character vector of predictor column names. NULL means rely entirely on special_predictors.
predictors.op	Name of the function used to aggregate predictors over pre_periods, for the treated and the control units alike. Default "mean". See dataprep for which functions can be used.
special_predictors	List of list(name, periods, op) triples passed to dataprep .
pre_periods	Optional integer vector of pre-treatment time values. Defaults to every time_col value strictly before treatment_time.
plot_periods	Optional integer vector of times for the full plot horizon. Defaults to the full panel range.
unit_names_col	Optional column name carrying readable unit labels. Required when treated or controls are passed as names.

Value

A dataprep-shaped list (the same structure returned by [dataprep](#)). The list also carries tag\$synth_data_treatment_time and an attr(., "synth_data_call") for debugging.

See Also

[dataprep](#) for the long-form constructor; [synth](#), [synth_inference](#), [synth_placebos](#) for what to do next.

Examples

```
## Not run:
data(basque)
# A simplified variant of the Basque example in ?dataprep, in one call:
dp <- synth_data(
  panel          = basque,
  outcome        = "gdpcap",
  unit_col       = "regionno",
  time_col       = "year",
  treated        = 17,                # Basque country
  controls       = c(2:16, 18),
  treatment_time = 1970,
  predictors      = c("school.illit", "school.prim", "school.med",
                     "school.high", "school.post.high", "invest"),
  special_predictors = list(
    list("gdpcap", 1960:1969, "mean"),
    list("sec.agriculture", seq(1961, 1969, 2), "mean")
  ),
  unit_names_col  = "regionname"
)
fit <- synth(dp)
inf <- synth_inference(fit, dp, method = "conformal", alpha = 0.10)

## End(Not run)
```

synth_inference	<i>Prediction intervals for synthetic control counterfactuals</i>
-----------------	---

Description

Computes a prediction band around the synthetic counterfactual produced by [synth](#) and [dataprep](#). Two methods are supported: split-conformal intervals (the default) and parametric Gaussian intervals.

Usage

```
synth_inference(synth.res = NULL,
                dataprep.res = NULL,
                method = c("conformal", "parametric"),
                alpha = 0.05,
                treatment_time = NULL)
```

Arguments

synth.res	Output list from synth .
dataprep.res	Output list from dataprep .
method	One of "conformal" (default) or "parametric". See Details.
alpha	Miscoverage level. The band targets nominal $(1 - \alpha)$ coverage. Default 0.05.
treatment_time	Optional first post-treatment period (a value of <code>dataprep.res\$tag\$time.plot</code>). If NULL (default), the first post-treatment period is the <code>treatment_time</code> stored by synth_data when <code>dataprep.res</code> was built with it, and <code>max(time.optimize.ssr) + 1</code> otherwise. The post window is <code>time.plot >= treatment_time</code> ; pre-treatment plot years <i>before</i> the SSR optimization window (e.g. 1955–1959 in a typical Basque setup with <code>time.optimize.ssr = 1960:1969</code>) are pre-treatment and are excluded from post-period MSPE summaries.

Details

With `method = "conformal"`, the half-width of the band is the order statistic at rank $k = \lceil (n + 1)(1 - \alpha) \rceil$ of the absolute pre-treatment residuals $|\hat{Y}_{1,t} - \hat{Y}_{0,t}\hat{w}|$ (Chernozhukov, Wuthrich, and Zhu 2021). The $(n + 1)$ adjustment delivers exact $(1 - \alpha)$ finite-sample coverage under exchangeability. When $k > n$ the requested level is infeasible at this calibration sample size, the function emits a warning, and `conformal_q` is Inf.

With `method = "parametric"`, the half-width is `qnorm(1 - alpha/2)` times the standard deviation of pre-period residuals. This assumes residuals are i.i.d. Gaussian.

Both methods produce constant-width bands. They do *not* separately quantify uncertainty about the synthetic weights $\hat{w}$ or decompose in-sample versus out-of-sample uncertainty. Users who need period-varying intervals or that decomposition should see the **scpi** package.

Validity in both methods is approximate when outcomes are autocorrelated.

Value

An object of S3 class `c("synth_<method>", "synth_inference")`, a list with components:

method	The method used (echoed).
alpha	Miscoverage level (echoed).
time	The full plot horizon (<code>dataprep.res\$tag\$time.plot</code>).
pre_idx, post_idx	Integer indices into time for pre- and post-treatment periods.
treated, synthetic	Numeric vectors over time: observed treated outcome and synthetic counterfactual.
effect	<code>treated - synthetic</code> .
intervals	Numeric matrix with lower and upper columns for each period in time.
conformal_q	(conformal only) The $(1 - \alpha)$ -quantile of $ \text{gap}_{\text{pre}} $.
sigma_pre	(parametric only) Standard deviation of pre-period residuals.
pre_mspe, post_mspe, mspe_ratio	Mean squared prediction error in the pre- and post-treatment periods, and their ratio.

Author(s)

Jens Hainmueller and Alexis Diamond

References

Chernozhukov, V., Wuthrich, K., and Zhu, Y. (2021). An exact and robust conformal inference method for counterfactual and synthetic controls. *Journal of the American Statistical Association* 116 (536) 1849–1864.

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies. *Journal of the American Statistical Association* 105 (490) 493–505.

Cattaneo, M. D., Feng, Y., Palomba, F., and Titiunik, R. (2025). Uncertainty quantification in synthetic controls with staggered treatment adoption. *Review of Economics and Statistics*.

See Also

[synth](#), [dataprep](#), [plot.synth_inference](#), [synth_placebos](#). The **SCtools** package provides additional placebo-based inference; the **scpi** package provides period-varying CFPT prediction intervals that decompose in-sample and out-of-sample uncertainty.

Examples

```
## Not run:
data(synth.data)
d <- dataprep(
  foo = synth.data,
  predictors = c("X1", "X2", "X3"),
  predictors.op = "mean",
  dependent = "Y",
  unit.variable = "unit.num",
  time.variable = "year",
  treatment.identifier = 7,
  controls.identifier = c(29, 2, 13, 17, 32, 38),
  time.predictors.prior = 1984:1989,
  time.optimize.ssr = 1984:1990,
  unit.names.variable = "name",
  time.plot = 1984:1996
)
fit <- synth(d)
inf <- synth_inference(fit, d, method = "conformal", alpha = 0.20)
print(inf)
plot(inf)

## End(Not run)
```

synth_mspe_plot	<i>Plot of placebo post/pre MSPE ratios</i>
-----------------	---

Description

Plots the distribution of post/pre MSPE ratios across placebos with the treated unit highlighted. Operates on the output of [synth_placebos](#).

Usage

```
synth_mspe_plot(placebos,
                 Main = "Post/Pre MSPE Ratio",
                 Xlab = "MSPE ratio",
                 Ylab = "")
```

Arguments

placebos	Output of synth_placebos .
Main	Optional main title.
Xlab	X axis label.
Ylab	Y axis label.

Value

Invisibly returns NULL.

See Also

[synth_placebos](#), [synth_mspe_test](#), [plot.synth_placebos](#).

synth_mspe_test	<i>One-sided placebo p-value via post/pre MSPE ratio</i>
-----------------	--

Description

Computes a one-sided p-value for the treated unit's post/pre mean squared prediction error (MSPE) ratio against the empirical distribution of placebo ratios. Operates on the output of [synth_placebos](#).

Usage

```
synth_mspe_test(placebos)
```

Arguments

placebos	Output of synth_placebos .
----------	--

Details

The p-value is the empirical rank

$$p = \text{mean}(\{r_{\text{treated}}\} \cup \{r_i : r_i \text{ valid}\} \geq r_{\text{treated}}),$$

where r_{treated} is the treated unit's post/pre MSPE ratio and r_i is the corresponding ratio for placebo donor i . Failed refits are excluded from the denominator.

Value

A list with elements

`mspe_ratio_treated`

The treated unit's post/pre MSPE ratio.

`mspe_ratios_placebos`

Numeric vector of placebo ratios, with NA for failed refits.

`pvalue`

One-sided empirical p-value.

`n_valid_placebos`

Count of placebos with non-missing ratios.

References

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies. *Journal of the American Statistical Association* 105 (490) 493–505.

See Also

[synth_placebos](#), [synth_mspe_plot](#), [plot.synth_placebos](#).

`synth_placebos`

In-space placebo refits for synthetic control inference

Description

For each donor in the original control pool, swap that donor into the treated slot, refit [synth](#), and record the resulting gap series. The returned object feeds [synth_mspe_test](#), [synth_mspe_plot](#), and the [plot](#) method.

Usage

```
synth_placebos(synth.res = NULL,
               dataprep.res = NULL,
               Sigf.ipop = 5,
               Margin.ipop = 0.0005,
               Bound.ipop = 10,
               optimxmethod = c("Nelder-Mead", "BFGS"),
               genoud = FALSE,
```

```

custom.v = NULL,
verbose = FALSE,
parallel = FALSE,
n_cores = NULL,
quadopt = "ipop",
quadopt_inner = NULL,
quadopt_outer = NULL,
cvxr_pars = list(),
cvxr_pars_inner = NULL,
cvxr_pars_outer = NULL,
torch_pars = list(),
torch_pars_inner = NULL,
torch_pars_outer = NULL,
treatment_time = NULL,
keep_fits = FALSE)

```

Arguments

synth.res	Output list from synth for the real treated unit. Used only for the reference summary; placebos are refit from dataprep.res.
dataprep.res	Output list from dataprep .
Sigf.ipop, Margin.ipop, Bound.ipop, optimxmethod, custom.v, verbose	Passed through to each synth placebo refit. Match these to the configuration that produced synth.res for like-for-like inference.
parallel	Parallelization mode. One of FALSE / "none" (serial; default), TRUE / "auto" (forks via mclapply on unix-likes; PSOCK cluster on Windows), "multicore" (forces mclapply; errors on Windows), or "snow" (forces a PSOCK cluster via makePSOCKcluster + parLapply ; works on all platforms).
n_cores	Number of cores when parallel = TRUE. Defaults to one less than the number of physical cores.
quadopt, quadopt_inner, quadopt_outer, cvxr_pars, cvxr_pars_inner, cvxr_pars_outer, torch_pars, torch_pars_inner, torch_pars_outer	Forwarded to each placebo refit's synth call. Match these to the configuration that produced synth.res for like-for-like inference. See synth for the inner/outer split semantics.
genoud	Forwarded to each placebo refit's synth call. Set to TRUE when the original fit used genoud, so placebos use the same optimizer.
treatment_time	Optional first post-treatment period. Defaults to the treatment_time stored by synth_data when dataprep.res was built with it, otherwise to max(dataprep.res\$tag\$time.optimiz + 1. Pre-treatment plot years before the SSR optimization window are excluded from post-period MSPE so they cannot inflate placebo p-values.
keep_fits	If TRUE, store the full synth return value (with solution.w, solution.v, etc.) on each placebo entry under the fit field. Default FALSE keeps the output light by storing only the gap series and MSPE summaries.

Details

For donor i , the swap moves donor i 's columns from X_0 , Z_0 , and Y_0 into the treated slots, and the original treated unit takes donor i 's former column in the control pool. The donor pool size is preserved.

Refits that error are caught with `tryCatch`; the corresponding failed flag is set and downstream functions exclude them from denominators.

Value

An object of class `synth_placebos`, a list with components:

<code>treated</code>	Named list with the real fit's gap series and MSPE summaries (<code>gap</code> , <code>pre_mspe</code> , <code>post_mspe</code> , <code>mspe_ratio</code>).
<code>placebos</code>	Named list of length <code>ncol(dataprep.res[X0])</code> ; each element has the same shape as <code>treated</code> plus a failed flag and an <code>error_message</code> (NA on success, the captured <code>conditionMessage(e)</code> on <code>tryCatch</code> failure). When <code>keep_fits = TRUE</code> the entry also has a <code>fit</code> field carrying the full synth return value for that donor.
<code>time</code>	Plot horizon, equal to <code>dataprep.res\$tag\$time.plot</code> .
<code>pre_idx</code> , <code>post_idx</code>	Integer indices into <code>time</code> .
<code>treatment_time</code>	First post-treatment period, used to define <code>post_idx</code> .
<code>donor_names</code>	Character vector of donor labels.
<code>failed</code>	Logical vector aligned with <code>donor_names</code> .

Note

Development versions of **Synth** 1.2-0 called these functions `generate_placebos`, `mspe_test`, `mspe_plot`, and `plot_placebos`. Those are also the names of functions in the **SCtools** package, which take different arguments, so attaching both packages made one set mask the other. The functions here are now `synth_placebos`, `synth_mspe_test`, `synth_mspe_plot`, and the `plot` method for `synth_placebos` objects.

Author(s)

Jens Hainmueller and Alexis Diamond

References

Abadie, A., Diamond, A., and Hainmueller, J. (2010). Synthetic Control Methods for Comparative Case Studies. *Journal of the American Statistical Association* 105 (490) 493–505.

See Also

[synth_mspe_test](#), [synth_mspe_plot](#), [plot.synth_placebos](#), [synth_inference](#).

Examples

```
## Not run:
data(synth.data)
d <- dataprep(
  foo = synth.data,
  predictors = c("X1", "X2", "X3"),
  predictors.op = "mean",
  dependent = "Y",
  unit.variable = "unit.num",
  time.variable = "year",
  treatment.identifier = 7,
  controls.identifier = c(29, 2, 13, 17, 32, 38),
  time.predictors.prior = 1984:1989,
  time.optimize.ssr = 1984:1990,
  unit.names.variable = "name",
  time.plot = 1984:1996
)
fit <- synth(d)
pl <- synth_placebos(fit, d)
print(pl)
synth_mspe_test(pl)$pvalue
plot(pl)

## End(Not run)
```

Index

- * **datasets**
 - smoking, [18](#)
 - synth.data, [26](#)
- synth_mspe_plot, [18](#), [32](#), [33](#), [35](#)
- synth_mspe_test, [18](#), [32](#), [32](#), [33](#), [35](#)
- synth_placebos, [17](#), [18](#), [25](#), [29](#), [31](#), [32](#), [33](#), [33](#)
- basque, [2](#), [19](#)
- collect.optimx, [4](#)
- dataprep, [5](#), [5](#), [13–15](#), [19–22](#), [25–31](#), [34](#)
- fn.V, [11](#), [23](#)
- gaps.plot, [6](#), [7](#), [13](#), [13](#), [15](#), [21](#), [25](#), [27](#)
- genoud, [22](#), [24](#)
- ipop, [12](#), [23](#), [24](#)
- legend, [16](#)
- makePSOCKcluster, [34](#)
- mclapply, [34](#)
- optimx, [4](#), [22](#), [24](#)
- parLapply, [34](#)
- path.plot, [6](#), [7](#), [13](#), [14](#), [14](#), [21](#), [25](#), [27](#)
- plot, [33](#)
- plot.synth_inference, [16](#), [31](#)
- plot.synth_placebos, [17](#), [32](#), [33](#), [35](#)
- print.synth_inference
 - (synth_inference), [29](#)
- print.synth_placebos (synth_placebos),
[33](#)
- smoking, [18](#)
- spec.pred.func, [19](#)
- synth, [5](#), [7](#), [11–15](#), [19](#), [21](#), [21](#), [22](#), [24](#), [26](#), [27](#),
[29–31](#), [33–35](#)
- synth.data, [26](#)
- synth.tab, [7](#), [13–15](#), [21](#), [25](#), [26](#)
- synth_data, [25](#), [27](#), [30](#), [34](#)
- synth_inference, [16](#), [19](#), [25](#), [29](#), [29](#), [35](#)