Package: bscm (via r-universe)

October 8, 2026

Title Bayesian Synthetic Control Models

Version 1.0.1

Description Implements the synthetic control method of Abadie, Diamond, and Hainmueller (2010) <doi:10.1198/jasa.2009.ap08746> within a Bayesian framework, enabling straightforward uncertainty quantification of treatment effects and other quantities of interest. Supports time-varying covariates with potentially time-varying effects, single or multiple treated units, and staggered treatment adoption. Provides methods for model assessment, comparison, and selection based on placebo studies, cross-validation, and posterior predictive checks. Posterior sampling is performed using Markov chain Monte Carlo via Stan.

License GPL (>= 3)

Encoding UTF-8

RoxygenNote 7.3.3

Biarch true

Depends R (>= 4.1.0)

Imports checkmate, cli, dplyr, ggplot2, loo, methods, posterior, progressr, projpred, quadprog, Rcpp (>= 0.12.0), RcppParallel (>= 5.0.1), rlang, rstan (>= 2.32.7), rstantools (>= 2.6.0), tidyr

LinkingTo BH (>= 1.66.0), Rcpp (>= 0.12.0), RcppEigen (>= 0.3.3.3.0), RcppParallel (>= 5.0.1), rstan (>= 2.32.7), StanHeaders (>= 2.32.10)

SystemRequirements GNU make

LazyData true

LazyDataCompression xz

URL https://github.com/helske/bscm

BugReports https://github.com/helske/bscm/issues

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

NeedsCompilation yes

Author Jouni Helske [aut, cre] (ORCID:
<https://orcid.org/0000-0001-7130-793X>)

Maintainer Jouni Helske <jouni.helske@iki.fi>

Config/pak/sysreqs make

Repository https://cran.r-universe.dev

Date/Publication 2026-08-09 06:40:02 UTC

RemoteUrl https://github.com/cran/bscm

RemoteRef HEAD

RemoteSha 89d2cec0d4a8333d6d8f4504669f035e40ecb6d2

Contents

bscm-package . . . 3
as.data.frame.bscmfit . . . 4
as_draws.bscmfit . . . 5
average_treatment_effect . . . 6
bscm . . . 7
bscm_prior . . . 9
check_mcmc_diagnostics . . . 11
coef.bscmfit . . . 12
covariate_adjustment . . . 13
covariate_imbalance . . . 14
donor_ranking . . . 15
donor_weights . . . 16
effective_donors . . . 16
fit_single_treated . . . 17
fitted.bscmfit . . . 18
get_priors . . . 19
get_refmodel.bscmfit . . . 19
get_standata . . . 20
get_stanfit . . . 21
leave_donor_out . . . 21
lfo . . . 23
log_lik.bscmfit . . . 24
loo.bscmfit . . . 25
loo_R2.bscmfit . . . 26
multiple_treated . . . 27
nchains.bscmfit . . . 27
ndraws.bscmfit . . . 28
placebo_effects . . . 28
plot.bscm_lfo . . . 29
plot.bscmfit . . . 30
plot_coefs . . . 31

plot_effects 32
plot_residuals 33
plot_weights 34
posterior_epred.bscmfit 35
posterior_linpred.bscmfit 36
posterior_predict.bscmfit 37
print.bscm_lfo 37
print.bscmfit 38
print.bscmfit_diagnostics 38
proj_predict_bscm 39
residual_acf 39
residuals.bscmfit 40
rmse 41
rmse_ratio 42
sigma.bscmfit 43
single_treated 43
summary.bscmfit 44
synthetic_control 45
treatment_effect 45
tv 46

Index **48**

`bscm-package` *The 'bscm' package*

Description

Implements the synthetic control method of Abadie, Diamond, and Hainmueller (2010) within a Bayesian framework, enabling straightforward uncertainty quantification of treatment effects and other quantities of interest. Posterior sampling is performed using Markov chain Monte Carlo via Stan.

`bscm` supports time-varying covariates with potentially time-varying effects, single treated unit as well as multiple treated units possibly with staggered treatment adoption.

Output of `bscm()` is a `bscmfit` object with methods for model assessment, comparison, and selection based on placebo studies, cross-validation, and posterior predictive checks.

Author(s)

Maintainer: Jouni Helske <jouni.helske@iki.fi> (ORCID)

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, doi:10.1198/jasa.2009.ap08746.

Stan Development Team (2025). RStan: the R interface to Stan. R package version 2.32.7. https://mc-stan.org.

See Also

Useful links:

- https://github.com/helske/bscm
- Report bugs at https://github.com/helske/bscm/issues

`as.data.frame.bscmfit` *Extract posterior draws of model parameters as a data frame*

Description

Returns a `data.frame` representation of the posterior sample of the model parameters. For samples from posterior predictive distribution, see `posterior_epred()` and `posterior_predict()`. For donor weights, use `donor_weights()`.

Usage

```
## S3 method for class 'bscmfit'
as.data.frame(
  x,
  row.names = NULL,
  optional = FALSE,
  parameters = NULL,
  include = TRUE,
  ...
)
```

Arguments

`x`	[`bscmfit`] The model fit object.
`row.names`	Ignored.
`optional`	Ignored.
`parameters`	[`character()`] Vector of parameter names. When `NULL`, (the default), corresponds to a relevant subset of `c("alpha", "beta", "sigma", "kappa", "rho")`. Other possible choices are `"gamma"` (time-varying regression coefficients) and `"lp__"` (log-posterior values without constants).
`include`	[`logical(1)`] If `TRUE` (the default), output includes only the variables defined by the argument `parameters`. If `FALSE`, these variables are excluded from the output. If `NULL`, same as `TRUE` but variables not present in the model object are silently ignored (whereas `TRUE` throws an error).
`...`	Ignored.

Value

A data.frame containing model parameters in a wide format.

See Also

as_draws.bscmfit().

Examples

```
head(as.data.frame(fit_single_treated, parameters = c("alpha", "sigma")))
```

---

as_draws.bscmfit *Return posterior draws from BSCM fit as a draws object*

Description

Return posterior draws from BSCM fit as a draws object

Usage

```
## S3 method for class 'bscmfit'
as_draws(x, parameters = NULL, inc_warmup = FALSE, include = TRUE, ...)
```

Arguments

x	[bscmfit] The model fit object.
parameters	[character()] Vector of parameter names. When NULL, (the default), corresponds to a relevant subset of c("alpha", "beta", "sigma", "kappa", "rho"). Other possible choices are "gamma" (time-varying regression coefficients) and "lp__ (log-posterior values without constants).
inc_warmup	[logical(1)] Whether to include warmup draws. Default is FALSE.
include	[logical(1)] If TRUE (the default), output includes only the variables defined by the argument parameters. If FALSE, these variables are excluded from the output. If NULL, same as TRUE but variables not present in the model object are silently ignored (whereas TRUE throws an error).
...	Ignored.

Value

An object of class draws_array containing the posterior draws of the specified parameters.

See Also

posterior::as_draws() for converting the output to other formats, and posterior::summarise_draws() for computing posterior summaries of the draws.

Examples

```
head(as_draws(fit_single_treated, parameters = c("alpha", "sigma")))
```

`average_treatment_effect` *Pre- and post-treatment average effects of a Bayesian synthetic control model*

Description

Pre- and post-treatment average effects of a Bayesian synthetic control model

Usage

```
average_treatment_effect(x, ...)

## S3 method for class 'bscmfit'
average_treatment_effect(
  x,
  average = TRUE,
  summary = TRUE,
  probs = c(0.025, 0.975),
  ...
)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
average	[logical(1)] If TRUE (the default), returns the average treatment effect over treated units at each time since treatment (event time). If FALSE, unit-specific effects at each calendar time point are returned.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A `tibble` of posterior summaries (`summary = TRUE`) or posterior draws (`summary = FALSE`) in long format.

Examples

```
average_treatment_effect(fit_single_treated)
```

`bscm` *Estimate Bayesian synthetic control model*

Description

Function `bscm` estimates a Bayesian synthetic control models of varying types.

Usage

```
bscm(
  formula,
  data,
  treatment = "treatment",
  time = "time",
  unit = "id",
  error = "iid",
  priors = NULL,
  prior_only = FALSE,
  mcmc_diagnostics = TRUE,
  save_data = TRUE,
  compute_predictions = TRUE,
  ...
)
```

Arguments

`formula`	[`formula`] The model formula containing the outcome variable on the left-hand side and optional time-varying predictors on the right-hand side (RHS) of `~`. See details.
`data`	[`data.frame` or an object coercible to one] The long format data that contains the model variables.
`treatment`	[`character(1)`] Name of the treatment indicator variable in `data`. Default is `"treatment"`.
`time`	[`character(1)`] Name of the time index variable in `data`. Default is `"time"`.
`unit`	[`character(1)`] Name of the variable in `data` identifying different units. Default is `"id"`.

`error` [character(1)]
Assumed error structure of the model. Either "iid" (independent and identically distributed errors) or "ar1" (first-order autoregressive process). Default is "iid".

`priors` [list()]
List of prior definitions for the model parameters. See `bscm_prior()` for details on the available prior families and how to define them. If NULL (the default), default priors are used.

`prior_only` [logical(1)]
If TRUE, samples from prior predictive distribution. Default is FALSE.

`mcmc_diagnostics` [logical(1)]
If TRUE (the default), the output of `bscm()` includes the results of MCMC diagnostics checks performed by `check_mcmc_diagnostics.bscmfit()`. Note that regardless of the value of mcmc_diagnostics, `rstan::sampling()` can still generate warnings regarding convergence and other sampling issues.

`save_data` [logical(1)]
If TRUE (the default), bscmfit object returned by `bscm()` includes the input data.frame (argument data), after dropping unused factor levels and potentially rearranging data by unit and time variables.

`compute_predictions` [logical(1)]
If TRUE (the default), posterior predictive draws (y_rep) are computed during sampling. Set to FALSE to skip this computation and reduce memory usage; this is used internally by `lfo()` when refitting the model repeatedly. Note that setting this to FALSE will cause `treatment_effect()`, `synthetic_control()`, `posterior_predict()`, `rmse()`, `summary()`, and other methods that rely on posterior predictions to fail, so you rarely want to set this to FALSE.

`...` Additional parameters passed on to `rstan::sampling()` to adjust the sampling options, for example iter and chains. Note that defaults iter = 5000 and warmup = 2500 differ from the defaults of `rstan::sampling()` (which are 2000 and 1000 respectively). Many other control arguments for Stan can be passed using a named list control, such as control = list(adapt_delta = 0.95) which corresponds to the default adapt_delta value of bscm (while rstan default is 0.8).

Details

To define formula in case of no predictors, use outcome ~ 1 or outcome ~ 0. In the former case, as well by the default when using predictors (e.g., outcome ~ x + z), the model includes intercept term for each treated unit. Intercept can be omitted by using 0 in the RHS, e.g., outcome ~ 0 or outcome ~ 0 + x + z (equivalently, you can use -1 in place of 0). Formula should not contain the variable defining the treatment, which is defined separately using the argument treatment. In case the variable is present also in the formula, it is automatically removed.

To specify predictors with time-varying coefficients in formula, wrap them in tv(), e.g., outcome ~ x + tv(~ z, df = 10, type = "rw1") defines a model where x has a time-constant coefficient and z has a time-varying coefficient following penalized cubic spline with 10 spline basis functions and

random walk prior on the spline coefficients. Terms inside `tv()` are automatically also included in the time-constant part of the model, since the time-varying coefficients are defined to have zero mean in the pre-treatment period (in case of multiple treated units, minimum period). Note that a common time-varying intercept is omitted as it would cancel out in the linear predictor.

Both the time-constant and time-varying regression part is assumed to apply for all treated and donor units with common coefficients. If you want to apply a covariate only for treated units, just set the covariate values to zero for donors. This approach can be also used to vary behaviour of intercept: Defining formula such as `y ~ 0 + intercept + x`, where `intercept` is a name of constant column in the data, will define common intercept for all units, instead of unit-specific intercepts (e.g., fixed effects).

When model contains covariates X, their effect is subtracted from donors, i.e., for treated unit i, $y_i \sim N(\alpha_i + X_i\beta + Z^*\omega_i, \sigma_i^2)$, where column j of Z^* is $z_j - (\alpha_j + X_j\beta)$, i.e., the original donor outcome vector z_j minus the donor-specific intercept α_j and the effect of covariates $X_j\beta$ for that donor. Note that β is common across donors and the treated units.

Model can contain missing values in the outcome variable of the treated, but not of the donors, nor in the covariates. Missing outcomes are automatically imputed during MCMC sampling under a missing at random (MAR) assumption.

Value

An object of class `bscmfit`.

See Also

`summary.bscmfit()`, `as_draws.bscmfit()`, `rstan::sampling()`.

Examples

```
# skip diagnostics and use small number of iterations for CRAN checks
fit <- bscm(
  y ~ 1, single_treated, "treatment", "time", "id",
  priors = list(omega = dirichlet_pr(0.5)),
  chains = 1, cores = 1, refresh = 0, iter = 1000,
  mcmc_diagnostics = FALSE
)
fit
```

---

bscm_prior *Prior distributions in the Bayesian synthetic control model*

Description

The functions `normal_pr()`, `student_pr()`, `exponential_pr()`, `gamma_pr()`, `beta_pr()`, `half_normal_pr()`, `dirichlet_pr()`, and `logistic_normal_pr()` specify prior distributions for parameters of the Bayesian synthetic control model estimated by `bscm()`.

Usage

```
normal_pr(location, scale)

student_pr(df, location, scale)

gamma_pr(shape, rate)

exponential_pr(rate)

beta_pr(shape1, shape2)

half_normal_pr(scale)

dirichlet_pr(concentration)

logistic_normal_pr(scale)
```

Arguments

location	The location parameter for the prior.
scale	The scale parameter for the prior.
df	The degrees of freedom for the Student's t prior.
shape	The shape parameter for the gamma prior.
rate	The rate parameter for the gamma and exponential priors.
shape1	The first shape parameter for the beta prior.
shape2	The second shape parameter for the beta prior.
concentration	The concentration parameter for the Dirichlet prior.

Details

For parameters that can take any real value (alpha, beta), currently supported prior distributions are normal_pr(location, scale) and student_pr(df, location, scale). Note however that for the intercept the prior corresponds to a value of the intercept when donors and predictors corresponding to time-constant coefficients are centered.

For parameters constrained to be positive (sigma, kappa), supported distributions are exponential_pr(rate), gamma_pr(shape, rate), and half_normal_pr(scale).

For autoregressive coefficients of the residuals (rho) the prior corresponds to transformation 0.5 * (1 + rho), so that the support is (0, 1). Only supported prior distribution for this is beta_pr(shape1, shape2).

For the donor weight vector omega, supported priors are dirichlet_pr(concentration) and logistic_normal_pr(scale). The (symmetric) Dirichlet prior with concentration α_ω defined as $\omega \sim \text{Dirichlet}(\alpha_\omega, \ldots, \alpha_\omega)$. Values $\alpha_\omega < 1$ concentrate weight on few donors, while $\alpha_\omega = 1$ is uniform over the simplex, and $\alpha_\omega > 1$ pulls weights toward the center of the simplex. Logistic normal with with scale σ_α is defined as $\omega = \text{softmax}(\eta)$, where $\eta \sim N(0, \sigma_\alpha^2 I)$ constrained to sum to zero. Larger scale induces more concentrated (sparser) weights.

Value

A bscm_prior object.

See Also

bscm()

Examples

```
normal_pr(c(0, 0), c(1, 2))
student_pr(df = 4, location = 0, scale = 2)
exponential_pr(rate = c(1, 0.1))
gamma_pr(2, 1)
beta_pr(2, 2)
half_normal_pr(1)
dirichlet_pr(1)
logistic_normal_pr(2)
```

---

check_mcmc_diagnostics

Check the validity of the posterior of bscmfit *object*

Description

This function is automatically called at the end of bscm() to check that the output can be trusted in terms of convergence of the MCMC sampling. Checks consists of the common diagnostics of Hamiltonian Monte Carlo variant used by Stan, as well as the Rhat values and effective sample sizes of model parameters. See rstan::check_hmc_diagnostics() and posterior::default_convergence_measures() for details on the definitions of these.

Usage

```
check_mcmc_diagnostics(x, ...)

## S3 method for class 'bscmfit'
check_mcmc_diagnostics(x, warn = TRUE, ...)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
warn	[logical(1)] If TRUE (the default), generates and (typically) prints out a warning in a case of problematic results. Setting this to FALSE silently returns the check results.

Value

Invisibly returns a list containing the results of the check.

References

https://mc-stan.org/learn-stan/diagnostics-warnings.html

Examples

```
check_mcmc_diagnostics(fit_single_treated, warn = FALSE)
```

`coef.bscmfit` *Extract regression coefficients of a Bayesian synthetic control model*

Description

Extract regression coefficients of a Bayesian synthetic control model

Usage

```
## S3 method for class 'bscmfit'
coef(object, type = NULL, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

`object`	[`bscmfit`] The model fit object.
`type`	[`character()`] Type of coefficients to return. Should be one or more of `"alpha"` (intercepts), `"beta"` (regression coefficients), `"gamma"` (time-varying regression coefficients), `"kappa"` (SDs of time-varying coefficients), and `"rho"` (autoregressive coefficients of the residuals). The default `NULL` corresponds to all terms above contained in the model.
`summary`	[`logical(1)`] If `TRUE` (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the `probs` argument), and MCMC convergence measures. If `FALSE`, returns the posterior draws instead.
`probs`	[`numeric()`] Probabilities for quantile summaries. Default is `c(0.025, 0.975)`.
`...`	Ignored.

Value

A `tibble` of posterior summaries (`summary = TRUE`) or list of `tibbles` of posterior draws (`summary = FALSE`).

Examples

```
coef(fit_single_treated)
coef(fit_single_treated, type = "beta")
```

`covariate_adjustment` *Covariate adjustments for Bayesian synthetic control model*

Description

Computes the posterior distribution of the covariate adjustment to the synthetic control for each predictor, defined as the regression coefficient multiplied by the difference between the predictor value of the treated unit and the corresponding weighted predictor value of the donor units.

Usage

```
covariate_adjustment(x, ...)

## S3 method for class 'bscmfit'
covariate_adjustment(
  x,
  plot = TRUE,
  probs = c(0.025, 0.975),
  alpha = 0.5,
  scales = "free_y",
  ...
)
```

Arguments

`x`	[`bscmfit`] An object of class `bscmfit`.
`...`	Ignored.
`plot`	[`logical(1)`] If `TRUE` (the default), plot the posterior covariate adjustments.
`probs`	[`numeric(2)`] Vector of length two defining the limits of the posterior interval. Default is `c(0.025, 0.975)`.
`alpha`	[`numeric(1)`] Opacity of the credible-interval ribbon (used only when `type = "varying"`). Default is `0.5`.
`scales`	[`character(1)`] Passed to `ggplot2::facet_wrap()` (used only when `type = "varying"`). Default is `"free_y"`.

Details

For predictor k, treated unit i, and time point t, the covariate adjustment is

$$b_{tk}\left(X_{y,itk} - \sum_j \omega_{ji} X_{z,jtk}\right),$$

where $b_{tk} = \beta_k$ for predictors with a fixed coefficient and $b_{tk} = \beta_k + \gamma_{tk}$ for predictors with a time-varying coefficient.

If there are multiple treated units, the covariate adjustments are averaged over treated units within each posterior draw before computing posterior summaries.

Value

A data frame containing the posterior mean and requested posterior quantiles of the covariate adjustment for each predictor and time point. If `plot = TRUE`, the corresponding plot is printed as a side effect.

`covariate_imbalance` *Covariate imbalance of Bayesian synthetic control model*

Description

For models with covariates, returns and optionally visualizes the covariate imbalances

$$\delta_t = \sqrt{\frac{1}{K}\sum_{k=1}^{K}(x_{k,0,t} - \bar{x}_{k,0,t})^2},$$

$t = 1, \ldots, T$, where $\bar{x}_{k,0,t} = \sum_{j=1}^{J} \omega_j x_{k,j,t}$ and $x_{k,0,t}$ is the value of kth covariate of a treated unit at time t, and similarly for donors $j = 1, \ldots, J$. This is computed separately for each treated unit in case of multiple treated units.

Usage

```
covariate_imbalance(x, ...)

## S3 method for class 'bscmfit'
covariate_imbalance(x, plot = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

`x`	[`bscmfit`] The model fit object.
`...`	Optional arguments passed to `ggplot2::facet_wrap()`.

plot [logical(1)]
If TRUE (the default), plots the posterior mean and interval of the synthetic covariate distances over time.

probs [numeric()]
Probabilities for quantile summaries. Default is c(0.025, 0.975). If length of probs less than 2, no posterior intervals are drawn, and if length of probs is larger than two, the most extreme values are used for the posterior intervals.

Value

A data.frame of posterior summaries of synthetic covariate distances.

Examples

```
covariate_imbalance(fit_single_treated, plot = TRUE, probs = c(0.05, 0.95))
```

---

donor_ranking *Donor ranking*

Description

Returns ranking of donors either using posterior means of donor weights from bscmfit object, or ranking from a vsel object returned by projpred::varsel() or projpred::cv_varsel() when applied to a bscmfit object.

Usage

```
donor_ranking(x)
```

Arguments

x [vsel] or [bscmfit]
Output from bscm(), projpred::varsel() or projpred::cv_varsel().

Value

Character vector of donor IDs in order of selection.

donor_weights *Extract donor weights of a Bayesian synthetic control model*

Description

Extract donor weights of a Bayesian synthetic control model

Usage

```
donor_weights(x, ...)

## S3 method for class 'bscmfit'
donor_weights(x, summary = TRUE, probs = c(0.025, 0.5, 0.975), ...)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A tibble of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
donor_weights(fit_single_treated) |> head(5)
```

effective_donors *Extract the number of effective donors in a Bayesian synthetic control model*

Description

Effective donors is defined as $1/\sum_{j=1}^{J}\omega_j^2$, where ω_j is the donor weight of control unit j.

Usage

```
effective_donors(x, ...)

## S3 method for class 'bscmfit'
effective_donors(
  x,
  average = FALSE,
  summary = TRUE,
  probs = c(0.025, 0.975),
  ...
)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
average	[logical(1)] If TRUE, returns the average effective donors over treated units in case of multiple treated units. The default is FALSE.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A data.frame of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
effective_donors(fit_single_treated)
```

---

fit_single_treated *Example bscmfit object*

Description

The object fit_single_treated contains a Bayesian synthetic control model estimated as

```
fit <- bscm(
  formula = y ~ x, data = single_treated, treatment = "treatment",
  chains = 2, cores = 1, refresh = 0, iter = 2000, warmup = 1000,
  control = list(adapt_delta = 0.8)
)
```

Format

A bscmfit object.

Examples

```
fit_single_treated
plot(fit_single_treated)
```

---

fitted.bscmfit *Expected values of posterior predictive distribution of Bayesian synthetic control model*

Description

Returns posterior draws (or summaries) of expected values y_mean of the posterior predictive distribution of the model.

Usage

```
## S3 method for class 'bscmfit'
fitted(object, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

object	[bscmfit] The model fit object.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).
...	Ignored.

Value

A data.frame of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
fitted(fit_single_treated) |> head()
```

get_priors *Get prior specifications used in the Bayesian synthetic control model*

Description

Get prior specifications used in the Bayesian synthetic control model

Usage

```
get_priors(x, ...)

## S3 method for class 'bscmfit'
get_priors(x, ...)
```

Arguments

x	[bscmfit] Output from bscm().
...	Ignored.

Value

A named list of bscm_prior objects for the parameters of the model.

get_refmodel.bscmfit *Create a projpred reference model from a BSCM fit*

Description

Creates a refmodel object from bscmfit to be used with projpred package. This enables the usage of projpred::varsel() and projpred::cv_varsel() for donor selection. Note that for predictions with newdata, you need to call proj_predict_bscm() which is a wrapper of projpred::proj_predict() to which converts the data to the wide format used by projpred.

Usage

```
## S3 method for class 'bscmfit'
get_refmodel(object, ...)
```

Arguments

object	A bscmfit object from bscm().
...	Additional arguments passed to projpred::init_refmodel().

Details

Projection are based on only on the pretreatment period.

This function is experimental and currently only a single treated unit is supported.

For projpred integration, donors are treated as separate "predictors" in the formula. Currently only supported model is one without extra predictors. There are also other restrictions, namely lack of support for K-fold.

Value

An object of class `refmodel`.

See Also

`projpred::varsel()`, `projpred::cv_varsel()`

Examples

```
fit <- bscm(
  y ~ 1, treatment = "treatment", time = "time", unit = "id",
  data = single_treated, refresh = 0, chains = 2, cores = 2
)
refmodel <- get_refmodel(fit)
vs <- projpred::varsel(refmodel)
plot(vs, stats = c("elpd", "rmse"), alpha = 0.05)
# predictions using the projection:
predictions <- projpred::proj_predict(vs)
# posterior mean of the predictions
colMeans(predictions)
```

`get_standata` *Get input data to Stan from* `bscmfit` *object*

Description

Reconstructs the data list passed to Stan in `bscm()`. Used by methods such as `loo()` with `reloo = TRUE`.Requires the original data (`save_data = TRUE`).

Usage

```
get_standata(x, ...)
```

Arguments

`x`	[`bscmfit`] The model fit object.
`...`	Ignored.

Value

A named list suitable for passing to `rstan::sampling()`.

`get_stanfit` *Extract the* `stanfit` *object from the* `bscmfit` *object*

Description

This function returns the output (`stanfit` object) from `rstan::sampling()`.

Usage

```
get_stanfit(x, ...)

## Default S3 method:
get_stanfit(x, ...)

## S3 method for class 'bscmfit'
get_stanfit(x, ...)
```

Arguments

`x`	[`bscmfit`] The output returned by the `bscm()`.
`...`	Ignored.

Value

Object of class `stanfit`.

`leave_donor_out` *Leave-out donor sensitivity of a Bayesian synthetic control model*

Description

`leave_donor_out()` re-estimates the original model after omitting donors from the donor pool. By default, it removes one donor at a time. Setting `cumulative = TRUE` instead removes donors cumulatively following the chosen order, which can be used to assess how sensitive the results are to the most influential donors.

Usage

```
leave_donor_out(x, ...)

## S3 method for class 'bscmfit'
leave_donor_out(
  x,
  order = "descending",
  cumulative = FALSE,
  probs = c(0.025, 0.5, 0.975),
  ...
)
```

Arguments

x	[bscmfit] The output returned by the bscm().
...	Additional arguments passed on to bscm().
order	[character() or NULL] Donor order used for leave-out runs. Either a vector of donor names, or "descending" / "ascending" to rank donors by posterior means of donor weights. Default is "descending". Use NULL to use the original donor order.
cumulative	[logical(1)] If FALSE (the default), omit one donor at a time. If TRUE, omit donors cumulatively following order.
probs	[numeric()] Probabilities for quantile summaries of the treatment effects and RMSE estimates. Default is c(0.025, 0.975).

Details

The donor order can always be supplied explicitly. For one-donor leave-out runs (cumulative = FALSE), that order controls the order of the returned runs. For cumulative leave-out runs (cumulative = TRUE), donors are removed from the ordered pool one by one. If no order is supplied, one-donor leave-out uses the original donor order, while cumulative leave-out orders donors by posterior mean donor weight in descending order.

For models with multiple treated units, automatic donor ranking is based on the average posterior mean donor weight across treated units.

Value

An object of class bscm_ldo with data frames effect, rmse, rmse_ratio, weights, and diagnostics, and a metadata list. The data frames contain posterior summaries for the original fit (step = 0) and for each leave-out run, identified by step, n_removed, and last_removed. The metadata list contains the omitted donors, whether donor omission was cumulative, summary probabilities, and model metadata needed for plotting. The result can be visualized with plot_weights() and plot_effects().

lfo *Leave-Future-Out Cross-Validation*

Description

Estimates the leave-future-out (LFO) expected log predictive density (ELPD) for `bscmfit` models. The LFO-CV is performed over the pre-treatment period: at each step `t` (from `L` to `T_pre_min - 1`), the model is evaluated on its ability to predict the next pre-treatment observation `y[t+1]` having been fitted on `y[1:t]`. For models with multiple treated units, all units must have at least `L + 1` pre-treatment observations, and only the first `min(T_pre)` time points are used so the number of treated units stays constant throughout the evaluation period. In this case, a joint ELPD of all treated units is computed at each step.

Usage

```
lfo(x, ...)

## S3 method for class 'bscmfit'
lfo(x, L, exact = FALSE, k_threshold = 0.7, ...)
```

Arguments

`x`	[`bscmfit`] The output returned by `bscm()`.
`...`	Additional arguments passed on to `bscm()` when refitting.
`L`	[`integer(1)`] Minimum number of pre-treatment observations used for the first fit. Must satisfy `2 <= L <= T_pre_min - 2`, where `T_pre_min = min(T_pre)`. Too small value of `L` can lead to unstable estimation, so a value of at least `10` is recommended.
`exact`	[`logical(1)`] If `TRUE`, computes exact LFO by refitting at every step. If `FALSE` (the default), uses the approximate PSIS-LFO method.
`k_threshold`	[`numeric(1)`] Threshold for the Pareto k diagnostic that triggers a model refit. Default is `0.7`. Ignored when `exact = TRUE`.

Value

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

- `ELPD`: Total expected log predictive density.
- `ELPD_SE`: A crude approximation of standard error of ELPD, ignoring serial dependency of ELPD estimates.
- `ELPDs`: Vector of per-step ELPDs (length `T_pre_min - L`). Element `k` is the ELPD for predicting observations at `L + k`.

- ks: Pareto k values (length T_pre_min - L - 1, NULL for exact LFO).
- refits: Time indices at which the model was re-estimated.
- L: The value of L used.
- T_pre_min: The minimum pre-treatment period length across treated units.
- times: Vector of all unique time values from the original data.
- time: Name of the time variable.
- k_thres: The Pareto k threshold used.

References

Bürkner PC, Gabry J, and Vehtari A (2020). Approximate leave-future-out cross-validation for Bayesian time series models. *Journal of Statistical Computation and Simulation*, 90(14), 2499–2523, doi:10.1080/00949655.2020.1783262.

Examples

```
lfo_approx <- lfo(fit_single_treated, L = 10)
lfo_approx

lfo_exact <- lfo(fit_single_treated, L = 10, exact = TRUE)
lfo_exact
```

---

log_lik.bscmfit *Posterior draws of pointwise log-likelihood*

Description

Returns posterior draws of pointwise log-likelihoods of the treated units per pre-treatment time point

Usage

```
## S3 method for class 'bscmfit'
log_lik(object, ...)
```

Arguments

object	[bscmfit] The model fit object.
...	Ignored.

Value

A matrix where rows correspond to posterior draws and columns to observed (non-missing) time periods. Output columns are ordered by unit then time: all observed pre-treatment time points for the first treated unit, then the second, and so on.

Examples

```
log_lik(fit_single_treated) |> head()
```

`loo.bscmfit` *Approximate leave-one-out (LOO) cross-validation for Bayesian synthetic control models*

Description

Approximate leave-one-out (LOO) cross-validation for Bayesian synthetic control models

Usage

```
## S3 method for class 'bscmfit'
loo(x, r_eff = TRUE, reloo = FALSE, k_threshold = 0.7, ...)
```

Arguments

`x`	[`bscmfit`] The model fit object.
`r_eff`	[`logical(1)`] If `TRUE` (the default), `loo::loo()` computes more accurate Monte Carlo error estimates at the cost of increased computation time.
`reloo`	[`logical(1)`] If `TRUE`, observations whose Pareto `k` diagnostic exceeds `k_threshold` are handled by exact leave-one-out refits rather than PSIS approximation. Requires `save_data = TRUE` in the original fit. Default is `FALSE`.
`k_threshold`	[`numeric(1)`] Threshold for the Pareto `k` diagnostic that triggers an exact refit when `reloo = TRUE`. Default is `0.7`.
`...`	Additional arguments to `loo::loo()`.

Value

An output from `loo::loo()`.

References

Vehtari A, Gelman A, and Gabry J (2017). Practical Bayesian model evaluation using leave-one-out cross-validation and WAIC. *Statistics and Computing*. 27(5), 1413–1432, doi:10.1007/s11222-016-9696-4.

Examples

```
loo(fit_single_treated)
```

`loo_R2.bscmfit` *Bayesian R-squared value*

Description

`bayes_R2` computes the Bayesian R^2 measure of model fit for a Bayesian synthetic control model, while `loo_R2` computes a leave-one-out adjusted version of the same quantity.

Usage

```
## S3 method for class 'bscmfit'
loo_R2(object, summary = TRUE, probs = c(0.025, 0.975), fixed_seed = TRUE, ...)

## S3 method for class 'bscmfit'
bayes_R2(object, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

`object`	[`bscmfit`] The model fit object.
`summary`	[`logical(1)`] If `TRUE` (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the `probs` argument), and MCMC convergence measures. If `FALSE`, returns the posterior draws instead.
`probs`	[`numeric()`] Probabilities for quantile summaries. Default is `c(0.025, 0.975)`.
`fixed_seed`	[logical(1)] If `TRUE` (the default), fixes the seed of random number generator (RNG) so that `loo_R2`, which uses Bayesian bootstrap, returns identical results in repeated calls for the same model object. On exit, the state of the RNG is restored to the original state.
`...`	Ignored.

Value

`data.frame` of posterior summary of (LOO-adjusted) R-squared values.

References

Gelman A, Goodrich B, Gabry J, and Vehtari A (2019). R-squared for Bayesian regression models. *The American Statistician*. 73(3), 307–309, doi:10.1080/00031305.2018.1549100.

Examples

```
bayes_R2(fit_single_treated)
```

multiple_treated *Simulated example data with multiple treated units*

Description

This simulated data is generated based on a latent factor model with two factors, 40 time points and 53 units. For the first three units, a "treatment" $\tau = 2$ for the last 5 time points $t = 0, \ldots, 4$ is added to an outcome `y`. There are two time-varying covariates `x` and `z`, with latter one having time-varying effect on `y`. To see exactly how the data was generated, see `data-raw` folder on the Github repository of the package.

Format

A data frame with 1,550 rows and 5 columns:

- time: Time index from ranging from -30 to 9.
- id: Unit index variable ‘ranging from 1 to 33.
- y: Outcome variable.
- x: Time-varying predictor.
- z: Time-varying predictor.
- treatment: Binary indicator variable where 1 corresponds to the treatment.

Examples

```
head(multiple_treated)
multiple_treated |>
  dplyr::mutate(treated = any(treatment), .by = id) |>
  ggplot2::ggplot(ggplot2::aes(time, y)) +
  ggplot2::geom_line(
    ggplot2::aes(group = id, colour = treated, alpha = treated)
  ) +
  ggplot2::scale_alpha_manual(values = c(0.3, 1)) +
  ggplot2::theme_bw()
```

nchains.bscmfit *Return the number of chains of* `bscmfit` *object*

Description

Return the number of chains of `bscmfit` object

Usage

```
## S3 method for class 'bscmfit'
nchains(x)
```

Arguments

x	[bscmfit] The model fit object.

Value

Number of Markov chains used in sampling as a single integer value.

---

ndraws.bscmfit *Return the number of posterior draws of a* bscmfit *object*

Description

Return the number of posterior draws of a bscmfit object

Usage

```
## S3 method for class 'bscmfit'
ndraws(x)
```

Arguments

x	[bscmfit] The model fit object.

Value

Number of posterior draws as a single integer value.

---

placebo_effects *Placebo effects of a Bayesian synthetic control model*

Description

Estimates in-space and in-time placebo effects based on the model definition of previously estimated Bayesian synthetic control model.

Usage

```
placebo_effects(x, ...)

## S3 method for class 'bscmfit'
placebo_effects(x, type, L = NULL, probs = c(0.025, 0.975), ...)
```

Arguments

x	[bscmfit] The output returned by the bscm().
...	Additional arguments passed on to bscm().
type	[character(1)] Type of the placebo effects to compute. Either "donor" for in-space placebos, "time" for in-time placebos. See details.
L	[integer(1)] If type = "time, minimum number of observations to use for the in-time placebos, i.e. the number of pre-treatment time points for the first fit. For too small L, estimation can be unstable, so you should likely use at least L = 10 or so.
probs	[numeric()] Probabilities for quantile summaries of the treatment effects and RMSE estimates. Default is c(0.025, 0.975).

Details

For the in-space placebo (type = "donor"), original model is re-estimated using each donor as the treated unit in turn (omitting the original, true treated unit). The obtained effect estimates should be around zero, under the assumption that the treatment affected only the true treated unit.

For the in-time placebo (type = "time") we still estimate the treatment effect for the original treated, but move the start of the treatment from $L + 1$ to $T_{pre} + 1$, where L is the minimum number of pre-treatment time points to use, and T_{pre} is the true last pre-treatment time point. In all cases, the obtained treatment effects should fluctuate around zero for time points before the true treatment time, under the assumption of no anticipation effects.

In both cases, the output includes also the results of the input model.

Value

A list with data frames effect, rmse, rmse_ratio and diagnostics, and a metadata list. The data frames contain posterior summaries for each run, identified by a placebo column. For type = "donor", placebo is the treated unit name for the original fit and the donor name for each placebo run. For type = "time", placebo is the assumed treatment start time for each run. The metadata list contains the placebo type, summary probabilities, model setup, and the placebo labels used. The result can be visualized with plot_effects().

---

plot.bscm_lfo *Plot Pareto k diagnostics from LFO cross-validation*

Description

Plots Pareto k values against the corresponding time variable values, with a horizontal line at the refitting threshold. Only available (and relevant) for PSIS-LFO (not exact LFO).

Usage

```
## S3 method for class 'bscm_lfo'
plot(x, ...)
```

Arguments

x	[bscm_lfo] Output from lfo().
...	Ignored.

Value

A ggplot object, or NULL invisibly for exact LFO.

---

plot.bscmfit *Visualize BSCM estimates*

Description

A plot of the posterior mean and posterior interval of the treatment effect and synthetic control over time for single treated unit models. For models with multiple treated units, plots the average treatment effect over time since treatment across all treated units.

Usage

```
## S3 method for class 'bscmfit'
plot(x, probs = c(0.025, 0.975), ...)
```

Arguments

x	[bscmfit] object.
probs	[numeric(2)] Vector of length two defining the limits of the posterior interval. Default is c(0.025, 0.975).
...	Ignored

Value

A ggplot object

Examples

```
plot(fit_single_treated, probs = c(0.05, 0.95))
```

plot_coefs *Visualize regression coefficients of Bayesian synthetic control model*

Description

Plots posterior means and posterior intervals for the model coefficients. For time-constant coefficients (`type = "fixed"`), a horizontal point-range plot is produced. For time-varying coefficients (`type = "varying"`), a ribbon-and-line plot over time is produced, faceted by parameter.

Usage

```
plot_coefs(x, ...)

## S3 method for class 'bscmfit'
plot_coefs(
  x,
  type = c("fixed", "varying"),
  probs = c(0.025, 0.975),
  combine = FALSE,
  alpha = 0.5,
  scales = "free_y",
  ...
)
```

Arguments

`x`	[bscmfit] Output from `bscm()`.
`...`	Ignored.
`type`	[character(1)] Type of coefficients to plot. Either `"fixed"` for time-constant coefficients or `"varying"` (the default) for time-varying coefficients.
`probs`	[numeric(2)] Vector of length two defining the limits of the posterior interval. Default is `c(0.025, 0.975)`.
`combine`	[logical(1)] If `TRUE` and `type = "varying"`, plot the total effect (beta + gamma) instead of gamma alone. Ignored when `type = "fixed"`. Default is `FALSE`.
`alpha`	[numeric(1)] Opacity of the credible-interval ribbon (used only when `type = "varying"`). Default is `0.5`.
`scales`	[character(1)] Passed to `ggplot2::facet_wrap()` (used only when `type = "varying"`). Default is `"free_y"`.

Value

A ggplot object.

Examples

```
plot_coefs(fit_single_treated)
```

`plot_effects` *Visualize BSCM treatment effects*

Description

`plot_effects()` plots the posterior mean and posterior interval of the treatment effect over time. For the output of `leave_donor_out()` or `placebo_effects()`, it additionally overlays the posterior mean of the treatment effect from each leave-out or placebo fit.

Usage

```
plot_effects(x, ...)

## S3 method for class 'bscmfit'
plot_effects(x, probs = c(0.025, 0.975), unit = NULL, ...)

## S3 method for class 'bscm_ldo'
plot_effects(x, probs = NULL, ...)

## S3 method for class 'bscm_placebo_effects'
plot_effects(x, probs = NULL, ...)
```

Arguments

`x`	[`bscmfit`, `bscm_ldo`, or `bscm_placebo_effects`] Object from `bscm()`, `leave_donor_out()`, or `placebo_effects()`.
`...`	Ignored.
`probs`	[`numeric(2)`] Vector of length two defining the limits of the posterior interval. Default is `c(0.025, 0.975)`. For `bscm_ldo` and `bscm_placebo_effects` objects, defaults to the outermost probabilities used in `leave_donor_out()` or `placebo_effects()`.
`unit`	[`character(1)` or `numeric(1)` or `NULL`] For models with multiple treated units, the identifier of the specific treated unit to plot. If `NULL` (the default), the average treatment effect across all treated units is plotted.

Details

For models with multiple treated units, `plot_effects.bscmfit()` defaults to plotting the average treatment effect over time since treatment. Supply a `unit` identifier to plot the effect for a specific treated unit instead.

Value

A ggplot object.

Examples

```
plot_effects(fit_single_treated)
```

`plot_residuals` *Visualize BSCM residuals*

Description

`plot_residuals()` plots the posterior mean and posterior interval of the pre-treatment residuals over time (`type = "time"`), or the posterior distribution of the residual autocorrelations up to a given number of lags (`type = "autocorrelation"`), again only for pre-treatment time points.

Usage

```
plot_residuals(x, ...)

## S3 method for class 'bscmfit'
plot_residuals(x, type = "time", probs = c(0.025, 0.975), max_lag = 5L, ...)
```

Arguments

`x`	[bscmfit] Object from `bscm()`.
`...`	Ignored.
`type`	[character(1)] Plot type. Either `"time"` (default) for a ribbon plot of residuals over time, or `"autocorrelation"` for posterior distributions of residual autocorrelations at each lag.
`probs`	[numeric(2)] Vector of length two defining the limits of the posterior interval. Default is `c(0.025, 0.975)`.
`max_lag`	[integer(1)] Maximum number of lags for `type = "autocorrelation"`. Default is `5L`.

Details

For models with multiple treated units, returns a named list of per-unit plots.

Value

A ggplot object, or a named list of ggplot objects when `x` is a `bscmfit` with multiple treated units.

Examples

```
plot_residuals(fit_single_treated)
plot_residuals(fit_single_treated, type = "autocorrelation", max_lag = 10L)
```

`plot_weights` *Visualize donor weights*

Description

`plot_weights()` visualizes posterior summaries of donor weights for a fitted Bayesian synthetic control model. When applied to the output of `leave_donor_out()`, it instead visualizes how donor weights or their ranks change across the leave-out runs.

Usage

```
plot_weights(x, ...)

## S3 method for class 'bscmfit'
plot_weights(
  x,
  point_estimate = "median",
  order = NULL,
  coverage = c(0.5, 0.95),
  linewidth = 1,
  point_size = 2,
  reverse = NULL,
  ...
)

## S3 method for class 'bscm_ldo'
plot_weights(
  x,
  type = "weight",
  point_estimate = "median",
  coverage = 0.95,
  linewidth = 1,
  point_size = 2,
  reverse = TRUE,
  ...
)
```

Arguments

`x`	[`bscmfit` or `bscm_ldo`] Output from `bscm()` or `leave_donor_out()`.
`...`	Ignored.

point_estimate [´character(1)] Should the point estimate in weight plot correspond to posterior "median"(the default), or"mean"‘?

order [character() or NULL]
Order of donors in y-axis. Either a vector of donor names, or "descending" / "ascending" to rank donors by posterior means of donor weights. Default is "descending". Use NULL to use the original donor order. Ignored for bscm_ldo method which uses the ordering which was used in leave_donor_out().

coverage [numeric()]
Coverages of posterior intervals of donor weights when type = "weight". By default 50% and 95% posterior intervals are drawn.

linewidth [numeric(1)]
Maximum line width used for intervals or donor rank trajectories. Default is 1.

point_size [numeric(1)]
Point size for point estimates or trajectory endpoints. Default is 2.

reverse [logical(1)]
Should the y-axis be reversed? Default is NULL, in which case y-axis is reversed except when order = "ascending", so in both ascending and descending ordering of donors leads to plot where donor with largest weight is on top.

type [character(1)]
For bscm_ldo objects, plot posteriors of donor weights ("weight") or donor weight ranks ("rank"). Default is "weight", which is also only option for bscmfit objects.

Value

A ggplot object, or a named list of ggplot objects for models with multiple treated units.

Examples

```
plot_weights(fit_single_treated)
```

---

posterior_epred.bscmfit

Posterior Draws of the Expected Predictive Distribution

Description

Returns draws from the posterior distribution of the expected value of the Bayesian synthetic control model.

Usage

```
## S3 method for class 'bscmfit'
posterior_epred(object, ...)
```

Arguments

`object`	[bscmfit] The model fit object.
`...`	Ignored.

Value

A matrix where rows correspond to posterior draws and columns to time periods.

`posterior_linpred.bscmfit`

Posterior Draws of the Linear Predictor

Description

Returns draws from the posterior distribution of the linear predictor of the Bayesian synthetic control model. Since the model uses a Gaussian likelihood with identity link, this is equivalent to `posterior_epred.bscmfit()`.

Usage

```
## S3 method for class 'bscmfit'
posterior_linpred(object, transform = FALSE, ...)
```

Arguments

`object`	[bscmfit] The model fit object.
`transform`	[logical(1)] Ignored.
`...`	Ignored.

Value

A matrix where rows correspond to posterior draws and columns to time periods.

posterior_predict.bscmfit *Posterior Predictive Distribution of Bayesian Synthetic Control Model*

Description

Returns draws from the posterior predictive distribution of the Bayesian synthetic control. Note that this function does not simulate new realizations from this distribution, but rather returns the posterior draws computed during the model estimation.

Usage

```
## S3 method for class 'bscmfit'
posterior_predict(object, ...)
```

Arguments

object	[bscmfit] The model fit object.
...	Ignored.

Value

A matrix of posterior predictive draws of the synthetic control, where rows correspond to posterior draws and columns to time periods.

print.bscm_lfo *Print method for LFO cross-validation output*

Description

Prints the summary of the leave-future-out cross-validation.

Usage

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

Arguments

x	[bscm_lfo] Output from lfo().
...	Ignored.

Value

Returns x invisibly.

print.bscmfit *Print method for bscmfit objects*

Description

Print method for bscmfit objects

Usage

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

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

Arguments

x	[bscmfit or summary_bscmfit] Output from bscm() or summary.bscmfit().
...	Ignored.

Value

Returns x (invisibly).

Returns x (invisibly).

print.bscmfit_diagnostics *Print MCMC Diagnostics*

Description

Print MCMC Diagnostics

Usage

```
## S3 method for class 'bscmfit_diagnostics'
print(x, print_table = NULL, ...)
```

Arguments

x	[bscmfit_diagnostics] The diagnostics object returned by check_mcmc_diagnostics.bscmfit().
print_table	[logical(1)] If NULL (the default) only prints the table of largest Rhat and smallest ESS values in case diagnostics indicate problems. If TRUE or FALSE, always prints or does not print the table.
...	Ignored.

Value

input x (invisibility).

`proj_predict_bscm` *Predictions from a projected BSCM model*

Description

Convenience wrapper around `projpred::proj_predict()` that accepts data in the original long format used by `bscm()`. If `newdata` is supplied in long format, it is converted internally to the wide format for `projpred`.

Usage

```
proj_predict_bscm(object, newdata = NULL, ...)
```

Arguments

`object`	A projection object or variable selection object returned by `projpred::project()`, `projpred::varsel()`, or `projpred::cv_varsel()`.
`newdata`	Optional new data used for predictions. The outcome, unit, and time variables should have should have names matching the original data. If `NULL` (the default), predictions are made for the pre-treatment period of the original data used to fit the model.
`...`	Additional arguments passed to `projpred::proj_predict()`.

Value

The output of `projpred::proj_predict()`.

`residual_acf` *Autocorrelation function for residuals of BSCM*

Description

Autocorrelation function for residuals of BSCM

Usage

```
residual_acf(x, max_lag = 5L, probs = c(0.025, 0.975))
```

Arguments

x	Posterior draws from `residuals.bscmfit()` with `summary = FALSE`, or `bscmfit` object.
max_lag	[integer(1)] Positive integer defining the maximum lag at which to compute the autocorrelations. The default is `5L`.
probs	[numeric(2)] Vector of length two defining the limits of the posterior interval. Default is `c(0.025, 0.975)`.

Value

A data frame.

Examples

```
residual_acf(fit_single_treated, max_lag = 10)
```

---

residuals.bscmfit *Posterior residuals of a Bayesian synthetic control model*

Description

Returns posterior draws or summaries of residuals, defined as the difference between the observed outcome and the posterior expected value.

Usage

```
## S3 method for class 'bscmfit'
residuals(
  object,
  summary = TRUE,
  probs = c(0.025, 0.975),
  pretreatment_only = TRUE,
  ...
)
```

Arguments

object	[bscmfit] The model fit object.
summary	[logical(1)] If `TRUE` (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the `probs` argument), and MCMC convergence measures. If `FALSE`, returns the posterior draws instead.

probs [numeric()]
Probabilities for quantile summaries. Default is c(0.025, 0.975).

pretreatment_only
[logical(1)]
If TRUE (the default), only pre-treatment time points are returned, while FALSE returns all time points.

... Ignored.

Value

A data.frame of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
residuals(fit_single_treated) |> head()
```

---

rmse *Extract root mean squared errors of a Bayesian synthetic control model*

Description

Returns posterior draws (or summaries) of root mean squared errors (RMSEs) for the pre-treatment and post-treatment periods.

Usage

```
rmse(x, ...)

## S3 method for class 'bscmfit'
rmse(x, average = FALSE, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

x [bscmfit]
The model fit object.

... Ignored.

average [logical(1)]
If TRUE, returns the average RMSEs over treated units in case of multiple treated units. If FALSE (the default), unit-specific values are returned.

summary [logical(1)]
If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.

probs [numeric()]
Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A tibble of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE).

Examples

```
rmse(fit_single_treated, probs = c(0.01, 0.5, 0.8))
```

---

rmse_ratio *Extract the ratio of post- to pre-treatment RMSE*

Description

Returns posterior draws (or summaries) of the ratio of the post-treatment RMSE to the pre-treatment RMSE.

Usage

```
rmse_ratio(x, ...)

## S3 method for class 'bscmfit'
rmse_ratio(x, average = FALSE, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
average	[logical(1)] If TRUE, returns the average ratio over treated units in case of multiple treated units. If FALSE (the default), unit-specific values are returned.
summary	[logical(1)] If TRUE (the default), returns posterior summaries. If FALSE, returns posterior draws.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A tibble of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE).

Examples

```
rmse_ratio(fit_single_treated, probs = c(0.01, 0.5, 0.8))
```

sigma.bscmfit *Extract residual standard deviations of a Bayesian synthetic control model*

Description

Extract residual standard deviations of a Bayesian synthetic control model

Usage

```
## S3 method for class 'bscmfit'
sigma(object, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

object	[bscmfit] The model fit object.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).
...	Ignored.

Value

A data.frame of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
sigma(fit_single_treated)
```

single_treated *Simulated example data with single treated unit*

Description

This simulated data is generated based on a latent factor model with three factors, 50 time points and 51 units. For the first unit (id = 1), a "treatment" $\tau = 1 + t$ for the last 10 time points $t = 0, \ldots, 9$ is added to an outcome y. To see exactly how the data was generated, see data-raw folder on the Github repository of the package.

Format

A data frame with 2550 rows and 5 columns:

- time: Time index from ranging from -40 to 9.
- id: Unit index variable ‘ranging from 1 to 31.
- y: Outcome variable.
- x: Time-varying predictor.
- treatment: Binary indicator variable where 1 corresponds to the treatment.

Examples

```
head(single_treated)
single_treated |>
  ggplot2::ggplot(ggplot2::aes(time, y, group = id)) +
  ggplot2::geom_line(alpha = 0.3) +
  ggplot2::geom_line(
    data = single_treated |> dplyr::filter(id == 1), colour = "tomato"
  ) +
  ggplot2::theme_bw()
```

`summary.bscmfit` *Summarise posterior draws of an estimated Bayesian synthetic control model*

Description

Generates posterior summary statistics for a Bayesian synthetic control model estimated with `bscm()`. Note that many (more) summaries are available via separate methods, e.g., `donor_weights()`.

Usage

```
## S3 method for class 'bscmfit'
summary(object, probs = c(0.025, 0.975), ...)
```

Arguments

`object`	[`bscmfit`] The model fit object.
`probs`	[`numeric()`] Probabilities for quantile summaries. Default is `c(0.025, 0.975)`.
`...`	Ignored.

Value

A data frame with various posterior summaries of the model.

synthetic_control *Synthetic control series of a Bayesian synthetic control model*

Description

Returns posterior draws (or summaries) of the pre- and post-treatment trajectories of treated units, i.e. draws y_repfrom the posterior predictive distribution of the model.

Usage

```
synthetic_control(x, ...)

## S3 method for class 'bscmfit'
synthetic_control(x, summary = TRUE, probs = c(0.025, 0.975), ...)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A data.frame of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
synthetic_control(fit_single_treated) |> tail()
```

treatment_effect *Treatment effect estimates of a Bayesian synthetic control model*

Description

Treatment effect estimates of a Bayesian synthetic control model

Usage

```
treatment_effect(x, ...)

## S3 method for class 'bscmfit'
treatment_effect(
  x,
  average = TRUE,
  summary = TRUE,
  probs = c(0.025, 0.975),
  ...
)
```

Arguments

x	[bscmfit] The model fit object.
...	Ignored.
average	[logical(1)] If TRUE (the default), returns the average treatment effect over treated units at each time since treatment (event time). If FALSE, unit-specific effects at each calendar time point are returned.
summary	[logical(1)] If TRUE (the default), returns posterior mean, standard deviation, posterior quantiles (as defined by the probs argument), and MCMC convergence measures. If FALSE, returns the posterior draws instead.
probs	[numeric()] Probabilities for quantile summaries. Default is c(0.025, 0.975).

Value

A tibble of posterior summaries (summary = TRUE) or posterior draws (summary = FALSE) in long format.

Examples

```
treatment_effect(fit_single_treated) |> tail()
```

---

tv *Define the time-varying coefficients for BSCM*

Description

Function tv() validates its arguments and returns the necessary variables needed for the construct the time-varying coefficients in bscm(). You should not call tv() separately, instead it should be part of a formula, e.g., y ~ x + tv(~ z * w, df = 20).

Usage

```
tv(tv_formula, df = 10, type = "rw1", noncentered = TRUE)
```

Arguments

tv_formula	[formula] One-sided formula, e.g., ~ x + z defining the the time-varying part of the BSCM formula.
df	[integer(1)] Integer defining the number of spline basis functions. Default is df = 10.
type	[character(1)] Prior type for spline coeffients. Either "rw1" or "rw2" for first and second order random walks.
noncentered	[logical(1)] If TRUE (the default), the spline coefficients are sampled using a non-centered parameterization. If FALSE, a centered parameterization is used. Depending on the case, one of these might lead to more efficient and numerically stable sampling, so if you encounter divergences, try changing this.

Value

Object of class tv_term (a list).

Index

∗ **datasets**
multiple_treated, 27
single_treated, 43

as.data.frame *(*as.data.frame.bscmfit*)*, 4
as.data.frame.bscmfit, 4
as_draws *(*as_draws.bscmfit*)*, 5
as_draws.bscmfit, 5
as_draws.bscmfit(), *5*, *9*
average_treatment_effect, 6

bayes_R2 *(*loo_R2.bscmfit*)*, 26
beta_pr *(*bscm_prior*)*, 9
bscm, 7
bscm(), *3*, *8*, *9*, *11*, *15*, *19–23*, *29*, *31–34*, *38*, *39*, *44*, *46*
bscm-package, 3
bscm_prior, 9
bscm_prior(), *8*

check_mcmc_diagnostics, 11
check_mcmc_diagnostics.bscmfit(), *8*, *38*
coef *(*coef.bscmfit*)*, 12
coef.bscmfit, 12
covariate_adjustment, 13
covariate_imbalance, 14

dirichlet_pr *(*bscm_prior*)*, 9
donor_ranking, 15
donor_weights, 16
donor_weights(), *4*, *44*

effective_donors, 16
exponential_pr *(*bscm_prior*)*, 9

fit_single_treated, 17
fitted *(*fitted.bscmfit*)*, 18
fitted.bscmfit, 18

gamma_pr *(*bscm_prior*)*, 9
get_priors, 19
get_refmodel *(*get_refmodel.bscmfit*)*, 19
get_refmodel.bscmfit, 19
get_standata, 20
get_stanfit, 21
ggplot2::facet_wrap(), *13*, *14*, *31*

half_normal_pr *(*bscm_prior*)*, 9

leave_donor_out, 21
leave_donor_out(), *32*, *34*, *35*
lfo, 23
lfo(), *8*, *30*, *37*
log_lik *(*log_lik.bscmfit*)*, 24
log_lik.bscmfit, 24
logistic_normal_pr *(*bscm_prior*)*, 9
loo *(*loo.bscmfit*)*, 25
loo(), *20*
loo.bscmfit, 25
loo::loo(), *25*
loo_R2 *(*loo_R2.bscmfit*)*, 26
loo_R2.bscmfit, 26

multiple_treated, 27

nchains *(*nchains.bscmfit*)*, 27
nchains.bscmfit, 27
ndraws *(*ndraws.bscmfit*)*, 28
ndraws.bscmfit, 28
normal_pr *(*bscm_prior*)*, 9

placebo_effects, 28
placebo_effects(), *32*
plot *(*plot.bscmfit*)*, 30
plot.bscm_lfo, 29
plot.bscmfit, 30
plot_coefs, 31
plot_effects, 32
plot_effects(), *22*, *29*
plot_residuals, 33
plot_weights, 34
plot_weights(), *22*

posterior::as_draws(), *6*
posterior::default_convergence_measures(), *11*
posterior::summarise_draws(), *6*
posterior_epred (posterior_epred.bscmfit), 35
posterior_epred(), *4*
posterior_epred.bscmfit, 35
posterior_epred.bscmfit(), *36*
posterior_linpred (posterior_linpred.bscmfit), 36
posterior_linpred.bscmfit, 36
posterior_predict (posterior_predict.bscmfit), 37
posterior_predict(), *4*, *8*
posterior_predict.bscmfit, 37
print.bscm_lfo, 37
print.bscmfit, 38
print.bscmfit_diagnostics, 38
print.summary_bscmfit (print.bscmfit), 38
proj_predict_bscm, 39
proj_predict_bscm(), *19*
projpred::cv_varsel(), *15*, *19, 20*, *39*
projpred::init_refmodel(), *19*
projpred::proj_predict(), *19*, *39*
projpred::project(), *39*
projpred::varsel(), *15*, *19, 20*, *39*

residual_acf, 39
residuals (residuals.bscmfit), 40
residuals.bscmfit, 40
residuals.bscmfit(), *40*
rmse, 41
rmse(), *8*
rmse_ratio, 42
rstan::check_hmc_diagnostics(), *11*
rstan::sampling(), *8, 9*, *21*

sigma (sigma.bscmfit), 43
sigma.bscmfit, 43
single_treated, 43
student_pr (bscm_prior), 9
summary (summary.bscmfit), 44
summary(), *8*
summary.bscmfit, 44
summary.bscmfit(), *9*, *38*
synthetic_control, 45
synthetic_control(), *8*
treatment_effect, 45
treatment_effect(), *8*
tv, 46