Package 'SelectBoost.beta'

May 7, 2026

Type Package

Title Stability-Selection via Correlated Resampling for Beta-Regression Models

Version 0.4.5

Date 2025-11-04

Depends R (>= 4.0)

Imports betareg, gamlss, gamlss.dist, glmnet, MASS, Rcpp, rlang, stats, withr

Suggests future, future.apply, gamlss.lasso, ggplot2, knitr, rmarkdown, testthat (>= 3.0.0)

LinkingTo Rcpp, RcppArmadillo

Author Frederic Bertrand [cre, aut] (ORCID: <https://orcid.org/0000-0002-0837-8281>)

Maintainer Frederic Bertrand <frederic.bertrand@lecnam.net>

Description Adds variable-selection functions for Beta regression models (both mean and phi submodels) so they can be used within the 'SelectBoost' algorithm. Includes stepwise AIC, BIC, and corrected AIC on betareg() fits, 'gamlss'-based LASSO/Elastic-Net, a pure 'glmnet' iterative re-weighted least squares-based selector with an optional standardization speedup, and 'C++' helpers for iterative re-weighted least squares working steps and precision updates. Also provides a fastboost_interval() variant for interval responses, comparison helpers, and a flexible simulator simulation_DATA.beta() for interval-valued data. For more details see Bertrand and Maumy (2023) <doi:10.7490/f1000research.1119552.1>.

License GPL-3

Encoding UTF-8

Classification/MSC 62H11, 62J12, 62J99

VignetteBuilder knitr

RoxygenNote 7.3.3

URL https://fbertran.github.io/SelectBoost.beta/, https://github.com/fbertran/SelectBoost.beta/

BugReports https://github.com/fbertran/SelectBoost.beta/issues/

Config/testthat/edition 3

NeedsCompilation yes

Repository CRAN

Date/Publication 2025-11-11 09:20:09 UTC

Contents

`betareg_enet_gamlss` *Beta regression Elastic-Net via GAMLSS (gamlss.lasso)*

Description

Uses `gamlss.lasso::gnet()` to fit ENet on the mean submodel of `gamlss(dist = BE)`. The routine assumes complete cases and does not expose offsets or precision-model terms.

Usage

```
betareg_enet_gamlss(
  X,
  Y,
  method = c("IC", "CV"),
  ICpen = c("BIC", "AIC", "HQC"),
```

```
  alpha = 1,
  trace = FALSE
)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
method	"IC" (information criterion) or "CV".
ICpen	Penalty for "IC" selection: "BIC", "AIC", or "HQC".
alpha	Elastic-net mixing (1 = LASSO, 0 = ridge).
trace	Logical; print stepwise trace.

Value

Named numeric vector of coefficients as in betareg_lasso_gamlss().

See Also

gamlss.lasso::gnet(), gamlss::gamlss(), gamlss.dist::BE()

---

betareg_glmnet *Pure glmnet IRLS selector for Beta regression*

Description

Runs an IRLS loop with Beta working responses/weights and calls glmnet on the weighted least-squares surrogate. Supports BIC/AIC/CV model choice and an optional prestandardize speedup. The helper uses only the mean submodel, requires complete cases, and does not expose offset terms.

Usage

```
betareg_glmnet(
  X,
  Y,
  alpha = 1,
  choose = c("bic", "aic", "cv"),
  nfolds = 5,
  n_iter = 6,
  tol = 1e-05,
  standardize = TRUE,
  lambda = NULL,
  phi_init = 20,
  update_phi = TRUE,
  phi_maxit = 5,
  prestandardize = FALSE,
  trace = FALSE
)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
alpha	Elastic-net mixing parameter.
choose	One of "bic", "aic", or "cv" to pick lambda.
nfolds	Folds for CV when choose = "cv".
n_iter	Max IRLS iterations;
tol	Convergence tolerance for the IRLS parameter change (Euclidean norm of the difference in [a0, beta]), default 1e-5.
standardize	Forwarded to glmnet (ignored if prestandardize = TRUE).
lambda	Optional fixed lambda; if NULL, chosen by choose.
phi_init	Initial precision (phi).
update_phi	Logical; update phi inside the IRLS loop.
phi_maxit	Newton steps for phi update.
prestandardize	If TRUE, manually center/scale X once and disable glmnet's internal standardization (speed trick).
trace	Logical; print IRLS progress.

Value

Named numeric vector (Intercept) + colnames(X) with zeros for unselected variables.

See Also

glmnet::glmnet(), glmnet::cv.glmnet()

Examples

```
set.seed(1); X <- matrix(rnorm(500), 100, 5); Y <- plogis(X[,1]-0.5*X[,3])
Y <- rbeta(100, Y*40, (1-Y)*40)
betareg_glmnet(X, Y, alpha = 1, choose = "bic", prestandardize = TRUE)
```

---

betareg_lasso_gamlss *Beta regression LASSO via GAMLSS*

Description

Uses gamlss::ri() (L1 penalty) in a gamlss(dist = BE) mean submodel to select variables. The helper works on complete cases of X/Y, targets the mean component, and does not yet expose offset handling.

Usage

```
betareg_lasso_gamlss(
  X,
  Y,
  method = c("ML", "GAIC"),
  k = 2,
  degf = NULL,
  lambda = NULL,
  trace = FALSE
)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
method	"ML" or "GAIC" (see gamlss::ri).
k	Penalty multiplier for GAIC when method = "GAIC".
degf	Optional degrees of freedom for the L1 term.
lambda	Optional penalty strength.
trace	Logical; print stepwise trace.

Value

Named numeric vector of coefficients (Intercept) + colnames(X), with 0 for unselected variables.

See Also

gamlss::gamlss(), gamlss::ri(), gamlss.dist::BE()

Examples

```
set.seed(1); X <- matrix(rnorm(300), 100, 3); Y <- plogis(X[,1]); Y <- rbeta(100, Y*30, (1-Y)*30)
betareg_lasso_gamlss(X, Y, method = "GAIC", k = 2)
```

---

betareg_step_aic *Stepwise Beta regression by AIC*

Description

Fits a Beta regression with optional joint selection of the mean and precision (phi) submodels using betareg::betareg(). The routine performs greedy forward/backward search using the requested information criterion and returns coefficients aligned with the supplied design matrix. The selectors currently target the mean submodel only, require complete cases, and do not expose offsets. Observation weights are passed through to betareg() when provided.

Usage

```
betareg_step_aic(
  X,
  Y,
  direction = "both",
  link = "logit",
  link.phi = "log",
  type = "ML",
  trace = FALSE,
  max_steps = NULL,
  epsilon = 1e-08,
  X_phi = NULL,
  direction_phi = c("none", "both", "forward", "backward"),
  weights = NULL
)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
direction	Stepwise direction for the mean submodel: "both", "forward", or "backward".
link	Link for the mean submodel (passed to betareg). Default "logit".
link.phi	Link for precision parameter. Default "log".
type	Likelihood type for betareg, e.g. "ML".
trace	Logical; print stepwise trace.
max_steps	Integer; maximum number of greedy steps (default p).
epsilon	Numeric; minimum improvement required to accept a move (default 1e-8).
X_phi	Optional matrix of candidate predictors for the precision (phi) submodel. When direction_phi enables precision updates and X_phi is NULL, the function reuses X.
direction_phi	Stepwise direction for the precision submodel. Defaults to "none" (no phi selection). Supported values mirror direction.
weights	Optional non-negative observation weights passed to betareg().

Value

Named numeric vector of length p_mean + p_phi + 1 containing the intercept, mean coefficients, phi-intercept (prefixed by "phi|"), and phi coefficients (also prefixed by "phi|"). Non-selected variables have coefficient 0.

See Also

betareg::betareg()

Examples

```
set.seed(1)
X <- matrix(rnorm(200), 100, 2);
Y <- plogis(0.5 + X[,1]-X[,2]);
betareg_step_aic(X, Y)
Y <- rbeta(100, Y*20, (1-Y)*20)
betareg_step_aic(X, Y)
```

`betareg_step_aicc` *Stepwise Beta regression by AICc (finite-sample corrected AIC)*

Description

Greedy forward/backward search minimizing AICc computed on `betareg` fits with optional precision-submodel selection and observation weights.

Usage

```
betareg_step_aicc(
  X,
  Y,
  direction = "both",
  link = "logit",
  link.phi = "log",
  type = "ML",
  trace = FALSE,
  max_steps = NULL,
  epsilon = 1e-08,
  X_phi = NULL,
  direction_phi = c("none", "both", "forward", "backward"),
  weights = NULL
)
```

Arguments

`X`	Numeric matrix (n × p) of mean-submodel predictors.
`Y`	Numeric response in (0,1). Values are squeezed to (0,1) internally.
`direction`	Stepwise direction for the mean submodel: `"both"`, `"forward"`, or `"backward"`.
`link`	Link for the mean submodel (passed to `betareg`). Default `"logit"`.
`link.phi`	Link for precision parameter. Default `"log"`.
`type`	Likelihood type for `betareg`, e.g. `"ML"`.
`trace`	Logical; print stepwise trace.
`max_steps`	Maximum number of greedy steps (default `p`).
`epsilon`	Minimal AICc improvement to accept a move.

X_phi	Optional matrix of candidate predictors for the precision (phi) submodel. When direction_phi enables precision updates and X_phi is NULL, the function reuses X.
direction_phi	Stepwise direction for the precision submodel. Defaults to "none" (no phi selection). Supported values mirror direction.
weights	Optional non-negative observation weights passed to betareg().

Value

See betareg_step_aic().

Examples

```
set.seed(1);
X <- matrix(rnorm(400), 100, 4);
Y <- plogis(X[,1]+0.5*X[,2])
betareg_step_aicc(X, Y)
Y <- rbeta(100, Y*25, (1-Y)*25);
betareg_step_aicc(X, Y)
```

---

betareg_step_bic *Stepwise Beta regression by BIC*

Description

Stepwise Beta regression by BIC

Usage

```
betareg_step_bic(
  X,
  Y,
  direction = "both",
  link = "logit",
  link.phi = "log",
  type = "ML",
  trace = FALSE,
  max_steps = NULL,
  epsilon = 1e-08,
  X_phi = NULL,
  direction_phi = c("none", "both", "forward", "backward"),
  weights = NULL
)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
direction	Stepwise direction for the mean submodel: "both", "forward", or "backward".
link	Link for the mean submodel (passed to betareg). Default "logit".
link.phi	Link for precision parameter. Default "log".
type	Likelihood type for betareg, e.g. "ML".
trace	Logical; print stepwise trace.
max_steps	Integer; maximum number of greedy steps (default p).
epsilon	Numeric; minimum improvement required to accept a move (default 1e-8).
X_phi	Optional matrix of candidate predictors for the precision (phi) submodel. When direction_phi enables precision updates and X_phi is NULL, the function reuses X.
direction_phi	Stepwise direction for the precision submodel. Defaults to "none" (no phi selection). Supported values mirror direction.
weights	Optional non-negative observation weights passed to betareg().

Value

See betareg_step_aic().

Examples

```
set.seed(1); X <- matrix(rnorm(300), 100, 3);
Y <- plogis(X[,1]);
betareg_step_bic(X, Y)
Y <- rbeta(100, Y*30, (1-Y)*30)
betareg_step_bic(X, Y)
```

---

compare_selectors_bootstrap

Bootstrap selection frequencies across selectors

Description

Bootstraps the dataset B times and records how often each variable is selected by each selector. Observations containing NA in either X or Y are removed prior to resampling. Column names are abbreviated internally and mapped back to the originals in the output just like in compare_selectors_single().

Usage

```
compare_selectors_bootstrap(X, Y, B = 50, include_enet = TRUE, seed = NULL)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
B	Number of bootstrap replications.
include_enet	Logical; include ENet if gamlss.lasso is installed.
seed	Optional RNG seed.

Value

Long data frame with columns selector, variable, freq in [0,1], n_success, and n_fail. The freq column reports the share of bootstrap replicates where a variable was selected by the corresponding selector. Values near 1 signal high stability whereas small values indicate weak evidence. n_success counts the successful fits contributing to the frequency estimate (excluding failed replicates), while n_fail records the number of unsuccessful fits. A "failures" attribute attached to the returned data frame lists the replicate indices and messages for any encountered errors.

Examples

```
set.seed(1)
X <- matrix(rnorm(300), 100, 3); Y <- plogis(X[, 1])
Y <- rbeta(100, Y * 30, (1 - Y) * 30)
freq <- compare_selectors_bootstrap(X, Y, B = 10, include_enet = FALSE)
head(freq)
subset(freq, freq > 0.8)


# Increase B until the reported frequencies stabilise. For example,
freq_big <- compare_selectors_bootstrap(X, Y, B = 200, include_enet = FALSE)
stats::aggregate(freq ~ selector, freq_big, summary)
```

---

compare_selectors_single

Run all selectors once on a dataset

Description

Convenience wrapper that runs AIC/BIC/AICc stepwise, GAMLSS LASSO (and ENet when available), and the pure glmnet IRLS selector, then collates coefficients into a long table for comparison. Observations containing NA in either X or Y are removed prior to fitting. Column names are temporarily shortened to satisfy selector requirements and avoid clashes; the outputs remap them to the original labels before returning so the reported variables always match the input design.

Usage

```
compare_selectors_single(X, Y, include_enet = TRUE)
```

Arguments

X	Numeric matrix (n × p) of mean-submodel predictors.
Y	Numeric response in (0,1). Values are squeezed to (0,1) internally.
include_enet	Logical; include ENet if gamlss.lasso is installed.

Value

A list with:

coefs Named coefficient vectors for each selector.

table Long data frame with columns selector, variable, coef, selected.

Examples

```
set.seed(1)
X <- matrix(rnorm(300), 100, 3); Y <- plogis(X[, 1])
Y <- rbeta(100, Y * 30, (1 - Y) * 30)
single <- compare_selectors_single(X, Y, include_enet = FALSE)
head(single$table)
```

---

compare_table *Merge single-run results and bootstrap frequencies*

Description

Merge single-run results and bootstrap frequencies

Usage

```
compare_table(single_tab, freq_tab = NULL)
```

Arguments

single_tab	Data frame returned in compare_selectors_single()[["table"]].
freq_tab	Optional frequency table from compare_selectors_bootstrap().

Value

Merged data frame.

Examples

```
single_tab <- data.frame(
  selector = rep(c("AIC", "BIC"), each = 3),
  variable = rep(paste0("x", 1:3), times = 2),
  coef = c(0.5, 0, -0.2, 0.6, 0.1, -0.3)
)
single_tab$selected <- single_tab$coef != 0
freq_tab <- data.frame(
  selector = rep(c("AIC", "BIC"), each = 3),
  variable = rep(paste0("x", 1:3), times = 2),
  freq = c(0.9, 0.15, 0.4, 0.85, 0.3, 0.25)
)
compare_table(single_tab, freq_tab)
```

---

fastboost_interval *Interval-response stability selection (fastboost variant)*

Description

Repeats selection on interval-valued responses by sampling a pseudo-response from each interval (uniformly or midpoint), tallying variable selection frequencies across `B` replicates.

Usage

```
fastboost_interval(
  X,
  Y_low,
  Y_high,
  func,
  B = 100,
  sample = c("uniform", "midpoint"),
  version = "glmnet",
  use.parallel = FALSE,
  seed = NULL,
  ...
)
```

Arguments

`X`	Numeric matrix (n × p).
`Y_low`, `Y_high`	Interval bounds in [0,1]. Rows with missing bounds are dropped.
`func`	Function `function(X, y, ...)` returning a named coefficient vector as in the other selectors (nonselected = 0).
`B`	Number of interval resamples.
`sample`	`"uniform"` (default) or `"midpoint"` for drawing pseudo-responses.
`version`	Ignored (reserved for future).

use.parallel	Use parallel::mclapply if available.
seed	Optional RNG seed. Scoped via withr::with_seed() so the caller's RNG state is restored afterwards.
...	Extra args forwarded to func.

Value

A list with:

betas B × (p+1) matrix of coefficients over replicates.

freq Named vector of selection frequencies for each predictor.

Examples

```
# suppose you have interval data (Y_low, Y_high)
set.seed(1)
n <- 120; p <- 6
X <- matrix(rnorm(n*p), n, p); colnames(X) <- paste0("x",1:p)
mu <- plogis(X[,1] - 0.5*X[,2]); Y <- rbeta(n, mu*25, (1-mu)*25)
Y_low <- pmax(0, Y - 0.05); Y_high <- pmin(1, Y + 0.05)
fb <- fastboost_interval(X, Y_low, Y_high,
       func = function(X,y) betareg_glmnet(X,y, choose="bic", prestandardize=TRUE),
       B = 40)
sort(fb$freq, decreasing = TRUE)
```

---

plot_compare_coeff *Side-by-side coefficient heatmap*

Description

Visual comparison of coefficients returned by each selector. Requires ggplot2.

Usage

```
plot_compare_coeff(single_tab)
```

Arguments

single_tab	Data frame as returned by compare_selectors_single()[["table"]].

Value

A ggplot object when ggplot2 is available; otherwise draws a base R image.

Examples

```
demo_tab <- data.frame(
  selector = rep(c("AIC", "BIC"), each = 3),
  variable = rep(paste0("x", 1:3), times = 2),
  coef = c(0.6, 0, -0.2, 0.55, 0.05, -0.3)
)
demo_tab$selected <- demo_tab$coef != 0
plot_compare_coeff(demo_tab)
```

`plot_compare_freq` *Side-by-side selection-frequency heatmap*

Description

Visual comparison of bootstrap selection frequencies by selector. Requires `ggplot2`.

Usage

```
plot_compare_freq(freq_tab)
```

Arguments

`freq_tab`	Data frame as returned by `compare_selectors_bootstrap()`.

Value

A `ggplot` object when `ggplot2` is available; otherwise draws a base R image.

Examples

```
freq_tab <- data.frame(
  selector = rep(c("AIC", "BIC"), each = 3),
  variable = rep(paste0("x", 1:3), times = 2),
  freq = c(0.85, 0.2, 0.45, 0.75, 0.35, 0.3)
)
plot_compare_freq(freq_tab)
```

`sb_apply_selector_manual` *Apply a selector to a collection of resampled designs*

Description

Apply a selector to a collection of resampled designs

Usage

```
sb_apply_selector_manual(
  X_norm,
  resamples,
  Y,
  selector,
  ...,
  keep_template = TRUE
)
```

Arguments

`X_norm`	Normalised design matrix.
`resamples`	List of matrices returned by `sb_resample_groups()`.
`Y`	Numeric response.
`selector`	Variable-selection routine; function or character string. If it is a function, the selector name should be added as the fun.name attribute.
`...`	Extra arguments passed to the selector.
`keep_template`	Logical; when `TRUE` (default) the first column stores the coefficients fitted on `X_norm` before any resampling. This avoids recomputing the selector on the original data and keeps the baseline fit available for diagnostics.

Value

A numeric matrix of coefficients with one column per resample.

`sb_beta` *SelectBoost for beta-regression models*

Description

`sb_beta()` orchestrates all SelectBoost stages—normalisation, correlation analysis, grouping, correlated resampling, and stability tallying—while using the beta-regression selectors provided by this package. It can operate on point-valued or interval-valued responses and automatically squeezes the outcome into `(0, 1)` unless instructed otherwise.

Usage

```
sb_beta(
  X,
  Y = NULL,
  selector = betareg_step_aic,
  corrfunc = "cor",
  B = 100,
  step.num = 0.1,
  steps.seq = NULL,
  version = c("glmnet", "lars"),
  squeeze = TRUE,
  use.parallel = FALSE,
  seed = NULL,
  verbose = FALSE,
  threshold = 1e-04,
  interval = c("none", "uniform", "midpoint"),
  Y_low = NULL,
  Y_high = NULL,
  ...
)
```

Arguments

X	Numeric design matrix. Coerced with as.matrix() and normalised via sb_normalize().
Y	Numeric response vector. Values are squeezed to the open unit interval with the standard SelectBoost transformation unless squeeze = FALSE. Optional when interval bounds are supplied.
selector	Selection routine. Defaults to betareg_step_aic(). Function or character string. If it is a function, the selector name should be added as the fun.name attribute.
corrfunc	Correlation function passed to sb_compute_corr().
B	Number of replicates to generate.
step.num	Step length for the automatically generated c0 grid.
steps.seq	Optional user-supplied grid of absolute correlation thresholds.
version	Either "glmnet" (intercept in first row) or "lars".
squeeze	Logical; ensure the response lies in (0, 1).
use.parallel	Logical; enable parallel resampling and selector fits when supported by the current R session.
seed	Optional integer seed for reproducibility. The seed is scoped via withr::with_seed() so the caller's RNG state is restored on exit.
verbose	Logical; emit progress messages.
threshold	Numeric tolerance for considering a coefficient selected.
interval	Interval-resampling mode: "none" reuses Y, whereas "uniform" and "midpoint" draw pseudo-responses between Y_low and Y_high for each replicate.

Y_low, Y_high	Interval bounds in [0, 1] paired with the rows of X when interval is not "none".
...	Additional arguments forwarded to selector.

Details

The returned object carries a rich set of attributes:

- "c0.seq" – the grid of absolute-correlation thresholds explored during resampling.
- "steps.seq" – the raw sequence (if any) used to construct the grid.
- "selector" – the selector identifier (function name or expression).
- "B" – number of resampled designs passed to the selector.
- "interval" – the interval sampling mode ("none", "uniform", or "midpoint").
- "resample_diagnostics" – per-threshold data frames with summary statistics on the cached correlated draws.

These attributes mirror the historical SelectBoost beta implementation so the object can be consumed by existing plotting and reporting utilities.

Value

Matrix of selection frequencies with one row per c0 level and class "sb_beta". See *Details* for the recorded attributes.

Examples

```
set.seed(42)
sim <- simulation_DATA.beta(n = 80, p = 4, s = 2)
# increase B for real applications
res <- sb_beta(sim$X, sim$Y, B = 5)
res
```

---

sb_beta_interval *SelectBoost workflow for interval responses*

Description

sb_beta_interval() forwards to sb_beta() while activating interval sampling so that beta-regression SelectBoost runs can ingest lower/upper response bounds directly. It mirrors fastboost_interval() but reuses the correlated resampling pipeline of sb_beta().

Usage

```
sb_beta_interval(
  X,
  Y_low,
  Y_high,
  selector = betareg_step_aic,
  sample = c("uniform", "midpoint"),
  Y = NULL,
  ...
)
```

Arguments

X	Numeric design matrix. Coerced with as.matrix() and normalised via sb_normalize().
Y_low, Y_high	Interval bounds in [0, 1] paired with the rows of X when interval is not "none".
selector	Selection routine. Defaults to betareg_step_aic(). Function or character string. If it is a function, the selector name should be added as the fun.name attribute.
sample	Interval sampling scheme passed to the interval argument of sb_beta(). "uniform" draws a pseudo-response uniformly within each interval; "midpoint" always chooses the midpoint.
Y	Optional point-valued response. Supply it when you wish to keep the observed mean response but still resample within Y_low/Y_high for the stability steps.
...	Additional arguments forwarded to selector.

Value

See sb_beta(). The returned object carries the same "sb_beta"-class attributes describing the correlation thresholds, resampling diagnostics, selector, and number of replicates.

Examples

```
set.seed(1)
sim <- simulation_DATA.beta(n = 120, p = 5, s = 2)
y_low <- pmax(sim$Y - 0.05, 0)
y_high <- pmin(sim$Y + 0.05, 1)
interval_fit <- sb_beta_interval(
  sim$X,
  Y_low = y_low,
  Y_high = y_high,
  B = 5,
  step.num = 0.4
)
attr(interval_fit, "interval")
```

sb_beta_methods *User-friendly methods for* sb_beta() *results*

Description

These S3 helpers make it easier to inspect and visualise the correlation-threshold grid returned by sb_beta(). They surface the stored attributes, reshape the selection frequencies into tidy summaries, and produce quick ggplot2 visualisations for interactive use.

Usage

```
## S3 method for class 'sb_beta'
print(x, digits = 3, ...)

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

## S3 method for class 'summary.sb_beta'
print(x, digits = 3, n = 10, ...)

autoplot.sb_beta(object, variables = NULL, ...)
```

Arguments

x, object	An object of class sb_beta.
digits	Number of decimal places to display when printing.
...	Additional arguments passed on to lower-level methods.
n	Number of rows to show from the summary table when printing.
variables	Optional character vector of variables to retain in the plotted output.

Value

summary.sb_beta() returns an object of class summary.sb_beta containing a tidy data frame of selection frequencies. The plotting and printing methods are invoked for their side effects and return the input object invisibly.

Examples

```
set.seed(42)
sim <- simulation_DATA.beta(n = 50, p = 4, s = 2)
fit <- sb_beta(sim$X, sim$Y, B = 5, step.num = 0.5)
print(fit)
summary(fit)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  autoplot.sb_beta(fit)
}
```

sb_normalize *Core helpers for SelectBoost-style beta regression*

Description

These helpers expose the individual stages of the SelectBoost workflow so that beta-regression selectors can be combined with correlation-aware resampling directly from `SelectBoost.beta`. They normalise the design matrix, derive correlation structures, form groups of correlated predictors, generate Gaussian surrogates that mimic the observed dependency structure, and apply a user-provided selector on each resampled design.

Usage

```
sb_normalize(X, center = NULL, scale = NULL, eps = 1e-08)

sb_compute_corr(X, corrfunc = "cor")

sb_group_variables(corr_mat, c0)
```

Arguments

`X`	Numeric matrix of predictors.
`center`	Optional centering vector recycled to the number of columns. Defaults to the column means of `X`.
`scale`	Optional scaling vector recycled to the number of columns. Defaults to the column-wise ℓ_2 norms of the centred matrix.
`eps`	Small positive constant used when normalising columns.
`corrfunc`	Function or character string used to compute pairwise associations. Defaults to `"cor"`.
`corr_mat`	Numeric matrix of associations.
`c0`	Threshold applied to the absolute correlations.

Value

`sb_normalize()` returns a centred, ℓ_2-scaled copy of `X`.

`sb_compute_corr()` returns the association matrix.

`sb_group_variables()` returns a list of integer vectors, one per variable, describing the correlated group it belongs to.

Examples

```
sb_normalize(matrix(rnorm(20), 5))
```

sb_resample_groups *Generate correlated design replicates for a set of groups*

Description

Generate correlated design replicates for a set of groups

Usage

```
sb_resample_groups(
  X_norm,
  groups,
  B = 100,
  jitter = 1e-06,
  seed = NULL,
  use.parallel = FALSE,
  cache = NULL
)
```

Arguments

X_norm	Normalised design matrix.
groups	Correlation structure. Either a list as returned by sb_group_variables() or a vector of group labels matching the columns of X_norm.
B	Number of replicates to generate.
jitter	Numeric value added to covariance diagonals for stability.
seed	Optional integer seed for reproducibility. The seed is scoped via withr::with_seed() so the caller's RNG state is restored on exit.
use.parallel	Logical; when TRUE, compute the resampled designs using a parallel backend when available.
cache	Optional environment or named list used to cache previously generated surrogates. Passing the same cache across calls reuses draws for identical groups.

Details

When every group has size one (no correlated variables) the function simply returns B copies of X_norm. A warning is issued in that situation so downstream code can avoid mistaking the replicated designs for genuinely resampled surrogates. The covariance matrices underpinning each correlated draw are cached in the supplied cache environment; reusing the environment across calls lets sb_resample_groups() skip redundant covariance decompositions for identical groups and speeds up iterative workflows.

Value

An object of class sb_resamples, i.e. a list of length B whose elements are resampled design matrices. The object exposes per-group diagnostics in its "diagnostics" attribute and returns the cache via the "cache" attribute for reuse.

sb_selection_frequency

Compute selection frequencies from coefficient paths

Description

Compute selection frequencies from coefficient paths

Usage

```
sb_selection_frequency(
  coef_matrix,
  version = c("glmnet", "lars"),
  threshold = 1e-04
)
```

Arguments

coef_matrix	Matrix produced by sb_apply_selector_manual().
version	Either "glmnet" (first row is intercept) or "lars".
threshold	Coefficients with absolute value below this threshold are treated as zero.

Value

Numeric vector of selection frequencies.

simulation_DATA.beta *Simulate* **interval** *Beta-regression data (flexible)*

Description

Simulate **interval** Beta-regression data (flexible)

Usage

```
simulation_DATA.beta(
  n,
  p,
  s = min(5L, p),
  beta_size = 1,
  a0 = 0,
  X_dist = c("gaussian", "t", "bernoulli"),
  corr = c("indep", "ar1", "block"),
  rho = 0,
  block_size = 5L,
```

```
  df = 5,
  prob = 0.5,
  active_idx = NULL,
  phi = 20,
  mechanism = c("jitter", "quantile", "mixed"),
  mix_prob = 0.5,
  delta = 0.05,
  delta_low = NULL,
  delta_high = NULL,
  alpha = 0.1,
  alpha_low = NULL,
  alpha_high = NULL,
  na_rate = 0,
  na_side = c("random", "left", "right"),
  centerX = FALSE,
  scaleX = FALSE,
  seed = NULL
)
```

Arguments

n, p	Sample size and number of predictors.
s	Number of active (nonzero) coefficients.
beta_size	Scalar (alternating ±) or numeric vector of length greater then equal s.
a0	Intercept (logit scale).
X_dist	Distribution for X: "gaussian", "t", or "bernoulli".
corr	Correlation structure: "indep", "ar1", or "block".
rho	AR(1) correlation or within-block correlation.
block_size	Block size when corr = "block" (default 5).
df	Degrees of freedom for X_dist = "t" (default 5).
prob	Success prob for X_dist = "bernoulli" (default 0.5).
active_idx	Optional integer vector of active feature indices (length s). If NULL, uses 1:s.
phi	Precision parameter: scalar, length-n vector, or function (mu, X) -> length-n.
mechanism	Interval mechanism per row: "jitter", "quantile", or "mixed".
mix_prob	Probability of jitter when mechanism = "mixed".
delta	Symmetric jitter half-width (scalar / vector / function).
delta_low, delta_high	Asymmetric jitter widths (override delta if set).
alpha	Miscoverage for quantile intervals (scalar / vector / function).
alpha_low, alpha_high	Asymmetric miscoverage (override alpha if set).
na_rate	Fraction of rows with a missing bound (default 0).
na_side	Which bound to drop: "left", "right", or "random".
centerX, scaleX	Whether to center/scale X before returning.
seed	RNG seed.

Value

list with `X`, `Y`, `Y_low`, `Y_high`, `mu`, `beta`, `a0`, `phi`, `info`, `settings`.

Index