

Package ‘spareg’

September 25, 2026

Type Package

Title Sparse Projected Averaged Regression

Version 1.2.0

Description A flexible framework combining variable screening and random projection techniques for fitting ensembles of predictive generalized linear models to high-dimensional data. Designed for extensibility, the package implements key techniques as S3 classes with user-friendly constructors, enabling easy integration and development of new procedures for high-dimensional applications.

License GPL-3

Imports Matrix, ROCR, Rdpack, ggplot2, rlang, glmnet, methods

RdMacros Rdpack

Encoding UTF-8

URL <https://github.com/lauravana/spareg>

BugReports <https://github.com/lauravana/spareg/issues>

Suggests testthat (>= 3.0.0), foreach, doParallel, doRNG, robustbase, cellWise, VariableScreening, ggpubr, R.matlab

Config/testthat/edition 3

Depends R (>= 4.0.0)

Config/roxygen2/version 8.1.0

NeedsCompilation no

Author Laura Vana-Gür [aut, cre] (ORCID: <https://orcid.org/0000-0002-9613-7604>),
Roman Parzer [aut] (ORCID: <https://orcid.org/0000-0003-0893-3190>),
Peter Filzmoser [aut] (ORCID: <https://orcid.org/0000-0002-8014-4682>)

Maintainer Laura Vana-Gür <laura.vana.guer@tuwien.ac.at>

Repository CRAN

Date/Publication 2026-09-25 16:50:02 UTC

Contents

coef.spar	2
coef.spar.cv	4
constructor_randomprojection	5
constructor_screencoef	7
constructor_sparmodel	8
get_coef	9
get_intercept	10
get_measure	11
get_model	11
get_model.spar	12
get_model.spar.cv	13
plot.spar	14
plot.spar.cv	15
predict.spar	17
predict.spar.cv	18
print.coefspar	19
print.randomprojection	20
print.screencoef	21
print.spar	21
print.spar.cv	22
randomprojection-class	23
rp_cw	24
rp_gaussian	26
rp_sparse	27
screencoef-class	28
screen_cor	30
screen_glmnet	31
screen_marglik	32
simulate_spareg_data	33
spar	34
spar.cv	38
sparmodel-class	41
spar_glm	42
spar_glmnet	43
summary.coefspar	44

Index	45
--------------	-----------

coef.spar	<i>Coef Method for 'spar' Object</i>
-----------	--------------------------------------

Description

Extracts coefficients from 'spar' object

Usage

```
## S3 method for class 'spar'
coef(
  object,
  nummod = NULL,
  nu = NULL,
  aggregate = c("mean", "median", "none"),
  ...
)
```

Arguments

object	result of spar of class 'spar'.
nummod	number of models used to form coefficients; value with minimal validation measure is used if not provided.
nu	threshold level used to compute the coefficients; value with minimal validation measure is used if not provided.
aggregate	character, one of c("mean", "median", "none"), giving the method of aggregating the coefficients over the marginal models. If set to "none", the coefficients are not aggregated over the marginal models and a matrix of coefficients, one column for each marginal model, is returned. Otherwise the coefficients are aggregated using the specified method (mean or median). Defaults to mean aggregation.
...	further arguments passed to or from other methods.

Value

Returns an object of class 'coefspar', which is a list with elements

- intercept: The average intercept value or vector intercepts (one for each marginal model) if aggregate = "none".
- beta: Vector of length p of averaged coefficients or a $p \times \max(\text{nummods})$ matrix of coefficients if aggregate = "none".
- nummod: Number of models based on which the coefficients are computed.
- nu: Threshold value based on which the coefficients are computed.

See Also

[print.coefspar](#), [summary.coefspar](#)

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
coef(spar_res)
coef(spar_res, aggregate = "median")
coef(spar_res, aggregate = "none")
coef(spar_res, nummod = 5, nu = 0)
```

coef.spar.cv

*Coef Method for 'spar.cv' Object***Description**

Extract coefficients from 'spar.cv' object. This function extracts coefficients only if `precompute_mode = "precompute_all"`. In this case, coefficients are derived from the initial run on the full dataset using the optimal M and ν combination identified via cross-validation (best or 1se rule). For other modes ("`precompute_proj`" or "`full_cv`"), coefficients cannot be directly extracted because screening or projections vary across folds. To obtain coefficients in these cases, refit the model on the full dataset using `spar()` with the selected M and ν values.

Usage

```
## S3 method for class 'spar.cv'
coef(
  object,
  nummod = NULL,
  nu = NULL,
  opt_par = c("best", "1se"),
  aggregate = c("mean", "median", "none"),
  ...
)
```

Arguments

object	result of spar.cv function of class 'spar.cv'. Refitting can be done using <code>get_model()</code> .
nummod	optional number of models used to form coefficients.
nu	optional threshold level used to form coefficients.
opt_par	one of <code>c("1se", "best")</code> , chooses whether to select the best pair of nus and nummods according to cross-validated (CV) measure, or the sparsest solution within one sd of that optimal CV measure; ignored when nummod and nu are given.
aggregate	character one of <code>c("mean", "median", "none")</code> . If set to "none" the coefficients are not aggregated over the marginal models, otherwise the coefficients are aggregated using the specified method (mean or median). Defaults to mean aggregation.
...	further arguments passed to or from other methods.

Value

List with elements

- intercept average intercept value or vector intercepts (one for each marginal model) if `aggregate = "none"`.

- beta vector of length p of averaged coefficients or a $p \times \max(\text{nummods})$ matrix of coefficients if `agregate = "none"`.
- `nummod` number of models based on which the coefficient is computed.
- `nu` threshold based on which the coefficients is computed.

See Also

[predict.spar.cv](#), [get_model.spar.cv](#)

Examples

```
example_data <- simulate_spareg_data(n = 80, p = 200, ntest = 100)
spar_res <- spar.cv(example_data$x, example_data$y, nfolds = 3L,
  nummods = c(5, 10))
coef(spar_res)
```

constructor_randomprojection

Constructor Function for Building 'randomprojection' Object

Description

Creates an object class 'randomprojection' using arguments passed by user.

Usage

```
constructor_randomprojection(
  name = NULL,
  generate_fun,
  update_fun = update_rp_default,
  update_rpm_w_data = update_rpm_identity,
  control = list()
)
```

Arguments

- | | |
|--------------|--|
| name | optional string describing the random projection method. This is used for printing. |
| generate_fun | <p>A function for generating the random projection matrix. This function must accept the following arguments:</p> <ul style="list-style-type: none"> • <code>object</code>: An object of class randomprojection. • <code>x</code>: A matrix of standardized predictors. • <code>y</code>: A vector of standardized responses. • <code>m</code>: The target dimension for the projection. |

	• <code>included_vector</code>: A vector of column indices for the variables to be included in the projection. • <code>...</code>: Additional arguments passed from <code>spar()</code> or other functions, due to the fact that screening is employed pre-projection.
<code>update_fun</code>	A function for updating the <code>randomprojection</code> object with data-specific information. This function must accept: • <code>object</code>: An object of class <code>randomprojection</code>. • <code>x</code>: A matrix of standardized predictors. • <code>y</code>: A vector of standardized responses. • <code>family</code>: A <code>stats::family</code> object. • <code>...</code>: Additional arguments passed from <code>spar()</code>. If not provided, the default <code>update_rp_default</code> is used.
<code>update_rpm_w_data</code>	A function for updating an already-generated random projection matrix with data-dependent information. This function must accept: • <code>rpm</code>: The random projection matrix to update. • <code>object</code>: An object of class <code>randomprojection</code>. • <code>included_vector</code>: A vector of column indices for the variables included in the projection. • <code>x</code>: A matrix of standardized predictors. • <code>y</code>: A vector of standardized responses. • <code>family</code>: A <code>stats::family</code> object. • <code>...</code>: Additional arguments passed from <code>spar()</code>. If not provided, the default <code>update_rpm_identity</code> is used, which leaves the matrix unchanged.
<code>control</code>	A list of control parameters for the random projection. Default: <code>list()</code>. These parameters are passed to <code>generate_fun</code>, <code>update_fun</code>, and <code>update_rpm_w_data</code>.

Details

The `update_rpm_w_data` function is particularly relevant for cross-validation procedures where random projection matrices are precomputed (e.g., `precompute_mode = "precompute_all"` or `precompute_mode = "precompute_rpm"`). In such cases, the cross-validation procedure uses the precomputed matrices, but you may want to update data-dependent entries (e.g., diagonal elements) with the training data in each fold. For example, in `rp_cw(data = TRUE)`, the diagonal elements of the projection matrices are updated to reflect screening coefficients computed on the training data for each fold, while the random elements remain unchanged.

Value

Returns a function that, when called, creates and returns an object of class `randomprojection`.

Examples

```
generate_cauchy <- function(object, x, y, m, included_vector, ...) {
  p <- length(included_vector)
  control_rcauchy <- c(object$control[names(object$control) %in% names(formals(rcauchy))],
```

```

      attributes(object)[names(attributes(object)) %in% names(formals(rcauchy))])
control_rcauchy <- control_rcauchy[!duplicated(names(control_rcauchy))]
vals <- do.call(function(...)
  rcauchy(m * p, ...), control_rcauchy)
RM <- matrix(vals, nrow = m, ncol = p)
return(RM)
}
rp_cauchy <- constructor_randomprojection(
  generate_fun = generate_cauchy, name = "rp_cauchy")
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, rp = rp_cauchy(scale = 1/400))
spar_res

```

constructor_screencoef

Constructor Function for Building [screencoef](#) Objects

Description

The created function will return a object of class [screencoef](#) which constitutes of a list. The attributes of the generating object will include by default type, which can take one of two values "prob" (indicating probabilistic screening should be employed), "fixed" (indicating that the top nscreen variables should be employed).

Usage

```

constructor_screencoef(
  name = NULL,
  generate_fun,
  update_fun = update_screen_default
)

```

Arguments

name	character
generate_fun	function responsible for computing the screening coefficients. This function should have arguments x, y (the predictor matrix and response vector supplied by spar()) where both have already been standardized by spar(), 'screencoef' object and ..., whereas all other potentially relevant arguments of spar() are passed internally to this function through the ellipsis.
update_fun	optional function for updating the screencoef object with information from the data passed to spar(). This function should have arguments object, which is a screencoef object, x, y (the predictor matrix and response vector supplied by spar()) where both have already been standardized by spar(), family and ..., whereas all other potentially relevant arguments of spar() are passed internally to this function through the ellipsis. If update_fun is not provided, the object remains unchanged.

Details

Creates an object class `screencoef` using arguments passed by user.

Value

Returns a function that, when called, creates and returns an object of class 'screencoef'.

Examples

```
generate_scr_sirs <- function(y, x, object, ...) {
  ctrl <- object$control[names(object$control) %in%
    names(formals(VariableScreening::screenIID)))]
  res_screen <- do.call(function(...)
    VariableScreening::screenIID(x, y, ...), ctrl)
  res_screen$measurement
}
screen_sirs <- constructor_screencoef("screen_sirs",
  generate_fun = generate_scr_sirs)
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_example <- spar(example_data$x, example_data$y,
  screencoef = screen_sirs(control = list(method = "SIRS")),
  rp = rp_sparse())
spar_example
```

constructor_sparmodel *Constructor Function for Building `sparmodel` Object*

Description

The created function will return a object of class `sparmodel` which constitutes of a list.

Usage

```
constructor_sparmodel(
  name = NULL,
  generate_fun,
  update_fun = update_sparmodel_default
)
```

Arguments

name	optional string describing the model employed. This is used for printing.
generate_fun	function for estimating the marginal models which returns the function should have arguments a <code>sparmodel</code> object, x and y (matrix of predictors and vector of responses supplied by <code>spar()</code> , which have already been standardized), z (the matrix of projected predictors) and ..., whereas all other potentially relevant arguments of <code>spar()</code> are passed internally to this function through the ellipsis.

update_fun optional function for updating the [sparmodel](#) object. This function should have arguments object, which is a 'screencoef' object, x, y (the predictor matrix and response vector supplied by spar() where both have already been standardized by spar()), family and ..., whereas all other potentially relevant arguments of spar() are passed internally to this function through the ellipsis.

Details

Creates an object of class [sparmodel](#) using arguments passed by user.

Value

Returns a function that, when called, creates and returns an object of class [sparmodel](#).

Examples

```
model_glmrob <- function(object, x, y, z, ...) {
  requireNamespace("robustbase")
  glmrob_res <- do.call(function(...)
    robustbase::glmrob(y ~ as.matrix(z), ...),
    object$control)
  intercept <- coef(glmrob_res)[1]
  gammas <- coef(glmrob_res)[-1]
  list(gammas = gammas, intercept = intercept)
}
spar_glmrob <- constructor_sparmodel(
  generate_fun = model_glmrob, name = "glmrob")
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest,
  model = spar_glmrob())
spar_res
```

get_coef

Extractor for Model Coefficients from 'coefspar' Object

Description

Extractor for Model Coefficients from 'coefspar' Object

Usage

```
get_coef(x)
```

Arguments

x A 'coefspar' object.

Value

A numeric vector or matrix of coefficients.

See Also

[coef.spar](#), [coef.spar.cv](#), [print.coefspar](#), [summary.coefspar](#)

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
coefs <- coef(spar_res)
get_coef(coefs)
```

get_intercept

Extractor for Model Intercept from 'coefspar' Object

Description

Extractor for Model Intercept from 'coefspar' Object

Usage

```
get_intercept(x)
```

Arguments

x A 'coefspar' object.

Value

Intercept (numeric or vector).

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
coefs <- coef(spar_res)
get_coef(coefs)
```

get_measure	<i>Extractor for (Cross-)Validation Measure from 'spar' or 'spar.cv' Object</i>
-------------	---

Description

Extractor for (Cross-)Validation Measure from 'spar' or 'spar.cv' Object

Usage

```
get_measure(object)
```

Arguments

object	A fitted 'spar' or 'spar.cv' model
--------	------------------------------------

Value

data.frame containing the (cross-)validation measure for the considered threshold and number of model combinations. For 'spar' objects it contains information about the measure calculated on the validation set (or on the training sample if xval and yval are missing) and the number of active variables. For 'spar.cv' objects it contains information on the average measure obtained across folds together with the standard deviation across the folds and the average number of active variables. the nfolds of the training set.

See Also

[spar](#), [spar.cv](#), [get_model](#)

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
get_measure(spar_res)
```

get_model	<i>get_model</i>
-----------	------------------

Description

Retrieve the Best Model from a Fitted Object

Usage

```
get_model(object, ...)
```

Arguments

object	An object of class <code>spar</code> or <code>spar.cv</code> , typically the result of a model fitting function.
...	Additional arguments passed to the specific method.

Details

This generic function retrieves the best model from a fitted object of class `spar` or `spar.cv`. The method dispatched depends on the class of the input object.

Value

The modified object with the best model selected.

Examples

```
# Example usage with a fitted object

example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
fitted_obj <- spar(example_data$x, example_data$y, family = gaussian(), nfolds = 3L)
best_model <- get_model(fitted_obj)
```

get_model.spar	<i>Get Best Model for spar Objects</i>
----------------	--

Description

Extracts the best model from a fitted `spar` object based on validation results. The best model is determined by the minimum validation measure.

Usage

```
## S3 method for class 'spar'
get_model(object, ...)
```

Arguments

object	An object of class <code>spar</code> , typically the result of <code>spar()</code> .
...	Additional arguments (currently unused).

Value

An updated `spar` object containing:

- `betas`: Coefficients of the best model, with values below `nu` set to 0.
- `intercepts`: Intercepts for the selected number of models (`nummod`).
- `val_res`: Validation results filtered for the best `nummod` and `nu`.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
fitted_spar <- spar(example_data$x, example_data$y, family = gaussian())
best_spar <- get_model(fitted_spar)
```

get_model.spar.cv

Get Best Model for spar.cv Objects

Description

Extracts the best or 1SE model from a fitted `spar.cv` object. The selection is based on cross-validation results, and the model can be re-estimated if required.

Usage

```
## S3 method for class 'spar.cv'
get_model(
  object,
  opt_par = c("best", "1se"),
  x = NULL,
  y = NULL,
  xval = NULL,
  yval = NULL,
  ...
)
```

Arguments

<code>object</code>	An object of class <code>spar.cv</code> , typically the result of <code>spar.cv()</code> .
<code>opt_par</code>	A character string specifying the selection criterion: "best": Selects the model with the minimum validation measure. "1se": Selects the simplest model within 1 standard error of the best model.
<code>x</code>	A matrix or data frame of predictors. Required if <code>precompute_mode != "precompute_all"</code> .
<code>y</code>	A response vector. Required if <code>precompute_mode != "precompute_all"</code> .
<code>xval</code>	A matrix or data frame of validation predictors. If <code>NULL</code> , <code>x</code> is used.
<code>yval</code>	A validation response vector. If <code>NULL</code> , <code>y</code> is used.
<code>...</code>	Additional arguments passed to <code>spar()</code> for re-estimation.

Value

An updated `spar` or `spar.cv` object containing:

- `betas`: Coefficients of the selected model.
- `intercepts`: Intercepts for the selected model.
- `val_res`: Validation results filtered for the selected `nummod` and `nu`.
- Additional fields like `xscale`, `yscale`, `xcenter`, and `ycenter` if `precompute_mode == "precompute_all"`.

Examples

```
example_data <- simulate_spareg_data(n = 50, p = 200, ntest = 50)
fitted_spar_cv <- spar.cv(example_data$x, example_data$y, family = gaussian(), nfolds = 3L)
best_spar_cv <- get_model(fitted_spar_cv, opt_par = "best")
```

plot.spar

Plot Method for 'spar' Object

Description

Creates diagnostic plots for a 'spar' object, including:

- Validation measure vs. nus or nummods.
- Number of active variables vs. nus or nummods.
- Residuals vs. fitted values.
- Coefficient heatmap.

Usage

```
## S3 method for class 'spar'
plot(
  x,
  plot_type = c("val_measure", "val_numactive", "res_vs_fitted", "coefs"),
  plot_along = c("nu", "nummod"),
  nummod = NULL,
  nu = NULL,
  xfit = NULL,
  yfit = NULL,
  prange = NULL,
  coef_order = NULL,
  digits = 2L,
  ...
)
```

Arguments

x	A 'spar' object.
plot_type	The type of plot. Options: • "val_measure" (default): Validation measure vs. nus or nummods. • "val_numactive": Number of active variables vs. nus or nummods. • "res_vs_fitted": Residuals vs. fitted values. • "coefs": Heatmap of coefficients.
plot_along	The variable to plot along the x-axis. Options: • "nu" (default): Threshold values.

	"nummod": Number of marginal models. Ignored if plot_type = "res_vs_fitted" or plot_type = "coefs".
nummod	The number of models to fix when plot_along = "nu". If NULL, the optimal value is used. The value will be used in <code>predict.spar</code> when plot_type="res_vs_fitted".
nu	The threshold value to fix when plot_along = "nummod". If NULL, the optimal value is used. The value will be used in <code>predict.spar</code> when plot_type="res_vs_fitted".
xfit	The predictor data for fitted values (required if plot_type = "res_vs_fitted").
yfit	The response data for fitted values (required if plot_type = "res_vs_fitted").
prange	A vector of length 2 specifying the range of predictors to plot (for plot_type = "coefs"). Default: c(1, p).
coef_order	An optional vector specifying the order of coefficients for plot_type = "coefs". Default: 1:p (original order).
digits	The number of significant digits for axis labels. Default: 2L.
...	Additional arguments passed to or from other methods.

Value

Returns a 'ggplot2::ggplot' object.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
plot(spar_res)
plot(spar_res, plot_type = "val_measure", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_measure", plot_along = "nu", nummod = 10)
plot(spar_res, plot_type = "val_numactive", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_numactive", plot_along = "nu", nummod = 10)
plot(spar_res, plot_type = "res_vs_fitted", xfit = example_data$xtest,
  yfit = example_data$ytest)
plot(spar_res, plot_type = "coefs", prange = c(1,400))
```

plot.spar.cv

Plot Method for 'spar.cv' Object

Description

Plot cross-validation measure or number of active variables over different thresholds or number of models of 'spar.cv' object, produce a residuals vs fitted plot, or a plot of the estimated coefficients in each marginal model, sorted by their absolute value.

Usage

```
## S3 method for class 'spar.cv'
plot(
  x,
  plot_type = c("val_measure", "val_numactive", "res_vs_fitted", "coefs"),
  plot_along = c("nu", "nummod"),
  nummod = NULL,
  nu = NULL,
  xfit = NULL,
  yfit = NULL,
  opt_par = c("best", "1se"),
  prange = NULL,
  coef_order = NULL,
  digits = 2L,
  ...
)
```

Arguments

<code>x</code>	result of spar.cv function of class 'spar.cv'.
<code>plot_type</code>	one of <code>c("val_measure", "val_numactive", "res_vs_fitted", "coefs")</code> .
<code>plot_along</code>	one of <code>c("nu", "nummod")</code> .
<code>nummod</code>	fixed value for <code>nummod</code> when <code>plot_along="nu"</code> for <code>plot_type="val_measure"</code> or <code>"val_numactive"</code> ;
<code>nu</code>	fixed value for ν when <code>plot_along="nummod"</code> for <code>plot_type="val_measure"</code> or <code>"val_numactive"</code> ; same as for predict.spar.cv when <code>plot_type="res_vs_fitted"</code> .
<code>xfit</code>	optional vector of fitted values to be used for <code>plot_type="res_vs_fitted"</code> ; if not provided, the fitted values are computed using the best parameters. Argument is used only if <code>precompute_mode = "precompute_all"</code> and <code>plot_type = "res_vs_fitted"</code> .
<code>yfit</code>	optional vector of response values to be used for <code>plot_type="res_vs_fitted"</code> ; if not provided, the response values are computed using the best parameters. Argument is used only if <code>precompute_mode = "precompute_all"</code> and <code>plot_type = "res_vs_fitted"</code> .
<code>opt_par</code>	one of <code>c("1se", "best")</code> , chooses whether to select the best pair of <code>nus</code> and <code>nummods</code> according to cross-validated (CV) measure, or the sparsest solution within one sd of that optimal CV measure. Argument is used only if <code>precompute_mode = "precompute_all"</code> and <code>plot_type = "res_vs_fitted"</code> .
<code>prange</code>	optional vector of length 2 indicating the range of predictors to be plotted for <code>plot_type = "coefs"</code> ; defaults to <code>c(1, p)</code> where <code>p</code> is the number of predictors. Argument can be used only if <code>precompute_mode = "precompute_all"</code> .
<code>coef_order</code>	optional vector of length <code>p</code> indicating the order of predictors to be plotted for <code>plot_type = "coefs"</code> ; defaults to <code>seq_len(p)</code> where <code>p</code> is the number of predictors. Argument can be used only if <code>precompute_mode = "precompute_all"</code> .
<code>digits</code>	number of significant digits to be displayed in the axis; defaults to 2L.
<code>...</code>	further arguments passed to or from other methods

Value

'ggplot2::ggplot' object

Examples

```
example_data <- simulate_spareg_data(n = 80, p = 200, ntest = 100)
spar_res <- spar.cv(example_data$x, example_data$y, nfolds = 3L,
  screencoef = screen_cor(), rp = rp_gaussian(), nummods = c(5, 10))
plot(spar_res)
plot(spar_res, plot_type = "val_measure", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_measure", plot_along = "nu", nummod = 10)
plot(spar_res, plot_type = "val_numactive", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_numactive", plot_along = "nu", nummod = 10)
```

predict.spar

Predict Method for 'spar.cv' Object

Description

Predict responses for new predictors from 'spar' object

Usage

```
## S3 method for class 'spar'
predict(
  object,
  xnew = NULL,
  type = c("response", "link"),
  avg_type = c("link", "response"),
  nummod = NULL,
  nu = NULL,
  aggregate = c("mean", "median"),
  ...
)
```

Arguments

object	result of spar function of class 'spar'.
xnew	matrix of new predictor variables; must have same number of columns as x.
type	the type of required predictions; either on response level (default) or on link level
avg_type	type of averaging the marginal models; either on link (default) or on response level
nummod	number of models used to form coefficients; value with minimal validation measure is used if not provided.

nu	threshold level used to form coefficients; value with minimal validation measure is used if not provided.
aggregate	character one of c("mean", "median"); the aggregation over the ensembles is done using the specified method (mean or median). Defaults to mean aggregation.
...	further arguments passed to or from other methods

Value

Returns a vector of predictions.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
pred <- predict(spar_res, xnew = example_data$xtest)
```

predict.spar.cv

Predict Method for 'spar.cv' Object

Description

Predict responses for new predictors from 'spar.cv' object. Only allowed if precompute_mode = "precompute_all" is used when calling spar.cv. In this case the ensemble obtained on the whole data is used for the combination of threshold and number of models using the best or the 1se identified through cross-validation. Otherwise predictions cannot be generated without refitting on whole data with best or 1se parameters. To be able to generate predictions in case precompute_mode = "precompute_rpm" or precompute_mode = "full_cv", use spar() to refit with the desired (nu, M) combination.

Usage

```
## S3 method for class 'spar.cv'
predict(
  object,
  xnew = NULL,
  type = c("response", "link"),
  avg_type = c("link", "response"),
  opt_par = c("best", "1se"),
  nummod = NULL,
  nu = NULL,
  aggregate = c("mean", "median"),
  ...
)
```

Arguments

object	result of spar function of class 'spar.cv'.
xnew	matrix of new predictor variables; must have same number of columns as x.
type	the type of required predictions; either on response level (default) or on link level
avg_type	type of averaging the marginal models; either on link (default) or on response level
opt_par	one of c("best", "1se"), chooses whether to select the best pair of nus and nummods according to CV measure, or the sparsest solution within one sd of that optimal CV measure; ignored when nummod and nu, or coef are given
nummod	number of models used to form coefficients; value with minimal validation Meas is used if not provided.
nu	threshold level used to form coefficients; value with minimal validation Meas is used if not provided.
aggregate	character one of c("mean", "median"); the aggregation over the ensembles is done using the specified method (mean or median). Defaults to mean aggregation.
...	further arguments passed to or from other methods

Value

Vector of predictions

See Also

[coef.spar.cv](#), [get_model.spar.cv](#)

Examples

```
example_data <- simulate_spareg_data(n = 80, p = 200, ntest = 100)
spar_res <- spar.cv(example_data$x, example_data$y, nfolds = 3L,
  rp = rp_gaussian(), nummods = c(5, 10))
pred <- predict(spar_res, example_data$x)
```

print.coefspar

Print Method for 'coefspar' Object

Description

Prints a summary of coefficients from a 'coefspar' object, including the selected M and ν , and the number of active variables.

Usage

```
## S3 method for class 'coefspar'
print(x, digits = 4L, show = 6L, ...)
```

Arguments

x	A 'coefspar' object.
digits	The number of significant digits for numeric output. Default: 4L.
show	The number of coefficients to display. Default: 6L.
...	Additional arguments passed to or from other methods.

Value

Invisibly returns the input object x.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 2000, ntest = 100)
spar_res <- spareg(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
coef(spar_res)
coef(spar_res, aggregate = "median")
coef(spar_res, aggregate = "none")
print(coef(spar_res), show = 10L, digits = 6L)
```

```
print.randomprojection
```

Print Method for a 'randomprojection' Object

Description

Prints a summary of a [randomprojection](#) object, including its name, and key attributes such as mslow and msup.

Usage

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

Arguments

x	An object of class randomprojection .
...	Additional arguments (ignored).

Value

Invisibly returns the input object x.

Examples

```
rp <- rp_gaussian(mslow = 5, msup = 10)
print(rp)
```

print.screencoef	<i>Print Method for 'screencoef' Object</i>
------------------	---

Description

Print method for a 'screencoef' object

Usage

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

Arguments

x	description
...	further arguments passed to or from other methods

Value

text summary

print.spar	<i>Print summary of 'spar' Object</i>
------------	---------------------------------------

Description

Prints a summary of a [spar](#) object, including the validation measure, optimal M and ν , and the number of active predictors.

Usage

```
## S3 method for class 'spar'
print(x, digits = 4L, ...)
```

Arguments

x	A 'spar' object.
digits	The number of significant digits for numeric output. Default: 4L.
...	Additional arguments passed to or from other methods.

Value

Invisibly returns the input object x.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
print(spar_res)
```

print.spar.cv

Print Method for 'spar.cv' Object

Description

Print summary of 'spar.cv' object

Usage

```
## S3 method for class 'spar.cv'
print(x, digits = 4L, ...)
```

Arguments

x	result of spar.cv function of class 'spar.cv'.
digits	integer digits to be printed, defaults to 4L.
...	further arguments passed to or from other methods

Value

text summary

Examples

```
example_data <- simulate_spareg_data(n = 80, p = 200, ntest = 100)
spar_res <- spareg.cv(example_data$x, example_data$y, nfolds = 3L,
  screencoef = screen_cor(), rp = rp_gaussian(), nummods = c(5, 10))
print(spar_res)
```

randomprojection-class

Random Projection Object Class

Description

The 'randomprojection' class represents a configuration for generating and managing random projection matrices in the **SPAR (Sparse Projected Averaged Regression)** framework. Objects of this class encapsulate:

- Functions for generating, updating, and modifying random projection matrices.
- Control parameters for the random projection process.
- Metadata (e.g., name, attributes) for customization.

Details

Objects of class 'randomprojection' are created using the constructor_randomprojection function. They are used in the spar and spar.cv functions to define how predictors are projected into a lower-dimensional space.

The class includes the following components:

- name: A character string describing the random projection method (e.g., "rp_gaussian", "rp_sparse").
- generate_fun: A function to generate the random projection matrix. This function must accept arguments like object, x, y, m, included_vector, and ...
- update_fun: A function to update the randomprojection object with data-specific information. This function must accept arguments like object, x, y, family, and ...
- update_rpm_w_data: A function to update an already-generated random projection matrix with data-dependent information. This function must accept arguments like rpm, object, included_vector, x, y, family, and ...
- control: A list of control parameters for the random projection process (e.g., mslow, msup, psi).

Attributes

The following attributes are commonly used in randomprojection objects:

- mslow: Integer. The minimum dimension for projection. Default: $\lfloor \log(p) \rfloor$.
- msup: Integer. The maximum dimension for projection. Default: $\lfloor n/2 \rfloor$.
- data: Logical. If TRUE, the projection matrix is updated with data-dependent coefficients (e.g., for rp_cw).

Usage

'randomprojection' objects are typically created using predefined constructors like:

- `rp_gaussian()`: Gaussian random projection.
- `rp_sparse()`: Sparse random projection (Achlioptas, 2003).
- `rp_cw()`: Sparse Embedding (Clarkson-Woodruff) random projection.

Users can also define custom random projection methods by providing their own `generate_fun`, `update_fun`, and `update_rpm_w_data` functions to `constructor_randomprojection`.

References

Achlioptas D (2003). "Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins." *Journal of Computer and System Sciences*, **66**(4), 671-687. ISSN 0022-0000. doi:[10.1016/S00220000\(03\)000254](https://doi.org/10.1016/S00220000(03)000254). Special Issue on PODS 2001.

Clarkson KL, Woodruff DP (2013). "Low Rank Approximation and Regression in Input Sparsity Time." In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, 81–90. ISBN 9781450320290. doi:[10.1145/2488608.2488620](https://doi.org/10.1145/2488608.2488620).

Parzer R, Filzmoser P, Vana-Gür L (2025). "Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models." *Statistical Modelling*. doi:[10.1177/1471082X251392705](https://doi.org/10.1177/1471082X251392705).

See Also

[constructor_randomprojection](#), [rp_gaussian](#), [rp_sparse](#), [rp_cw](#), [spar](#)

Examples

```
rp_gauss <- rp_gaussian(mslow = 5, msup = 10)
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(
  example_data$x,
  example_data$y,
  xval = example_data$xtest,
  yval = example_data$ytest,
  rp = rp_gauss
)
```

rp_cw

Sparse Embedding Matrix

Description

Creates an object class 'randomprojection' using arguments passed by user which in turn can be employed to generate a sparse embedding matrix as in (Clarkson and Woodruff 2013).

Usage

```
rp_cw(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the random projection matrix
control	list of arguments to be used in functions generate_fun, update_fun, update_rpm_w_data

Details

The entries of the matrix are generated based on (Clarkson and Woodruff 2013). This matrix is constructed as $\Phi = BD \in \mathbb{R}^{m \times p}$, where B is a $(p \times p)$ binary matrix, where for each column j an index is uniformly sampled from $\{1, \dots, m\}$ and the corresponding entry is set to one, and D is a $(p \times p)$ diagonal matrix, with entries $d_j \sim \text{Unif}(\{-1, 1\})$. If specified as `rp_cw(data = TRUE)`, the random elements on the diagonal are replaced by the ridge coefficients with a small penalty, as introduced in (Parzer et al. 2025).

Arguments related to the random projection procedure can be passed to the `rp_cw()` function through ..., and will be saved as attributes of the 'randomprojection' object. The following attributes are relevant for [spar](#) and [spar.cv](#):

- `mslow`: integer giving the minimum dimension to which the predictors should be projected; defaults to $\log(p)$.
- `msup`: integer giving the maximum dimension to which the predictors should be projected; defaults to $n/2$.

Value

object of class 'randomprojection' which is a list with elements `name`, `generate_fun`, `update_fun`, `control`

References

Clarkson KL, Woodruff DP (2013). “Low Rank Approximation and Regression in Input Sparsity Time.” In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, 81–90. ISBN 9781450320290. [doi:10.1145/2488608.2488620](#).

Parzer R, Filzmoser P, Vana-Gür L (2025). “Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models.” *Statistical Modelling*. [doi:10.1177/1471082X251392705](#)..

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  rp = rp_cw(data = TRUE))
```

rp_gaussian

*Gaussian Random Projection Matrix***Description**

Creates an object class 'randomprojection' using arguments passed by user which in turn can be employed to generate a random matrix with normally distributed entries (mean 0 and standard deviation 1 by default).

Usage

```
rp_gaussian(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the random projection matrix
control	list of arguments to be used in functions generate_fun, update_fun, update_rpm_w_data

Details

Arguments related to the random projection procedure can be passed to the rp_gaussian() function through ..., and will be saved as attributes of the 'randomprojection' object. The following attributes are relevant for [spar](#) and [spar.cv](#):

- mslow: integer giving the minimum dimension to which the predictors should be projected; defaults to $\log(p)$.
- msup: integer giving the maximum dimension to which the predictors should be projected; defaults to $n/2$.

Value

Returns an object of class 'randomprojection' which is a list with elements name, generate_fun, update_fun, control

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  rp = rp_gaussian(control = list(sd = 1/sqrt(ncol(example_data$x)))))
```

rp_sparse

*Sparse Random Projection Matrix***Description**

Creates an object class 'randomprojection' using arguments passed by user which in turn can be employed to generate a sparse embedding matrix as in (Achlioptas 2003).

Usage

```
rp_sparse(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the random projection matrix.
control	list of arguments to be used in functions generate_fun, update_fun, update_rpm_w_data

Details

The sparse matrix used in (Achlioptas 2003) with entries equal to $\Psi_{ij} = \pm 1/\sqrt{\psi}$ with probability $\psi/2$ and zero otherwise for $\psi \in (0, 1]$. Default is $\psi = 1$.

Arguments related to the random projection procedure can be passed to the `rp_gaussian()` function through ..., and will be saved as attributes of the 'randomprojection' object. The following attributes are relevant for `spar` and `spar.cv`:

- `mslow`: integer giving the minimum dimension to which the predictors should be projected; defaults to $\log(p)$.
- `msup`: integer giving the maximum dimension to which the predictors should be projected; defaults to $n/2$.

Value

object of class 'randomprojection' which is a list with elements `name`, `generate_fun`, `update_fun`, `control`

References

Achlioptas D (2003). "Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins." *Journal of Computer and System Sciences*, **66**(4), 671-687. ISSN 0022-0000. doi:[10.1016/S00220000\(03\)000254](https://doi.org/10.1016/S00220000(03)000254). Special Issue on PODS 2001.

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  rp = rp_sparse(control = list(psi = 1/3)))
```

screencoef-class	<i>Screening Coefficient Object Class</i>
------------------	---

Description

The 'screencoef' class represents a configuration for computing and managing **screening coefficients** in the **SPAR (Sparse Projected Averaged Regression)** framework. Objects of this class encapsulate:

- Functions for generating and updating screening coefficients.
- Control parameters for the screening process.
- Metadata (e.g., name, attributes) for customization.

Screening coefficients are used to reduce the dimensionality of the predictor space by selecting the most relevant variables before applying random projections.

Details

Objects of class `screencoef` are created using the `constructor_screencoef` function. They are used in the `spar` and `spar.cv` functions to define how predictors are screened.

The class includes the following components:

- `name`: A character string describing the screening method (e.g., "screen_marglik", "screen_cor", "screen_glmnet").
- `generate_fun`: A function to compute the screening coefficients. This function must accept:
 - `x`: A matrix of standardized predictors.
 - `y`: A vector of standardized responses.
 - `object`: An object of class `screencoef`.
 - `...`: Additional arguments passed from `spar()`.
- `update_fun`: A function to update the `screencoef` object with data-specific information. This function must accept:
 - `object`: An object of class `screencoef`.
 - `x`: A matrix of standardized predictors.
 - `y`: A vector of standardized responses.
 - `family`: A `stats::family` object.
 - `...`: Additional arguments passed from `spar()`. If not provided, the default `update_screen_default` is used, which leaves the object unchanged.
- `control`: A list of control parameters for the screening process (e.g., `nscreen`, `split_data_prop`).

Attributes

The following attributes are commonly used in screencoef objects:

- `type`: Character. The type of screening to employ:
 - `"prob"`: Probabilistic screening (variables are selected probabilistically based on their screening coefficients).
 - `"fixed"`: Fixed screening (the top `nscreen` variables are selected). Default: `"prob"`.
- `nscreen`: Integer. The number of variables to retain after screening. Default: `2n` (twice the number of observations).
- `split_data_prop`: Numeric. The proportion of data to use for computing screening coefficients. The remaining data is used for fitting the marginal models. Default: `1` (use all data).
- `reuse_in_rp`: Logical. If `TRUE`, the screening coefficients are reused in the construction of the random projection. Default: `FALSE`.
- `importance`: Numeric vector. The screening coefficients computed from the data.
- `inc_prob`: Numeric vector. The inclusion probabilities for probabilistic screening (normalized screening coefficients).

Screening Methods

The following predefined screening methods are available:

- `screen_marglik`: Screening based on marginal likelihood in univariate GLMs. Uses `stats::glm` to fit a separate GLM for each predictor and extracts the coefficients.
- `screen_cor`: Screening based on correlation between predictors and the response. Uses `stats::cor` to compute correlation coefficients.
- `screen_glmnet`: Screening based on penalized regression coefficients from `glmnet::glmnet`. Uses Lasso or Ridge regression to compute coefficients.

Users can also define custom screening methods by providing their own `generate_fun` and `update_fun` functions to `constructor_screencoef`.

See Also

`constructor_screencoef`, `screen_marglik`, `screen_cor`, `screen_glmnet`, `spar`

Examples

```
screen_marglik_obj <- screen_marglik(nscreen = 500, type = "prob")
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(
  example_data$x,
  example_data$y,
  xval = example_data$xtest,
  yval = example_data$ytest,
  screencoef = screen_marglik_obj
)
```

screen_cor

*Screening Coefficient Based on Correlation***Description**

Creates an object class 'screencoef' using arguments passed by user, where the screening coefficient should be computed based on the correlation coefficient of response and each predictor separately.

Usage

```
screen_cor(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the 'screencoef' object
control	list of controls to be passed to the screening function

Details

Creates an object class 'screencoef' using arguments passed by user.

The function generate_fun relies on [cor](#).

Arguments related to the screening procedure can be passed to the screen_cor() function through ..., and will be saved as attributes of the 'screencoef' object.

Value

object of class 'screencoef' which is a list with elements

- name (character, optional, used for printing)
- control (list of controls passed as an argument)
- generate_fun for generating the screening coefficient. This function should have arguments and y (vector of (standardized for Gaussian) responses), x (the matrix of standardized predictors) and a 'screencoef' object.

See Also

[constructor_screencoef](#), [screencoef](#)

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  screencoef = screen_cor(control = list(method = "kendall")))
```

screen_glmnet

*Screening Coefficient Based on [glmnet](#) Coefficients***Description**

Creates an object class 'screencoef' using arguments passed by user, where the screening coefficient should be computed based on penalized coefficients.

Usage

```
screen_glmnet(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the 'screencoef' object
control	list of controls to be passed to the screening function

Details

Creates an object class 'screencoef' using arguments passed by user.

The function generate_fun relies on [glmnet](#).

Arguments related to the screening procedure can be passed to the screen_glmnet() function through ..., and will be saved as attributes of the 'screencoef' object.

Value

object of class 'screencoef' which is a list with elements

- name (character, optional, used for printing)
- control (list of controls passed as an argument)
- generate_fun for generating the screening coefficient. This function should have arguments and y (vector of (standardized for Gaussian) responses), x (the matrix of standardized predictors) and a 'screencoef' object.

See Also

[constructor_screencoef](#), [screencoef](#)

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  screencoef = screen_glmnet(control = list(alpha = 0.1)))
```

screen_marglik	<i>Screening Coefficient Based on Marginal GLMs</i>
----------------	---

Description

Creates an object class 'screencoef' using arguments passed by user, where the screening coefficient should be computed based on the marginal likelihood of the univariate GLM where the response is regressed on each predictor separately.

Usage

```
screen_marglik(..., control = list())
```

Arguments

...	includes arguments which can be passed as attributes to the 'screencoef' object
control	list of controls to be passed to the screening function

Details

The function `generate_fun` relies on [glm](#).

Arguments related to the screening procedure can be passed to the `screen_marglik()` function through ..., and will be saved as attributes of the 'screencoef' object. Note that if family is not provided in control, the family used in [spar](#) or [spar.cv](#) will be used.

Value

object of class 'screencoef' which is a list with elements:

- name (character, optional, used for printing)
- control (list of controls passed as an argument)
- generate_fun for generating the screening coefficient. This function should have arguments and y (vector of (standardized for Gaussian) responses), x (the matrix of standardized predictors) and a 'screencoef' object.

See Also

[constructor_screencoef](#), [screencoef](#)

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30),
  screencoef = screen_marglik(nscreen = 500))
```

simulate_spareg_data *Simulate Sparse Regression Data*

Description

Generates synthetic data for sparse linear regression problems. Returns training and test sets along with model parameters.

Usage

```
simulate_spareg_data(
  n,
  p,
  ntest,
  a = min(100, p/4),
  snr = 10,
  rho = 0.5,
  mu = 1,
  beta_vals = NULL,
  seed = NULL
)
```

Arguments

n	Integer. Number of training samples.
p	Integer. Number of predictors (features).
ntest	Integer. Number of test samples.
a	Integer. Number of non-zero coefficients in the true beta vector. Default is min(100, p/4).
snr	Numeric. Signal-to-noise ratio. Default is 10.
rho	Numeric between 0 and 1. Pairwise correlation coefficient among predictors. Default is 0.5. A compound symmetry correlation matrix is used. The variance of the predictors is fixed to 1.
mu	Numeric. Intercept term (mean of response). Default is 1.
beta_vals	Numeric. Possible values for non-zero coefficients in the true beta vector. Default to NULL, in which case the values -3, -2, -1, 1, 2, 3 will be used.
seed	Integer. Random seed for reproducibility. Default is NULL.

Value

A list with the following components:

- x** Training design matrix ($n \times p$).
- y** Training response vector (length n).

xtest Test design matrix (ntest x p).
ytest Test response vector (length ntest).
mu Intercept used in data generation.
beta True coefficient vector (length p).
sigma2 Noise variance used in data generation. Equals $\beta' \Sigma \beta / \text{snr}$.

Examples

```
set.seed(123)
data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
str(data)
```

spar

Sparse Projected Averaged Regression (SPAR)

Description

Fits a **Sparse Projected Averaged Regression (SPAR)** model to high-dimensional data. SPAR builds an ensemble of generalized linear models (GLMs) where high-dimensional predictors are first screened (using a screening coefficient) and then projected using random projection matrices. This function evaluates the model over a grid of thresholds (nus) and a grid of the number of marginal models (nummods). It is also used internally by the cross-validated procedure [spar.cv](#).

Usage

```
spar(
  x,
  y,
  family = gaussian("identity"),
  model = NULL,
  rp = NULL,
  screencoef = NULL,
  xval = NULL,
  yval = NULL,
  nnu = 20L,
  nus = NULL,
  nummods = 20L,
  measure = c("deviance", "mse", "mae", "class", "1-auc"),
  avg_type = c("link", "response"),
  parallel = FALSE,
  inds = NULL,
  RPMs = NULL,
  seed = NULL,
  ...
)
```

```

spareg(
  x,
  y,
  family = gaussian("identity"),
  model = NULL,
  rp = NULL,
  screencoef = NULL,
  xval = NULL,
  yval = NULL,
  nnu = 20L,
  nus = NULL,
  nummods = 20L,
  measure = c("deviance", "mse", "mae", "class", "1-auc"),
  avg_type = c("link", "response"),
  parallel = FALSE,
  inds = NULL,
  RPMs = NULL,
  seed = NULL,
  ...
)

```

Arguments

<code>x</code>	An $n \times p$ numeric matrix of predictor variables.
<code>y</code>	A quantitative response vector of length n .
<code>family</code>	A family object used for the marginal generalized linear model, default <code>gaussian("identity")</code> .
<code>model</code>	A function that creates a 'sparmodel' object. Defaults to: • <code>spar_glm()</code> for gaussian family with identity link. • <code>spar_glmnet()</code> for all other family-link combinations.
<code>rp</code>	A function that creates a 'randomprojection' object. Defaults to <code>NULL</code> . If <code>NULL</code> , <code>rp_cw(data = TRUE)</code> is used.
<code>screencoef</code>	function that creates a 'screencoef' object. Defaults to <code>NULL</code> . If <code>NULL</code> , no screening is used.
<code>xval</code>	An optional matrix of predictor variables used for validation. If <code>NULL</code> , <code>x</code> is used.
<code>yval</code>	An optional response vector for validation. If <code>NULL</code> , <code>y</code> is used.
<code>nnu</code>	number of different threshold values ν for thresholding. Ignored if <code>nus</code> is provided. Defaults to <code>20L</code> .
<code>nus</code>	An optional vector of thresholds ν . If <code>NULL</code> , <code>nnu</code> values between 0 and the maximum absolute marginal coefficient equally spaced on the probability scale are used (i.e., we use as grid for the thresholds zero and <code>nnu-1</code> quantiles of the absolute values of the estimated non-zero coefficients from the marginal models).
<code>nummods</code>	A vector of integers specifying the number of marginal models to consider for validation. Defaults to <code>20L</code> .
<code>measure</code>	The loss function for validation. Options:

	• "deviance" (default, available for all families). • "mse" or "mae" (mean squared/absolute error, for all families). • "class" (misclassification error, for binomial family only). • "1-auc" (one minus area under the ROC curve for binomial family only).
avg_type	The type of averaging for marginal models. Options: • "link" (default): Averaging on the link scale. • "response": Averaging on the response scale.
parallel	A logical indicating whether to use parallel estimation of the marginal models. Defaults to FALSE.
inds	An optional list of index vectors corresponding to variables retained after screening for each marginal model. Must have length $\max(\text{nummods})$.
RPMs	An optional list of projection matrices for each marginal model. Must have length $\max(\text{nummods})$.
seed	An optional integer seed for reproducibility. Default: NULL.
...	Additional arguments for backward compatibility.

Details

If a parallel backend (e.g., `doParallel`) is registered and `parallel = TRUE`, the `foreach` package is used to parallelize the estimation of marginal models. If a parallel backend is registered and `parallel = TRUE`, the `foreach` function is used to estimate the marginal models in parallel.

Value

An object of class `spar` with the following components:

- `betas`: A $p \times \max(\text{nummods})$ sparse matrix of standardized coefficients for each marginal model.
- `intercepts`: Intercepts for each marginal model.
- `scr_coef`: A vector of length p with screening coefficients for standardized predictors.
- `inds`: A list of index vectors for variables retained after screening.
- `RPMs`: A list of projection matrices for each marginal model.
- `val_res`: A `data.frame` with validation results (measure and number of active variables) for each M and ν .
- `val_set`: A logical flag indicating whether validation data were provided.
- `family`: The family object used for the GLM.
- `nus`: The vector of thresholds considered.
- `nummods`: The vector of numbers of marginal models considered.
- `ycenter`: The empirical mean of the initial response vector.
- `yscale`: The empirical standard deviation of the initial response vector.
- `xcenter`: A vector of empirical means for the initial predictors.
- `xscale`: A vector of empirical standard deviations for the initial predictors.

- `avg_type`: The averaging type used for validation.
- `measure`: The validation measure used.
- `rp`: The 'randomprojection' object.
- `screencoef`: The 'screencoef' object.
- `x_rows_for_fitting_marginal_models`: A vector of row indices from `x` used for fitting marginal models (if screening splits data).

References

- Parzer R, Filzmoser P, Vana-Gür L (2025). “Sparse Data-Driven Random Projection in Regression for High-Dimensional Data.” *Journal of Data Science, Statistics, and Visualisation*, **5**(5). doi:10.52933/jdssv.v5i5.138.
- Parzer R, Filzmoser P, Vana-Gür L (2025). “Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models.” *Statistical Modelling*. doi:10.1177/1471082X251392705.
- Clarkson KL, Woodruff DP (2013). “Low Rank Approximation and Regression in Input Sparsity Time.” In *Proceedings of the Forty-Fifth Annual ACM Symposium on Theory of Computing*, STOC '13, 81–90. ISBN 9781450320290. doi:10.1145/2488608.2488620.
- Achlioptas D (2003). “Database-Friendly Random Projections: Johnson-Lindenstrauss with Binary Coins.” *Journal of Computer and System Sciences*, **66**(4), 671–687. ISSN 0022-0000. doi:10.1016/S00220000(03)000254. Special Issue on PODS 2001.

See Also

[spar.cv](#), [coef.spar](#), [predict.spar](#), [plot.spar](#), [print.spar](#)

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30))
coefs <- coef(spar_res)
pred <- predict(spar_res, xnew = example_data$x)
plot(spar_res)
plot(spar_res, plot_type = "val_measure", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_measure", plot_along = "nu", nummod = 10)
plot(spar_res, plot_type = "val_numactive", plot_along = "nummod", nu = 0)
plot(spar_res, plot_type = "val_numactive", plot_along = "nu", nummod = 10)
plot(spar_res, plot_type = "res_vs_fitted", xfit = example_data$xtest,
  yfit = example_data$ytest)
plot(spar_res, plot_type = "coefs", prange = c(1,400))

spar_res <- spareg(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10, 15, 20, 25, 30))
```

spar.cv

*Sparse Projected Averaged Regression with Cross-Validation***Description**

Apply Sparse Projected Averaged Regression to High-Dimensional Data, where the number of models and the threshold parameter is chosen using a cross-validation procedure.

Usage

```
spar.cv(
  x,
  y,
  family = gaussian("identity"),
  model = spar_glmnet(),
  rp = NULL,
  screencoef = NULL,
  nfolds = 10,
  nnu = 20,
  nus = NULL,
  nummods = c(20),
  measure = c("deviance", "mse", "mae", "class", "1-auc"),
  avg_type = c("link", "response"),
  parallel = FALSE,
  seed = NULL,
  precompute_mode = c("precompute_all", "precompute_rpm", "full_cv"),
  ...
)

spareg.cv(
  x,
  y,
  family = gaussian("identity"),
  model = spar_glmnet(),
  rp = NULL,
  screencoef = NULL,
  nfolds = 10,
  nnu = 20,
  nus = NULL,
  nummods = c(20),
  measure = c("deviance", "mse", "mae", "class", "1-auc"),
  avg_type = c("link", "response"),
  parallel = FALSE,
  seed = NULL,
  precompute_mode = c("precompute_all", "precompute_rpm", "full_cv"),
  ...
)
```

Arguments

<code>x</code>	<code>n x p</code> numeric matrix of predictor variables.
<code>y</code>	quantitative response vector of length <code>n</code> .
<code>family</code>	a <code>'family'</code> object used for the marginal generalized linear model; defaults to <code>gaussian("identity")</code> .
<code>model</code>	function creating a <code>'sparmodel'</code> object; defaults to <code>spar_glm()</code> for gaussian family with identity link and to <code>spar_glmnet()</code> for all other family-link combinations.
<code>rp</code>	function creating a <code>'randomprojection'</code> object. If not specified by user it defaults to <code>rp_cw(data = TRUE)</code> .
<code>screencoef</code>	function creating a <code>'screeningcoef'</code> object. If not specified by user it defaults to no screening.
<code>nfolds</code>	number of folds to use for cross-validation; should be at least 2, defaults to 10.
<code>nnu</code>	number of different threshold values ν to consider for thresholding; ignored when <code>nus</code> is provided; defaults to 20.
<code>nus</code>	optional vector of ν 's to consider for thresholding; if not provided, <code>nnu</code> values ranging from 0 to the maximum absolute marginal coefficient are used.
<code>nummods</code>	vector of numbers of marginal models to consider for validation; defaults to <code>c(20)</code> .
<code>measure</code>	loss to use for validation; defaults to "deviance" available for all families. Other options are "mse" or "mae" (between responses and predicted means, for all families), "class" (misclassification error) and "1-auc" (one minus area under the ROC curve) both just for binomial family.
<code>avg_type</code>	type of averaging the marginal models; either on link (default) or on response level. This is used in computing the validation measure.
<code>parallel</code>	assuming a parallel backend is loaded and available, a logical indicating whether the function should use it in parallelizing the estimation of the marginal models. Defaults to FALSE.
<code>seed</code>	integer seed to be set at the beginning of the SPAR algorithm. Default to NULL, in which case no seed is set.
<code>precompute_mode</code>	character, one of <code>c("precompute_all", "precompute_rpm", "full_cv")</code> , indicating whether to use the same random projection matrix and/or the same indices for screening across folds. Defaults to "precompute_all".
<code>...</code>	further arguments mainly to ensure back-compatibility

Value

object of class `'spar.cv'` with elements

- `val_res` a `data.frame` with CV results for each fold and for each element of `nus` and `nummods`
- `fitted_objects` list of fitted objects for the whole data (first element) and for each fold, each element is a list with elements

- `betas_std` $p \times \max(\text{nummods})$ sparse matrix of class `'Matrix::dgCMatrix'` containing the standardized coefficients from each marginal model computed with the SPAR algorithm on the training data.
- `intercepts` in each marginal model, vector of length $\max(\text{nummods})$ computed with the SPAR algorithm on the training data.
- `scr_coef` p -vector of coefficients used for screening for standardized predictors
- `inds` list of index-vectors corresponding to variables kept after screening in each marginal model of length $\max(\text{nummods})$. If kept fixed, only the first element corresponding to the indicators computed on the whole data is populated.
- `RPMs` list of projection matrices used in each marginal model of length $\max(\text{nummods})$. If kept fixed, only the first element corresponding to the projection matrices generated once on the whole data is populated.
- `ycenter` empirical mean of response vector in training data
- `yscale` empirical standard deviation of response vector in training data .
- `xcenter` p -vector of empirical means of predictor variables in training data
- `xscale` p -vector of empirical standard deviations of predictor variables in training data
- `seed` integer seed used at the beginning of the algorithm. In each fold the seed is modified as `seed + fold_id`.
- `nus` vector of ν 's considered for thresholding
- `nummods` vector of numbers of marginal models considered for validation
- `family` a character corresponding to `family` object used for the marginal generalized linear model e.g., `"gaussian(identity)"`
- `measure` character, type of validation measure used
- `avg_type` character, averaging type for computing the validation measure
- `rp` an object of class `'randomprojection'`
- `screencoef` an object of class `'screeningcoef'`
- `model` an object of class `'sparmodel'`
- `precompute_mode` character, one of `c("precompute_all", "precompute_rpm", "full_cv")`, indicating whether the same random projection matrix and/or the same indices for screening across folds were used.
- `seed` integer seed used at the beginning of the algorithm.

See Also

[spar](#), [coef.spar.cv](#), [predict.spar.cv](#), [plot.spar.cv](#), [print.spar.cv](#)

Examples

```
example_data <- simulate_spareg_data(n = 80, p = 200, ntest = 100)
spar_res <- spar.cv(example_data$x, example_data$y, nfolds = 3L,
  rp = rp_gaussian(), nummods = c(5, 10))
spar_res
```

```
spar_res <- spareg.cv(example_data$x, example_data$y,
  nummods=c(5, 10, 15, 20, 25, 30))
```

sparmodel-class	<i>SPAR Model Object Class</i>
-----------------	--------------------------------

Description

The `sparmodel` class represents a configuration for **fitting marginal models** in the **SPAR (Sparse Projected Averaged Regression)** framework. Objects of this class encapsulate:

- Functions for estimating marginal models (e.g., GLM, GLMNET, or custom models).
- Control parameters for the model fitting process.
- Metadata (e.g., name, attributes) for customization.

Marginal models are fitted to **projected predictors** (after screening and random projection) to compute coefficients and intercepts for each model in the ensemble.

Details

Objects of class `sparmodel` are created using the `constructor_sparmodel` function or predefined constructors like `spar_glm` and `spar_glmnet`. They are used in the `spar` and `spar.cv` functions to define how marginal models are fitted to the projected data.

The class includes the following components:

- `name`: A character string describing the model type (e.g., "glm", "glmnet", "glmrob").
- `generate_fun`: A function to estimate the marginal model coefficients and intercept. This function must accept:
 - `object`: An object of class `sparmodel`.
 - `x`: A matrix of standardized predictors (unused in most cases, as models are fitted on projected data).
 - `y`: A vector of standardized responses.
 - `z`: A matrix of projected predictors (the design matrix for the marginal model).
 - `...`: Additional arguments passed from `spar()`. The function must return a list with two elements:
 - * `gammas`: A vector of regression coefficients for the projected predictors.
 - * `intercept`: The intercept of the model.
- `update_fun`: A function to update the `sparmodel` object with data-specific information before fitting. This function must accept:
 - `object`: An object of class `sparmodel`.
 - `x`: A matrix of standardized predictors.
 - `y`: A vector of standardized responses.
 - `family`: A `stats::family` object.
 - `...`: Additional arguments passed from `spar()`. If not provided, the default `update_sparmodel_default` is used, which only updates the `family` if missing.
- `control`: A list of control parameters for the model fitting process (e.g., `family`, `alpha` for `glmnet`).

Model Types

The following predefined model types are available:

- `spar_glm`: Fits marginal models using `stats::glm`. Supports all standard GLM families. For Gaussian models with identity link, it uses a fast OLS solver.
- `spar_glmnet`: Fits marginal models using `glmnet::glmnet`. Supports penalized regression (Ridge, Lasso, or Elastic Net). Automatically converts family objects to strings for compatibility with `glmnet`.
- **Custom Models**: Users can define their own model types by providing custom `generate_fun` and `update_fun` functions to `constructor_sparmodel`. For example, `spar_glmrob` (not exported by default) uses `robustbase::glmrob` for robust regression.

See Also

`constructor_sparmodel`, `spar_glm`, `spar_glmnet`, `spar`

Examples

```
model_glmnet <- spar_glmnet(alpha = 0.5)

example_data <- simulate_spareg_data(n = 200, p = 2000, ntest = 100)
spar_res <- spar(
  example_data$x,
  example_data$y,
  xval = example_data$xtest,
  yval = example_data$ytest,
  model = model_glmnet
)
```

spar_glm

GLM Marginal `sparmodel`

Description

Creates an object class `sparmodel` using arguments passed by user. The generating function computes the coefficients of the marginal models based on a GLM. Computation relies on `stats::glm`.

Usage

```
spar_glm(..., control = list())
```

Arguments

<code>...</code>	includes arguments which can be passed as attributes to the <code>sparmodel</code> object
<code>control</code>	list of controls to be passed to the model function

Details

Relies on [glm](#).

Value

Returns an object of class [sparmodel](#).

See Also

[sparmodel](#), [stats::glm](#)

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y,
  xval = example_data$xtest, yval = example_data$ytest,
  model = spar_glm())
```

spar_glmnet

Penalized GLM Marginal [sparmodel](#)

Description

Creates an object class [sparmodel](#) using arguments passed by user, where the generating function computes the coefficients of the marginal models based on a penalized GLM. Computation relies on [glmnet::glmnet](#). By default, the models assume $\alpha = 0$ and return the coefficients obtained with the penalty $\lambda_{\min}$.

Usage

```
spar_glmnet(..., control = list())
```

Arguments

<code>...</code>	includes arguments which can be passed as attributes to the sparmodel object
<code>control</code>	list of controls to be passed to the model function

Details

Relies on [glmnet](#).

Value

Returns an object of class [sparmodel](#).

See Also

[sparmodel](#), [glmnet::glmnet](#)

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 400, ntest = 100)
spar_res <- spar(example_data$x, example_data$y,
  xval = example_data$xtest, yval = example_data$ytest,
  model = spar_glmnet(alpha = 0.1))
```

summary.coefspars	<i>Summary Method for 'coefspars' Object</i>
-------------------	--

Description

Provides a detailed summary of a coefspars object, including coefficient statistics.

Usage

```
## S3 method for class 'coefspars'
summary(object, digits = 4L, ...)
```

Arguments

object	An object of class coefspars.
digits	Number of digits to be printed. Defaults to 4L.
...	Additional arguments passed to or from other methods.

Value

Invisibly returns input object.

Examples

```
example_data <- simulate_spareg_data(n = 100, p = 2000, ntest = 100)
spar_res <- spareg(example_data$x, example_data$y, xval = example_data$xtest,
  yval = example_data$ytest, nummods=c(5, 10))
summary(coef(spar_res))
summary(coef(spar_res, aggregate = "none"))
```

Index

`coef.spar`, 2, 10, 37
`coef.spar.cv`, 4, 10, 19, 40
`constructor_randomprojection`, 5, 24
`constructor_screencoef`, 7, 29–32
`constructor_sparmodel`, 8, 42
`cor`, 30

`family`, 35, 39, 40
`foreach`, 36

`get_coef`, 9
`get_intercept`, 10
`get_measure`, 11
`get_model`, 11, 11
`get_model.spar`, 12
`get_model.spar.cv`, 5, 13, 19
`ggplot2::ggplot`, 15, 17
`glm`, 32, 43
`glmnet`, 31, 43
`glmnet::glmnet`, 29, 42, 43

`Matrix::dgCMatrix`, 40

`plot.spar`, 14, 37
`plot.spar.cv`, 15, 40
`predict.spar`, 15, 17, 37
`predict.spar.cv`, 5, 16, 18, 40
`print.coefspar`, 3, 10, 19
`print.randomprojection`, 20
`print.screencoef`, 21
`print.spar`, 21, 37
`print.spar.cv`, 22, 40

`randomprojection`, 5, 6, 20
`randomprojection-class`, 23
`robustbase::glmrob`, 42
`rp_cw`, 24, 24
`rp_gaussian`, 24, 26
`rp_sparse`, 24, 27

`screen_cor`, 29, 30

`screen_glmnet`, 29, 31
`screen_marglik`, 29, 32
`screencoef`, 7, 8, 30–32
`screencoef-class`, 28
`simulate_spareg_data`, 33
`spar`, 3, 11, 21, 24–27, 29, 32, 34, 36, 40, 42
`spar.cv`, 4, 11, 16, 22, 25–27, 32, 34, 37, 38
`spar_glm`, 42, 42
`spar_glmnet`, 42, 43
`spareg(spar)`, 34
`spareg.cv(spar.cv)`, 38
`sparmodel`, 8, 9, 42, 43
`sparmodel-class`, 41
`stats::cor`, 29
`stats::family`, 6, 28, 41
`stats::glm`, 29, 42, 43
`summary.coefspar`, 3, 10, 44