Package ‘spareg’

July 18, 2025

Type Package

Title Sparse Projected Averaged Regression

Version 1.1.0

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. For more details see
Parzer et al (2024a) <doi:10.48550/arXiv.2312.00130> and
Parzer et al (2024b) <doi:10.48550/arXiv.2410.00971>.

License GPL-3

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

RdMacros Rdpack

Encoding UTF-8

RoxygenNote 7.3.2

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)

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 2025-07-18 15:20:02 UTC

Contents

`coef.spar` *coef.spar*

Description

Extract 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 function 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 form coefficients; value with minimal validation `measure` is used if not provided.
`aggregate`	character one of c("mean", "median", "none"). 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

object of class `'coefspar'` which is a list with elements

- `intercept` intercept value
- `beta` vector of length p of averaged coefficients
- `nummod` number of models based on which the coefficient is computed
- `nu` threshold based on which the coefficient is computed

See Also

print.coefspar, summary.coefspar

`coef.spar.cv` *coef.spar.cv*

Description

Extract coefficients from `'spar.cv'` object

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'`.
`nummod`	optional number of models used to form coefficients
`nu`	optional threshold level used to form coefficients
`opt_par`	one of `c("1se","best")`, 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 c("mean", "median", "none"). 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` intercept value
- `beta` vector of length p of averaged coefficients
- `nummod` number of models based on which the coefficient is computed
- `nu` threshold based on which the coefficient is computed

`constructor_randomprojection` *Constructor function for building* `'randomprojection'` *objects*

Description

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

Usage

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

Arguments

`name`	character
`generate_fun`	function for generating the random projection matrix. This function should have with arguments `rp`, which is a `'randomprojection'` object, `m`, the target dimension and a vector of indexes `included_vector`, `x` matrix of predictors and `y` matrix of predictors. Vector `included_vector` shows the column index of the original variables in the `x` matrix to be projected using the random projection. This is needed due to the fact that screening is employed pre-projection.
`update_fun`	function for updating the `'randomprojection'` object with information from the data. This function should have arguments `rp`, which is a `'randomprojection'` object and `x` (the matrix of predictors) and `y` (the vector of responses).
`update_rpm_w_data`	function for updating the random projection matrix with data. This can be used for the case where a list of random projection matrices is provided by argument `RPMs`. In this case, the random structure is kept fixed, but the data-dependent part gets updated with the provided data. Defaults to NULL. If not provided, the values of the provided RPMs do not change.
`control`	list of controls for random projection. Can include minimum and maximum dimension for the projection defaults to `list(mslow = NULL, msup = NULL)`

Value

a function which in turn creates an object of class `'randomprojection'`

---

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, generate_fun)
```

Arguments

`name`	character
`generate_fun`	function for generating the screening coefficient. This function should have arguments and `y` (vector of responses – standardized for Gaussian family), `x` (the matrix of standardized predictors) and a `'screencoef'` object.

Details

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

Value

a function which in turn creates an object of class `'screencoef'`

`constructor_sparmodel` *Constructor function for building* `'sparmodel'` *objects*

Description

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

Usage

```
constructor_sparmodel(name, model_fun, update_fun = NULL)
```

Arguments

`name`	character
`model_fun`	function for estimating the marginal models which returns the function should have arguments and `y` (vector of responses – standardized for Gaussian family), `z` (the matrix of projected predictors) and a `'sparmodel' object`.
`update_fun`	optional function for updating the `'sparmodel'` object before the start of the algorithm.

Details

Creates an object of class `'sparmodel'` using arguments passed by user.

Value

a function which in turn creates an object of class `'sparmodel'`.

`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

`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).

`get_measure` *Extractor for (cross-)validation measure from 'spar' or 'spar.cv' object*

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, 15, 20, 25, 30))
get_measure(spar_res)
```

`get_model` *Extractor of model from a 'spar' or 'spar.cv' object*

Description

Extractor of model from a 'spar' or 'spar.cv' object

Usage

```
get_model(object, opt_par = c("best", "1se"))
```

Arguments

`object`	A fitted '`spar`' or '`spar.cv`' model
`opt_par`	One of "best", "1se"

Value

A '`spar`' or '`spar.cv`' object where the beta and intercept elements are the ones which correspond to the best or the 1se 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, 15, 20, 25, 30))
best_model <- get_model(spar_res, opt_par = "best")

spar_cv <- spar.cv(example_data$x, example_data$y,
  nummods = c(5, 10, 15, 20, 25, 30), nfolds = 4)
best_model_cv <- get_model(spar_cv, opt_par = "best")
onese_model_cv <- get_model(spar_cv, opt_par = "1se")
```

`plot.spar` *plot.spar*

Description

Plot values of validation measure or number of active variables over different thresholds or number of models for `'spar'` object, or residuals vs fitted

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`	result of spar function of class `'spar'`.
`plot_type`	one of `c("val_measure", "val_numactive", "res_vs_fitted", "coefs")`.
`plot_along`	one of `c("nu","nummod")`; ignored when `plot_type = "res_vs_fitted"`.
`nummod`	fixed value for number of models when `plot_along = "nu"` for `plot_type = "val_measure"` or `"val_numactive"`; same as for `predict.spar` when `plot_type="res_vs_fitted"`.
`nu`	fixed value for ν when `plot_along="nummod"` for `plot_type = "val_measure"` or `"val_numactive"`; same as for `predict.spar` when `plot_type="res_vs_fitted"`.
`xfit`	data used for predictions in `"res_vs_fitted"`.
`yfit`	data used for predictions in `"res_vs_fitted"`.
`prange`	optional vector of length 2 for `"coefs"`-plot to give the limits of the predictors' plot range; defaults to `c(1, p)`.
`coef_order`	optional index vector of length p for `plot_type = "coefs"` to give the order of the predictors; defaults to `1 : p`.
`digits`	number of significant digits to be displayed in the axis; defaults to 2L.
`...`	further arguments passed to or from other methods

Value

`'ggplot2::ggplot'` object

plot.spar.cv *plot.spar.cv*

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 = 2,
  ...
)
```

Arguments

`x`	result of spar.cv function of class `'spar.cv'`.
`plot_type`	one of `c("val_measure","val_numactive","res_vs_fitted","coefs")`.
`plot_along`	one of `c("nu","nummod")`; ignored when `plot_type="res_vs_fitted"`.
`nummod`	fixed value for nummod when `plot_along="nu"` for `plot_type="val_measure"` or `"val_numactive"`; same as for `predict.spar.cv` when plot_type="res_vs_fitted".
`nu`	fixed value for ν when `plot_along="nummod"` for `plot_type="val_measure"` or `"val_numactive"`; same as for `predict.spar.cv` when `plot_type="res_vs_fitted"`.
`xfit`	data used for predictions in `"res_vs_fitted"`.
`yfit`	data used for predictions in `"res_vs_fitted"`.
`opt_par`	one of `c("best","1se")`, only needed for `plot_type="res_vs_fitted"` to set type of predictions, see `predict.spar.cv`.
`prange`	optional vector of length 2 for `"coefs"`-plot to give the limits of the predictors' plot range; defaults to `c(1, p)`.
`coef_order`	optional index vector of length p for `"coefs"`-plot to give the order of the predictors; defaults to `1 : p`.
`digits`	number of significant digits to be displayed in the axis; defaults to 2L.
`...`	further arguments passed to or from other methods

Value

`'ggplot2::ggplot'` object

`predict.spar` *predict.spar*

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

Vector of predictions

`predict.spar.cv` *predict.spar.cv*

Description

Predict responses for new predictors from `'spar.cv'` object

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

`print.coefspar` *Print method for coefspar objects*

Description

Print method showing the basic components of a `'coefspar'` object.

Usage

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

Arguments

`x`	An object of class `'coefspar'`, typically created by a custom model function.
`digits`	integer digits to be printed, defaults to 4L.
`show`	integer number of coefficients to be shown, defaults to 6L.
`...`	Additional arguments passed to or from other methods (ignored here).

Value

Invisibly returns the input object x.

Examples

```
example_data <- simulate_spareg_data(n = 200, 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, 15, 20, 25, 30))
coef(spar_res)
coef(spar_res, aggregate = "median")
coef(spar_res, aggregate = "none")
print(coef(spar_res), show = 10L, digits = 6L)
```

---

print.randomprojection *print.randomprojection*

Description

Print method for a 'randomprojection' object

Usage

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

Arguments

x	object of class 'randomprojection'
...	further arguments passed to or from other methods

Value

text summary

---

print.screencoef *print.screencoef*

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` *print.spar*

Description

Print summary of `'spar'` object

Usage

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

Arguments

`x`	result of spar function of class `'spar'`.
`...`	further arguments passed to or from other methods

Value

text summary

`print.spar.cv` *print.spar.cv*

Description

Print summary of `'spar.cv'` object

Usage

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

Arguments

`x`	result of spar.cv function of class `'spar.cv'`.
`...`	further arguments passed to or from other methods

Value

text summary

`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 $d_j \sim$ 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. 2024).

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 (2024). “Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models.” Technical Report 2410.00971, arXiv.org E-Print Archive. doi:10.48550/arXiv.2410.00971..

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

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, 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)))
```

`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. The following attributes are relevant for spar and spar.cv:

- `nscreen` integer giving the number of variables to be retained after screening; if not specified, defaults to $2n$.
- `split_data_prop`, double between 0 and 1 which indicates the proportion of the data that should be used for computing the screening coefficient. The remaining data will be used for estimating the marginal models in the SPAR algorithm; if not specified, the whole data will be used for estimating the screening coefficient and the marginal models.
- `type` character - either `"prob"` (indicating that probabilistic screening should be employed) or `"fixed"` (indicating that a fixed set of `nscreen` variables should be employed across the ensemble); defaults to `type = "prob"`.
- `reuse_in_rp` logical - indicates whether the screening coefficient should be reused at a later stage in the construction of the random projection. Defaults to `FALSE`.

Value

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

- `name` (character)
- `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.

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. The following attributes are relevant for spar and spar.cv:

- nscreen integer giving the number of variables to be retained after screening; if not specified, defaults to $2n$.
- split_data_prop, double between 0 and 1 which indicates the proportion of the data that should be used for computing the screening coefficient. The remaining data will be used for estimating the marginal models in the SPAR algorithm; if not specified, the whole data will be used for estimating the screening coefficient and the marginal models.
- type character - either "prob" (indicating that probabilistic screening should be employed) or "fixed" (indicating that a fixed set of nscreen variables should be employed across the ensemble); defaults to type = "prob".
- reuse_in_rp logical - indicates whether the screening coefficient should be reused at a later stage in the construction of the random projection. Defaults to FALSE.

Value

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

- `name` (character)
- `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.

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` *Screening coefficient based on marginal GLMs*

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. The following attributes are relevant for spar and spar.cv:

- `nscreen` integer giving the number of variables to be retained after screening; if not specified, defaults to $2n$.

- split_data_prop, double between 0 and 1 which indicates the proportion of the data that should be used for computing the screening coefficient. The remaining data will be used for estimating the marginal models in the SPAR algorithm; if not specified, the whole data will be used for estimating the screening coefficient and the marginal models.
- type character - either "prob" (indicating that probabilistic screening should be employed) or "fixed" (indicating that a fixed set of nscreen variables should be employed across the ensemble); defaults to type = "prob".
- reuse_in_rp logical - indicates whether the screening coefficient should be reused at a later stage in the construction of the random projection. Defaults to FALSE.

Value

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

- name (character)
- 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.

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 x 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 / snr.

Examples

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

`spar` *Sparse Projected Averaged Regression*

Description

Apply Sparse Projected Averaged Regression to high-dimensional data by building an ensemble of generalized linear models, where the high-dimensional predictors can be screened using a screening coefficient and then projected using data-agnostic or data-informed random projection matrices. This function performs the procedure for a given grid of thresholds ν and a grid of the number of marginal models to be employed in the ensemble. This function is also used in the cross-validated procedure spar.cv.

Usage

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

Arguments

`x`	n x p numeric matrix of predictor variables.
`y`	quantitative response vector of length n.
`family`	a family object used for the marginal generalized linear model, default `gaussian("identity")`.

`model`	function creating a `'sparmodel'` object; defaults to `spar_glm()` for gaussian family with identity link and to `spar_glmnet()` for all other family-link combinations.
`rp`	function creating a `'randomprojection'` object. Defaults to NULL. In this case `rp_cw(data = TRUE)` is used.
`screencoef`	function creating a `'screeningcoef'` object. Defaults to NULL. In this case no screening is used is used.
`xval`	optional matrix of predictor variables observations used for validation of threshold nu and number of models; `x` is used if not provided.
`yval`	optional response observations used for validation of threshold nu and number of models; `y` is used if not provided.
`nnu`	number of different threshold values ν to consider for thresholding; ignored when nus are given; defaults to 20.
`nus`	optional vector of ν's to consider for thresholding; if not provided, `nnu` values ranging from 0 to the maximum absolute marginal coefficient are used.
`nummods`	vector of numbers of marginal models to consider for validation; defaults to `c(20)`.
`measure`	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.
`avg_type`	type of averaging the marginal models; either on link (default) or on response level. This is used in computing the validation measure.
`parallel`	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.
`inds`	optional list of index-vectors corresponding to variables kept after screening in each marginal model of length `max(nummods)`; dimensions need to fit those of RPMs.
`RPMs`	optional list of projection matrices used in each marginal model of length `max(nummods)`, diagonal elements will be overwritten with a coefficient only depending on the given `x` and `y`.
`seed`	integer seed to be set at the beginning of the SPAR algorithm. Default to NULL, in which case no seed is set.
`...`	further arguments mainly to ensure back-compatibility

Value

object of class `'spar'` with elements

- `betas` p x `max(nummods)` sparse matrix of class `'Matrix::dgCMatrix'` containing the standardized coefficients from each marginal model
- `intercepts` used in each marginal model
- `scr_coef` vector of length p with coefficients used for screening the standardized predictors

- `inds` list of index-vectors corresponding to variables kept after screening in each marginal model of length max(nummods)
- `RPMs` list of projection matrices used in each marginal model of length `max(nummods)`
- `val_res` `data.frame` with validation results (validation measure and number of active variables) for each element of `nus` and `nummods`
- `val_set` logical flag, whether validation data were provided; if `FALSE`, training data were used for validation
- `family` a character corresponding to family object used for the marginal generalized linear model e.g., `"gaussian(identity)"`
- `nus` vector of ν's considered for thresholding
- `nummods` vector of numbers of marginal models considered for validation
- `ycenter` empirical mean of initial response vector
- `yscale` empirical standard deviation of initial response vector
- `xcenter` p-vector of empirical means of initial predictor variables
- `xscale` p-vector of empirical standard deviations of initial predictor variables
- `avg_type` character, averaging type for computing the validation measure
- `measure` character, type of validation measure used
- `rp` an object of class `"randomprojection"`
- `screencoef` an object of class `"screeningcoef"`
- `x_rows_for_fitting_marginal_models` vector of row indicators from `x` which were used for fitting the marginal models, if screening was performed using `screencoef` with `split_data_prop` argument. Is `NULL` otherwise.

If a parallel backend is registered and `parallel = TRUE`, the foreach function is used to estimate the marginal models in parallel.

References

Parzer R, Filzmoser P, Vana-Gür L (2024). "Sparse Data-Driven Random Projection in Regression for High-Dimensional Data." Technical Report 2312.00130, arXiv.org E-Print Archive. doi:10.48550/arXiv.2312.00130.

Parzer R, Filzmoser P, Vana-Gür L (2024). "Data-Driven Random Projection and Screening for High-Dimensional Generalized Linear Models." Technical Report 2410.00971, arXiv.org E-Print Archive. doi:10.48550/arXiv.2410.00971.

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*

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,
  ...
)

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,
  ...
)
```

Arguments

`x`	n x p numeric matrix of predictor variables.
`y`	quantitative response vector of length n.
`family`	a '`family`' object used for the marginal generalized linear model; defaults to `gaussian("identity")`.
`model`	function creating a '`sparmodel`' object; defaults to `spar_glm()` for gaussian family with identity link and to `spar_glmnet()` for all other family-link combinations.
`rp`	function creating a '`randomprojection`' object.
`screencoef`	function creating a '`screeningcoef`' object
`nfolds`	number of folds to use for cross-validation; should be at least 2, defaults to 10.
`nnu`	number of different threshold values ν to consider for thresholding; ignored when `nus` is provided; defaults to 20.
`nus`	optional vector of ν's to consider for thresholding; if not provided, `nnu` values ranging from 0 to the maximum absolute marginal coefficient are used.
`nummods`	vector of numbers of marginal models to consider for validation; defaults to `c(20)`.
`measure`	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.
`avg_type`	type of averaging the marginal models; either on link (default) or on response level. This is used in computing the validation measure.
`parallel`	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.
`seed`	integer seed to be set at the beginning of the SPAR algorithm. Default to NULL, in which case no seed is set.
`...`	further arguments mainly to ensure back-compatibility

Value

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

- `betas` p x `max(nummods)` sparse matrix of class `'Matrix::dgCMatrix'` containing the standardized coefficients from each marginal model computed with the spar algorithm on the whole training data.
- `intercepts` used in each marginal model, vector of length `max(nummods)` computed with the spar algorithm on the whole 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(nummods)`
- `RPMs` list of projection matrices used in each marginal model of length `max(nummods)`
- `val_res` a `data.frame` with CV results for each fold and for each element of nus and nummods
- `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'`
- `ycenter` empirical mean of initial response vector
- `yscale` empirical standard deviation of initial response vector .
- `xcenter` p-vector of empirical means of initial predictor variables
- `xscale` p-vector of empirical standard deviations of initial predictor variables

See Also

spar, coef.spar.cv, predict.spar.cv, plot.spar.cv, print.spar.cv

Examples

```
example_data <- simulate_spareg_data(n = 200, p = 400, ntest = 100)
spar_res <- spar.cv(example_data$x, example_data$y, nfolds = 3L,
  nummods = c(5, 10, 15, 20, 25, 30))
spar_res
coefs <- coef(spar_res)
pred <- predict(spar_res, 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, opt_par = "1se")
plot(spar_res, "coefs", prange = c(1, 400))


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

spar_glm *GLM marginal models*

Description

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

Usage

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

Arguments

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

Details

Relies on glm.

Value

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

- name (character)
- control (list of controls passed as an argument)
- model_fun function for estimating the model coefficients and the intercept. This function should have arguments y, vector of standardized responses, z, a matrix of projected predictors in each marginal model, and object, which is a 'sparmodel' object. Returns a list with two elements: gammas which is the vector of regression coefficients for the projected predictors and intercept which is the intercept of the model

`spar_glmnet` *Penalized GLM marginal models*

Description

Creates an object class `'sparmodel'` using arguments passed by user.

Usage

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

Arguments

`...`	includes arguments which can be passed as attributes to the `'sparmodel'` object
`control`	list of controls to be passed to the model function

Details

Relies on glmnet.

Value

object of class `'sparmodel'` which is a list with elements

- `name` (character)
- `control` (list of controls passed as an argument)
- `model_fun` for generating the screening coefficient. This function should have arguments `y`, vector of standardized responses, `z`, a matrix of projected predictors in each marginal model, and `object`, which is a `'sparmodel'` object. Returns a list with two elements: `gammas` which is the vector of regression coefficients for the projected predictors and `intercept` which is the intercept of the model.
- `update_fun` optional function for updating the `'sparmodel'` object before the start of the algorithm.

`summary.coefspar` *Summary method for coefspar objects*

Description

Provides a summary of a `coefspar` object.

Usage

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

Arguments

`object`	An object of class `coefspar`.
`digits`	integer digits to be printed, defaults to 4L.
`...`	Additional arguments (ignored).

Value

Invisibly returns `object`.

Examples

```
example_data <- simulate_spareg_data(n = 200, 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, 15, 20, 25, 30))
summary(coef(spar_res))
summary(coef(spar_res, aggregate = "none"))
```

Index