

Package ‘sparsegl’

September 22, 2022

Type Package

Title Sparse Group Lasso

Version 0.5.0

Description Efficient implementation of sparse group lasso with optional bound constraints on the coefficients. It supports the use of a sparse design matrix as well as returning coefficient estimates in a sparse matrix. Furthermore, it correctly calculates the degrees of freedom to allow for information criteria rather than cross-validation with very large data. Finally, the interface to compiled code avoids unnecessary copies and allows for the use of long integers.

License GPL (>= 2)

URL <https://github.com/dajmcdon/sparsegl/>

BugReports <https://github.com/dajmcdon/sparsegl/issues/>

Depends R (>= 3.5)

Imports assertthat, dotCall64, ggplot2, magrittr, Matrix, methods, rlang, RSpectra, tidyr

Suggests glmnet, knitr, markdown, rmarkdown, testthat (>= 3.0.0)

VignetteBuilder knitr

Config/testthat/edition 3

Encoding UTF-8

LazyData true

NeedsCompilation yes

RoxygenNote 7.2.1

Author Daniel J. McDonald [aut, cre],
Xiaoxuan Liang [aut],
Aaron Cohen [aut]

Maintainer Daniel J. McDonald <daniel@stat.ubc.ca>

Repository CRAN

Date/Publication 2022-09-22 08:30:02 UTC

R topics documented:

coef.cv.sparsegl	2
coef.sparsegl	3
cv.sparsegl	4
estimate_risk	6
plot.cv.sparsegl	7
plot.sparsegl	8
predict.cv.sparsegl	9
predict.sparsegl	10
print.sparsegl	11
sparsegl	12
trust_experts	14
zero_norm	15

Index	18
--------------	-----------

coef.cv.sparsegl	<i>Get coefficients from a cv.sparsegl object.</i>
------------------	--

Description

This function gets coefficients from a cross-validated `sparsegl()` model, using the stored "sparsegl.fit" object, and the optimal value chosen for lambda.

Usage

```
## S3 method for class 'cv.sparsegl'
coef(object, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	Fitted <code>cv.sparsegl()</code> object.
s	Value(s) of the penalty parameter lambda at which coefficients are desired. Default is the single value <code>s = "lambda.1se"</code> stored on the CV object (corresponding to the largest value of lambda such that CV error estimate is within 1 standard error of the minimum). Alternatively <code>s = "lambda.min"</code> can be used (corresponding to the minimum of cross validation error estimate). If s is numeric, it is taken as the value(s) of lambda to be used.
...	Not used. Other arguments to <code>predict()</code> .

Value

The coefficients at the requested value(s) for lambda.

See Also

`cv.sparsegl()`, and `predict.cv.sparsegl()` methods.

Examples

```

n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
cv_fit <- cv.sparsegl(X, y, groups)
coef(cv_fit, s = c(0.02, 0.03))

```

coef.sparsegl	<i>Extract model coefficients from a sparsegl object.</i>
---------------	---

Description

Computes the coefficients at the requested value(s) for lambda from a `sparsegl()` object.

Usage

```

## S3 method for class 'sparsegl'
coef(object, s = NULL, ...)

```

Arguments

object	Fitted <code>sparsegl()</code> object.
s	Value(s) of the penalty parameter lambda at which coefficients are required. Default is the entire sequence.
...	Not used.

Details

s is the new vector at which predictions are requested. If s is not in the lambda sequence used for fitting the model, the coef function will use linear interpolation to make predictions. The new values are interpolated using a fraction of coefficients from both left and right lambda indices.

Value

The coefficients at the requested values for lambda.

See Also

`sparsegl()`, `predict.sparsegl()` and `print.sparsegl()` methods.

Examples

```

n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
coef(fit1, s = c(0.02, 0.03))

```

cv.sparsegl

Cross-validation for a sparsegl object.

Description

Does k-fold cross-validation for `sparsegl()`. This function is largely similar `glmnet::cv.glmnet()`.

Usage

```

cv.sparsegl(
  x,
  y,
  group = NULL,
  family = c("gaussian", "binomial"),
  lambda = NULL,
  pred.loss = c("L2", "L1", "binomial", "misclass"),
  nfolds = 10,
  foldid = NULL,
  ...
)

```

Arguments

- | | |
|--------|--|
| x | Double. A matrix of predictors, of dimension $n \times p$; each row is a vector of measurements and each column is a feature. Objects of class <code>Matrix::sparseMatrix</code> are supported. |
| y | Double/Integer/Factor. The response variable. Quantitative for <code>family="gaussian"</code> . For <code>family="binomial"</code> should be either a factor with two levels or a vector of integers taking 2 unique values. For a factor, the last level in alphabetical order is the target class. |
| group | Integer. A vector of consecutive integers describing the grouping of the coefficients (see example below). |
| family | Character. Specifies the loss function to use, valid options are: <ul style="list-style-type: none"> • "gaussian" - least squares loss (regression, the default), • "binomial" - logistic loss (classification) |

lambda	A user supplied lambda sequence. The default, NULL results in an automatic computation based on nlambda, the smallest value of lambda that would give the null model (all coefficient estimates equal to zero), and lambda.factor. Supplying a value of lambda overrides this behaviour. It is likely better to supply a decreasing sequence of lambda values than a single (small) value. If supplied, the user-defined lambda sequence is automatically sorted in decreasing order.
pred.loss	Loss to use for cross-validation error. Valid options are: <ul style="list-style-type: none"> • "L2" for regression, mean square error • "L1" for regression, mean absolute error • "binomial" for classification, binomial deviance loss • "misclass" for classification, misclassification error.
nfolds	Number of folds - default is 10. Although nfolds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is nfolds = 3.
foldid	An optional vector of values between 1 and nfolds identifying which fold each observation is in. If supplied, nfolds can be missing.
...	Other arguments that can be passed to sparsegl.

Details

The function runs `sparsegl()` $\text{nfolds} + 1$ times; the first to get the lambda sequence, and then the remainder to compute the fit with each of the folds omitted. The average error and standard deviation over the folds are computed.

Value

An object of class `cv.sparsegl()` is returned, which is a list with the ingredients of the cross-validation fit.

lambda	The values of lambda used in the fits.
cvm	The mean cross-validated error - a vector of length <code>length(lambda)</code> .
cvsd	Estimate of standard error of cvm.
cvupper	Upper curve = $\text{cvm} + \text{cvsd}$.
cvlower	Lower curve = $\text{cvm} - \text{cvsd}$.
name	A text string indicating type of measure (for plotting purposes).
sparsegl.fit	A fitted <code>sparsegl()</code> object for the full data.
lambda.min	The optimal value of lambda that gives minimum cross validation error cvm.
lambda.1se	The largest value of lambda such that error is within 1 standard error of the minimum.

See Also

`sparsegl()`, `plot.cv.sparsegl()`, `predict.cv.sparsegl()`, and `coef.cv.sparsegl()` methods.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
cv_fit <- cv.sparsegl(X, y, groups)
```

estimate_risk	<i>Calculate information criteria.</i>
---------------	--

Description

This function uses the degrees of freedom to calculate various information criteria. This function uses the "unknown variance" version of the likelihood. Only implemented for Gaussian regression. The constant is ignored (as in `stats::extractAIC()`).

Usage

```
estimate_risk(object, x, type = c("AIC", "BIC", "GCV"), approx_df = FALSE)
```

Arguments

object	fitted object from a call to <code>sparsegl()</code> .
x	Matrix. The matrix of predictors used to estimate the <code>sparsegl</code> object. May be missing if <code>approx_df = TRUE</code> .
type	one or more of AIC, BIC, or GCV.
approx_df	the df component of a <code>sparsegl</code> object is an approximation (albeit a fairly accurate one) to the actual degrees-of-freedom. However, the exact value requires inverting a portion of $X'X$. So this computation may take some time (the default computes the exact df).

Value

a `data.frame` with as many rows as `object$lambda`. It contains columns `lambda`, `df`, and the requested risk types.

References

Vaïter S, Deledalle C, Peyré G, Fadili J, Dossal C. (2012). *The Degrees of Freedom of the Group Lasso for a General Design*. <https://arxiv.org/pdf/1212.6478.pdf>.

See Also

`sparsegl()` method.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
estimate_risk(fit1, type = "AIC", approx_df = TRUE)
```

plot.cv.sparsegl *Plot cross-validation curves produced from a cv.sparsegl object.*

Description

Plots the cross-validation curve, and upper and lower standard deviation curves, as a function of the lambda values used. This function is modified based on `glmnet::plot.cv.glmnet()`.

Usage

```
## S3 method for class 'cv.sparsegl'
plot(x, log_axis = c("xy", "x", "y", "none"), sign.lambda = 1, ...)
```

Arguments

x	Fitted <code>cv.sparsegl()</code> object
log_axis	Apply log scaling to the requested axes.
sign.lambda	Either plot against $\log(\lambda)$ (default) or the reverse if $\text{sign}.\lambda < 0$.
...	Not used.

Details

A plot is produced, a `ggplot2::ggplot()` object. Additional user modifications can be added as desired.

See Also

[cv.sparsegl\(\)](#).

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
```

```
groups <- rep(1:(p / 5), each = 5)
cv_fit <- cv.sparsegl(X, y, groups)
plot(cv_fit)
```

plot.sparsegl *Plot solution paths from a sparsegl object.*

Description

Produces a coefficient profile plot of a fitted `sparsegl()` object. The result is a `ggplot2::ggplot()`. Additional user modifications can be added as desired.

Usage

```
## S3 method for class 'sparsegl'
plot(
  x,
  y_axis = c("coef", "group"),
  x_axis = c("lambda", "penalty"),
  add_legend = TRUE,
  ...
)
```

Arguments

<code>x</code>	Fitted <code>sparsegl()</code> object.
<code>y_axis</code>	Variable on the y-axis. Either the coefficients (default) or the group norm.
<code>x_axis</code>	Variable on the x-axis. Either the (log)-lambda sequence (default) or the value of the penalty. In the second case, the penalty is scaled by its maximum along the path.
<code>add_legend</code>	Show the legend. Often, with many groups/predictors, this can become overwhelming.
<code>...</code>	Not used.

See Also

[sparsegl\(\)](#).

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
plot(fit1, y_axis = "coef", x_axis = "penalty")
```

predict.cv.sparsegl *Make predictions from a cv.sparsegl object.*

Description

This function makes predictions from a cross-validated `cv.sparsegl()` object, using the stored "sparsegl.fit" object, and the value chosen for lambda.

Usage

```
## S3 method for class 'cv.sparsegl'
predict(object, newx, s = c("lambda.1se", "lambda.min"), ...)
```

Arguments

object	Fitted <code>cv.sparsegl()</code> object.
newx	Matrix of new values for x at which predictions are to be made. Must be a matrix. See documentation for <code>predict.sparsegl()</code> .
s	Value(s) of the penalty parameter lambda at which coefficients are desired. Default is the single value <code>s = "lambda.1se"</code> stored on the CV object (corresponding to the largest value of lambda such that CV error estimate is within 1 standard error of the minimum). Alternatively <code>s = "lambda.min"</code> can be used (corresponding to the minimum of cross validation error estimate). If s is numeric, it is taken as the value(s) of lambda to be used.
...	Not used. Other arguments to <code>predict()</code> .

Value

A matrix or vector of predicted values.

See Also

`cv.sparsegl()`, and `coef.cv.sparsegl()` methods.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
cv_fit <- cv.sparsegl(X, y, groups)
predict(cv_fit, newx = X[50:60, ], s = "lambda.min")
```

predict.sparsegl *Make predictions from a sparsegl object.*

Description

Similar to other predict methods, this function produces fitted values and class labels from a fitted `sparsegl` object.

Usage

```
## S3 method for class 'sparsegl'
predict(
  object,
  newx,
  s = NULL,
  type = c("link", "response", "coefficients", "nonzero", "class"),
  ...
)
```

Arguments

object	Fitted <code>sparsegl()</code> model object.
newx	Matrix of new values for x at which predictions are to be made. Must be a matrix.
s	Value(s) of the penalty parameter lambda at which predictions are required. Default is the entire sequence used to create the model.
type	Type of prediction required. Type "link" gives the linear predictors for "binomial"; for "gaussian" models it gives the fitted values. Type "response" gives the fitted probabilities for "binomial"; for "gaussian" type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for s. Note that for "binomial" models, results are returned only for the class corresponding to the second level of the factor response. Type "class" applies only to "binomial" models, and produces the class label corresponding to the maximum probability. Type "nonzero" returns a list of the indices of the nonzero coefficients for each value of s.
...	Not used.

Details

s is new vector at which predictions are requested. If s is not in the lambda sequence used for fitting the model, the function will use linear interpolation to make predictions. The new values are interpolated using a fraction of predicted values from both left and right lambda indices.

Value

The object returned depends on type.

See Also

[sparsegl\(\)](#), [coef.sparsegl\(\)](#) and [print.sparsegl\(\)](#) methods.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
predict(fit1, newx = X[10, ], s = fit1$lambda[3:5])
```

```
print.sparsegl          Print a sparsegl object.
```

Description

Prints some summary information about the fitted [sparsegl\(\)](#) object.

Usage

```
## S3 method for class 'sparsegl'
print(x, digits = min(3, getOption("digits") - 3), ...)
```

Arguments

x	Fitted sparsegl() object.
digits	Significant digits in printout.
...	not used

See Also

[sparsegl\(\)](#), [coef.sparsegl\(\)](#) and [predict.sparsegl\(\)](#) methods.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit1 <- sparsegl(X, y, group = groups)
print(fit1)
```

sparsegl

*Regularization paths for sparse group-lasso models***Description**

Fits regularization paths for sparse group-lasso penalized learning problems at a sequence of regularization parameters `lambda`.

Usage

```
sparsegl(
  x,
  y,
  group = NULL,
  family = c("gaussian", "binomial"),
  nlambda = 100,
  lambda.factor = ifelse(nobs < nvars, 0.01, 1e-04),
  lambda = NULL,
  pf_group = sqrt(bs),
  pf_sparse = rep(1, nvars),
  intercept = TRUE,
  asparse = 0.05,
  standardize = TRUE,
  lower_bnd = -Inf,
  upper_bnd = Inf,
  dfmax = as.integer(max(group)) + 1L,
  pmax = min(dfmax * 1.2, as.integer(max(group))),
  eps = 1e-08,
  maxit = 3e+08
)
```

Arguments

- | | |
|---------------------|--|
| <code>x</code> | Double. A matrix of predictors, of dimension $n \times p$; each row is a vector of measurements and each column is a feature. Objects of class <code>Matrix::sparseMatrix</code> are supported. |
| <code>y</code> | Double/Integer/Factor. The response variable. Quantitative for <code>family="gaussian"</code> . For <code>family="binomial"</code> should be either a factor with two levels or a vector of integers taking 2 unique values. For a factor, the last level in alphabetical order is the target class. |
| <code>group</code> | Integer. A vector of consecutive integers describing the grouping of the coefficients (see example below). |
| <code>family</code> | Character. Specifies the loss function to use, valid options are: <ul style="list-style-type: none"> • "gaussian" - least squares loss (regression, the default), • "binomial" - logistic loss (classification) |

nlambda	The number of lambda values - default is 100.
lambda.factor	The factor for getting the minimal lambda in the lambda sequence, where $\min(\text{lambda}) = \text{lambda.factor} * \max(\text{lambda})$. $\max(\text{lambda})$ is the smallest value of lambda for which all coefficients are zero. The default depends on the relationship between n (the number of rows in the matrix of predictors) and p (the number of predictors). If $n \geq p$, the default is 0.0001. If $n < p$, the default is 0.01. A very small value of lambda.factor will lead to a saturated fit. This argument has no effect if there is user-defined lambda sequence.
lambda	A user supplied lambda sequence. The default, NULL results in an automatic computation based on nlambda, the smallest value of lambda that would give the null model (all coefficient estimates equal to zero), and lambda.factor. Supplying a value of lambda overrides this behaviour. It is likely better to supply a decreasing sequence of lambda values than a single (small) value. If supplied, the user-defined lambda sequence is automatically sorted in decreasing order.
pf_group	Penalty factor on the groups, a vector of the same length as the total number of groups. Separate penalty weights can be applied to each group of β s to allow differential shrinkage. Can be 0 for some groups, which implies no shrinkage, and results in that group always being included in the model (depending on pf_sparse). Default value for each entry is the square-root of the corresponding size of each group.
pf_sparse	Penalty factor on l1-norm, a vector the same length as the total number of columns in x. Each value corresponds to one predictor Can be 0 for some predictors, which implies that predictor will be receive by the group l2-norm penalty. Each entry should be non-negative in this vector.
intercept	Whether to include intercept in the model. Default is TRUE.
asparse	The weight to put on the ℓ_1 -norm in sparse group lasso. Default is 0.05.
standardize	Logical flag for variable standardization (scaling) prior to fitting the model. Default is TRUE.
lower_bnd	Lower bound for coefficient values, a vector in length of 1 or of length the number of groups. Must be non-positive numbers only. Default value for each entry is -Inf.
upper_bnd	Upper for coefficient values, a vector in length of 1 or of length the number of groups. Must be non-negative numbers only. Default value for each entry is Inf.
dfmax	Limit the maximum number of groups in the model. Default is no limit.
pmax	Limit the maximum number of groups ever to be nonzero. For example once a group enters the model, no matter how many times it exits or re-enters model through the path, it will be counted only once.
eps	Convergence termination tolerance. Defaults value is 1e-8.
maxit	Maximum number of outer-loop iterations allowed at fixed lambda value. Default is 3e8. If models do not converge, consider increasing maxit.

Details

Note that the objective function for least squares is

$$RSS/(2n) + \lambda \text{penalty}$$

Users can also tweak the penalty by choosing a different penalty factor.

For computing speed reason, if models are not converging or running slowly, consider increasing `eps`, decreasing `nlambda`, or increasing `lambda.factor` before increasing `maxit`.

Value

An object with S3 class `sparsegl()`.

- `call` The call that produced this object.
- `b0` Intercept sequence of length `length(lambda)`.
- `beta` A $p \times \text{length}(\text{lambda})$ sparse matrix of coefficients.
- `df` The number of features with nonzero coefficients for each value of `lambda`.
- `dim` Dimension of coefficient matrix.
- `lambda` The actual sequence of `lambda` values used.
- `npasses` Total number of iterations summed over all `lambda` values.
- `jerr` Error flag, for warnings and errors, 0 if no error.
- `group` A vector of consecutive integers describing the grouping of the coefficients.
- `nobs` The number of observations used to estimate the model.

See Also

`plot.sparsegl()`, `coef.sparsegl()`, `predict.sparsegl()` and `print.sparsegl()` methods.

Examples

```
n <- 100
p <- 20
X <- matrix(rnorm(n * p), nrow = n)
eps <- rnorm(n)
beta_star <- c(rep(5, 5), c(5, -5, 2, 0, 0), rep(-5, 5), rep(0, (p - 15)))
y <- X %*% beta_star + eps
groups <- rep(1:(p / 5), each = 5)
fit <- sparsegl(X, y, group = groups)
```

trust_experts

Trust in scientific experts during the Covid-19 pandemic

Description

A dataset containing a measurement of "trust" in experts along with other metrics collected through the Delphi Group at Carnegie Mellon University U.S. COVID-19 Trends and Impact Survey, in partnership with Facebook. This particular dataset is created from one of the public [contingency tables](#), specifically, the breakdown by state, age, gender, and race/ethnicity published on 05 February 2022.

Usage

trust_experts

Format

A sparse `Matrix::sparseMatrix()` with 3775 rows, 96 columns, and 51738 non-zero entries

y Real-valued response. This is the average of `pct_trust_covid_info_*` where * is each of doctors, experts, cdc, and govt_health.

yyyy-mm-01 0-1-valued predictor. Start date of data collection period. There are 8 monthly periods

AK-WY 0-1-valued predictor. State abbreviation.

age_* 0-1-valued predictor. Self-reported age bucket.

gender_* 0-1-valued predictor. Self-reported gender.

race_* 0-1-valued predictor. Self-reported race.

cli_* Real-valued predictor. `pct_cli` expanded in a B-spline basis with 10 degrees of freedom.

cmnty_cli_* Real-valued predictor. `pct_hh_cmnty_cli` expanded in a B-spline basis with 10 degrees of freedom.

Source

The U.S. COVID-19 Trends and Impact Survey.

The paper describing the survey:

Joshua A. Salomon, Alex Reinhart, Alyssa Bilinski, Eu Jing Chua, Wichada La Motte-Kerr, Minttu M. Rönn, Marissa Reitsma, Katherine Ann Morris, Sarah LaRocca, Tamar Farag, Frauke Kreuter, Roni Rosenfeld, and Ryan J. Tibshirani (2021). "The US COVID-19 Trends and Impact Survey: Continuous real-time measurement of COVID-19 symptoms, risks, protective behaviors, testing, and vaccination", *Proceedings of the National Academy of Sciences* 118 (51) e2111454118. [doi:10.1073/pnas.2111454118](https://doi.org/10.1073/pnas.2111454118).

[The Public Delphi US CTIS Documentation](#)

zero_norm

Calculate common group norms

Description

Norm calculation

Usage

```
zero_norm(x)

one_norm(x)

two_norm(x)

grouped_zero_norm(x, gr)

grouped_one_norm(x, gr)

grouped_two_norm(x, gr)

grouped_sp_norm(x, gr, asparse)

gr_one_norm(x, gr)

gr_two_norm(x, gr)

sp_group_norm(x, gr, asparse = 0.05)
```

Arguments

x	A numeric vector.
gr	A numeric vector of the same length as x.
asparse	The weight to put on the l1 norm when calculating the group norm.

Value

A numeric scalar or vector

Functions

- `zero_norm()`: l0-norm (number of nonzero entries).
- `one_norm()`: l1-norm (Absolute-value norm).
- `two_norm()`: l2-norm (Euclidean norm).
- `grouped_zero_norm()`: A vector of group-wise l0-norms.
- `grouped_one_norm()`: A vector of group-wise l1-norms.
- `grouped_two_norm()`: A vector of group-wise l2-norms.
- `grouped_sp_norm()`: A vector of length `unique(gr)` consisting of the asparse convex combination of the l1 and l2-norm for each group.
- `gr_one_norm()`: The l1-norm norm of a vector (a scalar).
- `gr_two_norm()`: The sum of the group-wise l2-norms of a vector (a scalar).
- `sp_group_norm()`: The sum of the asparse convex combination of group l1 and l2-norms vectors (a scalar).

Examples

```
x <- c(rep(-1, 5), rep(0, 5), rep(1,5))  
gr <- c(rep(1,5), rep(2,5), rep(3,5))  
aspase <- 0.05  
grouped_sp_norm(x, gr, aspase)
```

Index

* datasets

trust_experts, 14

coef.cv.sparsegl, 2

coef.cv.sparsegl(), 5, 9

coef.sparsegl, 3

coef.sparsegl(), 11, 14

cv.ls(cv.sparsegl), 4

cv.sparsegl, 4

cv.sparsegl(), 2, 5, 7, 9

estimate_risk, 6

ggplot2::ggplot(), 7, 8

glmnet::cv.glmnet(), 4

glmnet::plot.cv.glmnet(), 7

gr_one_norm(zero_norm), 15

gr_two_norm(zero_norm), 15

grouped_one_norm(zero_norm), 15

grouped_sp_norm(zero_norm), 15

grouped_two_norm(zero_norm), 15

grouped_zero_norm(zero_norm), 15

Matrix::sparseMatrix, 4, 12

Matrix::sparseMatrix(), 15

one_norm(zero_norm), 15

plot.cv.sparsegl, 7

plot.cv.sparsegl(), 5

plot.sparsegl, 8

plot.sparsegl(), 14

predict(), 2, 9

predict.cv.sparsegl, 9

predict.cv.sparsegl(), 2, 5

predict.sparsegl, 10

predict.sparsegl(), 3, 9, 11, 14

print.sparsegl, 11

print.sparsegl(), 3, 11, 14

sp_group_norm(zero_norm), 15

sparsegl, 10, 12

sparsegl(), 2–6, 8, 10, 11, 14

stats::extractAIC(), 6

trust_experts, 14

two_norm(zero_norm), 15

zero_norm, 15