Package ‘sparsegl’

September 25, 2023

Type Package
Title Sparse Group Lasso
Version 1.0.2
Date 2023-09-17
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)
       https://dajmcdon.github.io/sparsegl/
BugReports https://github.com/dajmcdon/sparsegl/issues/
Depends R (>= 3.5)
Imports cli, dotCall64, ggplot2, magrittr, Matrix, methods, rlang, RSpectra, tidyr
Suggests dplyr, gglasso, glmnet, knitr, markdown, rmarkdown, splines, testthat (>= 3.0.0)
VignetteBuilder knitr
Config/testthat/edition 3
Encoding UTF-8
LazyData true
NeedsCompilation yes
RoxygenNote 7.2.3
Author Daniel J. McDonald [aut, cre],
          Xiaoxuan Liang [aut],
          Anibal Solón Heinsfeld [aut],
          Aaron Cohen [aut].
coef.cv.sparsegl

Extract coefficients from a cv.sparsegl object.

Description

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

Usage

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

- **object**: Fitted `cv.sparsegl()` object.
- **s**: Value(s) of the penalty parameter `lambda` at which coefficients are desired. Default is the single value `s = "lambda.1se"` stored in the CV object (corresponding to the largest value of `lambda` such that CV error estimate is within 1 standard error of the minimum). Alternatively `s = "lambda.min"` 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.

**Value**

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

**See Also**

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

**Examples**

```r
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))
```

---

**Description**

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

**Usage**

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

**Arguments**

- **object**: Fitted `sparsegl()` 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 of lambda values 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()` and `predict.sparsegl()`.

**Examples**

```r
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**

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

**Usage**

```r
cv.sparsegl(
  x,
  y,
  group = NULL,
  family = c("gaussian", "binomial"),
  lambda = NULL,
  pred.loss = c("default", "mse", "deviance", "mae", "misclass"),
  nfolds = 10,
  foldid = NULL,
  weights = NULL,
  offset = NULL,
  ...
)
```
cv.sparsegl

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 `Matrix::sparseMatrix` are supported.

**y**
Double/Integer/Factor. The response variable. Quantitative for `family="gaussian"` and for other exponential families. If `family="binomial"` 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 or function. Specifies the generalized linear model to use. Valid options are:

- "gaussian" - least squares loss (regression, the default),
- "binomial" - logistic loss (classification)

For any other type, a valid `stats::family()` object may be passed. Note that these will generally be much slower to estimate than the built-in options passed as strings. So for example, `family = "gaussian"` and `family = gaussian()` will produce the same results, but the first will be much faster.

**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:

- "default" the same as deviance (mse for regression and deviance otherwise)
- "mse" mean square error
- "deviance" the default (mse for Gaussian regression, and negative log-likelihood otherwise)
- "mae" mean absolute error, can apply to any family
- "misclass" for classification only, 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.

**weights**
Double vector. Optional observation weights. These can only be used with a `stats::family()` object.

**offset**
Double vector. Optional offset (constant predictor without a corresponding coefficient). These can only be used with a `stats::family()` object.

**...**
Additional arguments to `sparsegl()`.
Details

The function runs `sparsegl()` `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 error over the folds are computed.

Value

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

- `lambda`: The values of `lambda` used in the fits.
- `cvm`: The mean cross-validated error - a vector of length `length(lambda)`.
- `cvsd`: Estimate of standard error of `cvm`.
- `cvupper`: Upper curve = `cvm` + `cvsd`.
- `cvlower`: Lower curve = `cvm` - `cvsd`.
- `name`: A text string indicating type of measure (for plotting purposes).
- `nnzero`: The number of non-zero coefficients for each `lambda`.
- `active_grps`: The number of active groups for each `lambda`.
- `sparsegl.fit`: A fitted `sparsegl()` 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.
- `call`: The function call.

See Also

`sparsegl()`, as well as `plot()`, `predict()`, and `coef()` methods for "cv.sparsegl" objects.

Examples

```r
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)
```
**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**

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

**Arguments**

- `object`: fitted object from a call to `sparsegl()`.
- `x`: Matrix. The matrix of predictors used to estimate the `sparsegl` object. May be missing if `approx_df = TRUE`.
- `type`: one or more of AIC, BIC, or GCV.
- `approx_df`: the df component of a `sparsegl` 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**


**See Also**

`sparsegl()` method.

**Examples**

```r
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)
```
make_irls_warmup  Create starting values for iterative reweighted least squares

Description

This function may be used to create potentially valid starting values for calling `sparsegl()` with a `stats::family()` object. It is not typically necessary to call this function (as it is used internally to create some), but in some cases, especially with custom generalized linear models, it may improve performance.

Usage

```
make_irls_warmup(nobs, nvars, b0 = 0, beta = double(nvars), r = double(nobs))
```

Arguments

- `nobs` Number of observations in the response (or rows in `x`).
- `nvars` Number of columns in `x`
- `b0` Scalar. Initial value for the intercept.
- `beta` Vector. Initial values for the coefficients. Must be length `nvars` (or a scalar).
- `r` Vector. Initial values for the deviance residuals. Must be length `nobs` (or a scalar).

Details

Occasionally, the irls fitting routine may fail with an admonition to create valid starting values.

Value

List of class `irlsspgl_warmup`

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.

Usage

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

- **x**: Fitted "cv.sparsegl" object, produced with `cv.sparsegl()`.
- **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 `ggplot2::ggplot()` plot is produced. Additional user modifications may be added as desired.

See Also

`cv.sparsegl()`.

Examples

```r
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)
```

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

```r
## S3 method for class 'sparsegl'
plot(
  x,
  y_axis = c("coef", "group"),
  x_axis = c("lambda", "penalty"),
  add.legend = n.legend_values < 20,
  ...
)
```
predict.cv.sparsegl

Arguments

x Fitted "sparsegl" object, produced by sparsegl().
y_axis Variable on the y_axis. Either the coefficients (default) or the group norm.
x_axis 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.
add_legend Show the legend. Often, with many groups/predictors, this can become overwhelming. The default produces a legend if the number of groups/predictors is less than 20.
...

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"),
type = c("link", "response", "coefficients", "nonzero", "class"),
... )
Arguments

object  Fitted \texttt{cv.sparsegl}() object.

newx  Matrix of new values for \texttt{x} at which predictions are to be made. Must be a matrix. This argument is mandatory.

s  Value(s) of the penalty parameter \(\lambda\) at which coefficients are desired. Default is the single value \(s = "\text{lambda.1se}"\) stored in the CV object (corresponding to the largest value of \(\lambda\) such that CV error estimate is within 1 standard error of the minimum). Alternatively \(s = "\text{lambda.min}"\) 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.

type  Type of prediction required. Type "link" gives the linear predictors for "binomial"; for "gaussian" models it gives the fitted values. Type "response" gives predictions on the scale of the response (for example, fitted probabilities for "binomial"); for "gaussian" type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for \(s\). 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.

Value

A matrix or vector of predicted values.

See Also

\texttt{cv.sparsegl()} and \texttt{coef.cv.sparsegl()}.

Examples

\begin{verbatim}
  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")
\end{verbatim}
Description
Similar to other predict methods, this function produces fitted values and class labels from a fitted `sparsegl` object.

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

Arguments
- **object**: Fitted `sparsegl()` model object.
- **newx**: Matrix of new values for x at which predictions are to be made. Must be a matrix. This argument is mandatory.
- **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 predictions on the scale of the response (for example, fitted probabilities for "binomial"); for "gaussian" type "response" is equivalent to type "link". Type "coefficients" computes the coefficients at the requested values for s. 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 the new vector of lambda values 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 object returned depends on type.
sparsegl

Description

Fits regularization paths for sparse group-lasso penalized learning problems at a sequence of regularization parameters lambda. Note that the objective function for least squares is

\[
\frac{RSS}{2n} + \lambda \text{penalty}
\]

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

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,
  weights = NULL,
  offset = NULL,
  warm = NULL,
  trace_it = 0,
)
\[
dfmax = \text{as.integer}(\text{max}(\text{group})) + 1L, \\
pmax = \text{min}(\text{dfmax} * 1.2, \text{as.integer}(\text{max}(\text{group}))), \\
\text{eps} = 1e-08, \\
\text{maxit} = 3e+06 
\]

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 \texttt{Matrix::sparseMatrix} are supported.

\(y\) Double/Integer/Factor. The response variable. Quantitative for family="gaussian" and for other exponential families. If family="binomial" 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.

\(\text{group}\) Integer. A vector of consecutive integers describing the grouping of the coefficients (see example below).

\(\text{family}\) Character or function. Specifies the generalized linear model to use. Valid options are:

- "gaussian" - least squares loss (regression, the default),
- "binomial" - logistic loss (classification)

For any other type, a valid \texttt{stats::family()} object may be passed. Note that these will generally be much slower to estimate than the built-in options passed as strings. So for example, \texttt{family = "gaussian"} and \texttt{family = gaussian()} will produce the same results, but the first will be much faster.

\(n\lambda\) The number of \(\lambda\) values - default is 100.

\(\lambda\) A multiplicative factor for the minimal \(\lambda\) in the \(\lambda\) sequence, where \(\text{min}(\lambda) = \lambda\text{.factor} \times \text{max}(\lambda)\). \(\text{max}(\lambda)\) 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 \(\lambda\text{.factor}\) will lead to a saturated fit. This argument has no effect if there is user-defined \(\lambda\) sequence.

\(p\lambda_\text{group}\) 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 \(\beta\)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 \(p\lambda\_\text{sparse}\)). Default value for each entry is the square-root of the corresponding size of each group.
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 only the group penalty.

Whether to include intercept in the model. Default is TRUE.

The relative weight to put on the $\ell_1$-norm in sparse group lasso. Default is 0.05 (resulting in 0.95 on the $\ell_2$-norm).

Logical flag for variable standardization (scaling) prior to fitting the model. Default is TRUE.

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 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.

Double vector. Optional observation weights. These can only be used with a stats::family() object.

Double vector. Optional offset (constant predictor without a corresponding coefficient). These can only be used with a stats::family() object.

List created with make_irls_warmup(). These can only be used with a stats::family() object, and is not typically necessary even then.

Scalar integer. Larger values print more output during the irls loop. Typical values are 0 (no printing), 1 (some printing and a progress bar), and 2 (more detailed printing). These can only be used with a stats::family() object.

Limit the maximum number of groups in the model. Default is no limit.

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.

Convergence termination tolerance. Defaults value is 1e-8.

Maximum number of outer-loop iterations allowed at fixed lambda value. Default is 3e8. If models do not converge, consider increasing maxit.

An object with S3 class "sparsegl". Among the list components:

- call The call that produced this object.
- b0 Intercept sequence of length length(lambda).
- beta A p x length(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.

If sparsegl() was called with a stats::family() method, this may also contain information about the deviance and the family used in fitting.

See Also

cv.sparsegl() and the plot(), predict(), and coef() methods for "sparsegl" objects.

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)

yp <- rpois(n, abs(X %*% beta_star))
fit_pois <- sparsegl(X, yp, group = groups, family = poisson())

trust_experts

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 data.frame with 9759 rows and 8 columns

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

period Factor. Start date of data collection period. There are 13 monthly periods

region Factor. State abbreviation.

age Factor. Self-reported age bucket.
trust_experts

**trust_experts**

gender  Factor. Self-reported gender.

raceethnicity  Factor. Self-reported race or ethnicity.

ccli  Real-valued. This is the wcli indicator measuring the percent of circulating Covid-like illness in a particular region. See the Delphi Epidata API for a complete description.

hh_cmnty_cli  Real-valued. This is the whh_cmnty_cli indicator measuring the percent of people reporting illness in their local community and household.

### Source


The paper describing the survey:


The Public Delphi US CTIS Documentation

### Examples

```r
## Not run:
library(splines)
library(dplyr)
library(magrittr)
df <- 10

trust_experts <- trust_experts %>%
  mutate(across(
    where(is.factor),
    ~ set_attr(.x, "contrasts", contr.sum(nlevels(.x), FALSE, TRUE))
  ))

x <- Matrix::sparse.model.matrix(
  ~ 0 + region + age + gender + raceethnicity + period +
  bs(ccli, df = df) + bs(hh_cmnty_cli, df = df),
  data = trust_experts, drop.unused.levels = TRUE)

gr <- sapply(trust_experts, function(x) ifelse(is.factor(x), nlevels(x), NA))
gr <- rep(seq(ncol(trust_experts) - 1), times = c(gr[!is.na(gr)], df, df))
fit <- cv.sparsegl(x, trust_experts$trust_experts, gr)

## End(Not run)
```
zero_norm  Calculate common norms

Description
Calculate different norms of vectors with or without grouping structures.

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  An integer (or factor) vector of the same length as x.
asparse  Scalar. 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))
asparse <- 0.05

grouped_sp_norm(x, gr, asparse)
Index

* datasets
  trust_experts, 16
  coef(), 6, 16
  coef.cv.sparsegl, 2
  coef.cv.sparsegl(), 11
  coef.sparsegl, 3
  coef.sparsegl(), 13
  cv.sparsegl(), 3, 6, 9–11, 16
  estimate_risk, 7
  ggplot2::ggplot(), 9
  glmnet::cv.glmnet(), 4
  gr_one_norm(zero_norm), 18
  gr_two_norm(zero_norm), 18
  grouped_one_norm(zero_norm), 18
  grouped_sp_norm(zero_norm), 18
  grouped_two_norm(zero_norm), 18
  grouped_zero_norm(zero_norm), 18
  make_irls_warmup, 8
  make_irls_warmup(), 15
  Matrix::sparseMatrix, 5, 14
  one_norm(zero_norm), 18
  plot(), 6, 16
  plot.cv.sparsegl, 8
  plot.sparsegl, 9
  predict(), 6, 16
  predict.cv.sparsegl, 10
  predict.cv.sparsegl(), 3
  predict.sparsegl, 12
  predict.sparsegl(), 4
  sp_group_norm(zero_norm), 18
  sparsegl, 12, 13
  sparsegl(), 2–10, 12, 13
  stats::extractAIC(), 7
  stats::family(), 5, 8, 14–16
  trust_experts, 16
  two_norm(zero_norm), 18
  zero_norm, 18