Package 'holiglm'

June 27, 2023

Type Package

Title Holistic Generalized Linear Models

Version 0.2.6

Description Holistic generalized linear models (HGLMs) extend generalized linear models (GLMs) by enabling the possibility to add further constraints to the model. The 'holiglm' package simplifies estimating HGLMs using convex optimization. Additional information about the package can be found in the reference manual, the 'README' and the accompanying paper <doi:10.48550/arXiv.2205.15447>.

Depends R (>= 3.5.0), ROI.plugin.ecos

Imports slam, checkmate, MASS, SuppDists, ROI (>= 0.3-0)

Encoding UTF-8

Suggests ROI.plugin.scs, tinytest (>= 1.0.0), parallel, knitr, rmarkdown, bookdown, DiagrammeR

License GPL-3

URL https://arxiv.org/abs/2205.15447

VignetteBuilder knitr

RoxygenNote 7.2.3

NeedsCompilation no

Author Benjamin Schwendinger [aut, cre], Florian Schwendinger [aut], Laura Vana [aut]

Maintainer Benjamin Schwendinger <benjaminschwe@gmail.com>

Repository CRAN

Date/Publication 2023-06-27 10:30:05 UTC

R topics documented:

holiglm-package 2
active_coefficients 4
agg_binomial 4

as.OP.hglm_model . . . 5
bike . . . 6
coef.hglm . . . 7
cov_matrix . . . 8
group_equal . . . 8
group_inout . . . 9
group_sparsity . . . 10
hglm . . . 10
hglmc . . . 13
hglm_fit . . . 14
hglm_model . . . 15
include . . . 16
k_max . . . 16
linear . . . 17
lower . . . 18
pairwise_sign_coherence . . . 19
predict.hglm . . . 20
rhglm . . . 21
rho_max . . . 22
scale_constraint_matrix . . . 23
sign_coherence . . . 24
solution.hglm . . . 25
update_objective . . . 25
upper . . . 26

Index **27**

`holiglm-package` *Holistic Generalized Linear Models Package*

Description

The holistic generalized linear models package simplifies estimating generalized linear models under constraints. The constraints can be used to,

- bound the domains of specific covariates,
- impose linear constraints on the covariates,
- induce sparsity via best subset selection,
- impose sparsity on groups of variables,
- restrict the pairwise correlation between the selected coefficients,
- impose sign coherence constraints on selected covariates and
- force all predictors within a group either to be selected or not.

This sophisticated constraints are internally implemented via conic optimization. However, the package is designed such that the user, is not required to be familiar with conic optimization but is only required to have basic R knowledge.

Author(s)

- Benjamin Schwendinger (**Maintainer** <benjaminschwe@gmail.com>)
- Florian Schwendinger
- Laura Vana

References

Holistic regression
Schwendinger, B., Schwendinger, F., & Vana, L. (2022). Holistic Generalized Linear Models. doi:10.48550/ARXIV.2205.15447.

Bertsimas, D., & King, A. (2016). OR Forum-An Algorithmic Approach to Linear Regression Operations Research 64(1):2-16. doi:10.1287/opre.2015.1436

Bertsimas, D., & Li, M. L. (2020). Scalable Holistic Linear Regression. Operations Research Letters 48 (3): 203–8. doi:10.1016/j.orl.2020.02.008.

Constrained regression
McDonald, J. W., & Diamond, I. D. (1990). On the Fitting of Generalized Linear Models with Nonnegativity Parameter Constraints. Biometrics, 46 (1): 201–206. doi:10.2307/2531643

Slawski, M., & Hein, M. (2013). Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. Electronic Journal of Statistics, 7: 3004-3056. doi:10.1214/13EJS868

Carrizosa, E., Olivares-Nadal, A. V., & Ramírez-Cobo, P. (2020). Integer Constraints for Enhancing Interpretability in Linear Regression. SORT. Statistics and Operations Research Transactions, 44: 67-98. doi:10.2436/20.8080.02.95.

Lawson, C. L., & Hanson, R. J. (1995). Solving least squares problems. Society for Industrial and Applied Mathematics. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611971217

Generalized Linear Models
McCullagh, P., & Nelder, J. A. (2019). Generalized Linear Models (2nd ed.) Routledge. doi:10.1201/9780203753736.

Conic Optimization
Boyd, S., & Vandenberghe, L. (2004). Convex Optimization (1st ed.) Cambridge University Press. https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf. doi:10.1017/cbo9780511804441

Theußl, S., Schwendinger, F., & Hornik, K. (2020). ROI: An Extensible R Optimization Infrastructure. Journal of Statistical Software 94 (15): 1–64. doi:10.18637/jss.v094.i15.

See Also

hglm, holiglm

active_coefficients *Obtain all Active Coefficients*

Description

The function returns a logical vector which is TRUE for all active (i.e., non-zero) coefficients in the fitted model and FALSE otherwise.

Usage

```
active_coefficients(object, ...)

acoef(object, ...)
```

Arguments

object	an object inheriting from "hglm" or "hglm.fit" from which the active coefficients obtained from.
...	optional arguments currently ignored.

Value

a logical vector giving the active coefficients.

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
fit <- hglm(y ~ ., constraints = k_max(3), data = dat)
active_coefficients(fit)
```

agg_binomial *Aggregate Binomial Data*

Description

A simple function for aggregating binomial data, from a form where y contains only 0 and 1 and X could contain duplicated rows, into a format where y is the matrix of counted successes and failures and X does not contain duplicates. If X contains factor variables, the model matrix corresponding to X will be returned.

Usage

```
agg_binomial(formula, data, as_list = TRUE)
```

Arguments

formula	a formula object defining the aggregation.
data	a data.frame to be aggregated.
as_list	a logical giving if the return value should be a list. If FALSE the return value is a data.frame.

Value

A list (or data.frame) containing aggregated binomial data with counted successes and failures.

Examples

```
set.seed(12345)
data <- data.frame(y = rbinom(50, 1, 0.7),
                   a = factor(sample(c(1, 2), 50, TRUE)),
                   b = factor(sample(c(1, 2, 3), 50, TRUE)))
agg_binomial(y ~ ., data)
```

---

as.OP.hglm_model *Convert to OP*

Description

Convert an object of class "hglm_model" into a **ROI** optimization problem (OP).

Usage

```
## S3 method for class 'hglm_model'
as.OP(x)
```

Arguments

x	an object inheriting from "hglm_model".

Details

This function is mainly for internal use and advanced users which want of alter the model object or the underlying optimization problem. This function converts the model object created by hglm_model into a conic optimization problem solveable via ROI_solve.

Value

A **ROI** object of class "OP".

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
# Use hglm with option dry_run
model <- hglm(y ~ ., data = dat, dry_run = TRUE)
op <- as.OP(model)
# User hglm_model
x <- model.matrix(y ~ ., data = dat)
model <- hglm_model(x, dat[["y"]])
op <- as.OP(model)
```

---

bike *Bike Sharing Dataset*

Description

This data set contains the daily count of rented bikes from the the Capital Bikeshare system in Washington D.C., USA, for the years 2011 and 2012. The dataset is already prepared (correct types + factor encodings) for model building.

Format

A data.frame of dimension 731 x 12 containing daily data related to related bikes.

dteday a date vector giving the date of the rental.

season a factor with levels ‘spring‘, ‘summer‘, ‘fall‘ and ‘winter‘.

year a factor with levels ‘2011‘ and ‘2012‘.

mnth a factor with levels ‘Jan‘, ‘Feb‘, ‘Mar‘, ‘Apr‘, ‘May‘, ‘Jun‘, ‘Jul‘, ‘Aug‘, ‘Sep‘, ‘Oct‘, ‘Nov‘ and ‘Dec‘.

holiday a boolean vector indicating if the day is a holiday.

weathersit a factor with levels ‘good‘, ‘neutral‘, ‘bad‘ and ‘very bad‘ giving the weather situation.

temp a numeric vector containing max-normalized temperature in Celsius with 41 as maximum.

atemp a numeric vector containing max-normalized feeling temperature in Celsius with 50 as maximum.

hum a numeric vector containing max-normalized humidity with 100 as maximum.

windspeed a numeric vector containing max-normalized windspeed with 67 as maximum.

cnt an integer vector containing counts of rented bikes.

Source

https://archive.ics.uci.edu/

References

Fanaee-T, Hadi. (2013). Bike Sharing Dataset. UCI Machine Learning Repository.

Examples

```
data("bike")
hglm(formula = cnt ~ ., data=bike, family="poisson")
```

`coef.hglm` *Extract Model Coefficients*

Description

The function returns different types of coefficients for a model.

Usage

```
## S3 method for class 'hglm'
coef(object, type = c("unscaled", "scaled", "selected"), ...)
```

Arguments

`object`	an object inheriting from `"hglm"` or `"hglm.fit"` from which the coefficients are obtained from.
`type`	Default value is `"unscaled"`. Allowed values are `"unscaled"`, `"scaled"` and `"selected"`.
`...`	optional arguments currently ignored.

Details

The types `"scaled"` and `"unscaled"` refer to the coefficients of the scaled/unscaled optimization problem. Type `"selected"` refers to the active coefficients in the model `active_coefficients`.

Value

a vector containing the unscaled, scaled or selected coefficients.

Examples

```
dat <- rhglm(1000, 1:3)
fit <- hglm(y ~ ., data = dat)
coef(fit)
coef(fit, type="scaled")
coef(fit, type="selected")
```

cov_matrix *Construct Covariance matrix*

Description

Utility function for constructing covariance matrices based on a simple triplet format (simple_triplet_matrix).

Usage

```
cov_matrix(k, i, j, v)
```

Arguments

k	an integer giving the number of rows and columns of the constructed covariance matrix.
i	an integer vector giving the row indices.
j	an integer vector giving the row indices.
v	a numeric vector giving the corresponding values.

Value

A dense matrix of covariances.

See Also

Other simulation: rhglm()

Examples

```
cov_matrix(5, c(1, 2), c(2, 3), c(0.8, 0.9))
```

group_equal *Group Equal Constraint*

Description

Forces all covariates in the specified group to have the same coefficient.

Usage

```
group_equal(vars)
```

Arguments

vars	a vector specifying the indices or names of the covariates to which the constraint shall be applied.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

See Also

Other Constraint-Constructors: group_inout(), group_sparsity(), include(), k_max(), linear(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence(), upper()

Examples

```
dat <- rhglm(100, c(1, 2, 3, 4, 5, 6))
constraints <- group_equal(vars = c("x1", "x3"))
hglm(y ~ ., constraints = constraints, data = dat)
```

---

group_inout *In-Out Constraint*

Description

Forces coefficients of the covariates in the specified group to be either all zero or all nonzero.

Usage

```
group_inout(vars)
```

Arguments

vars	a vector specifying the indices or names of the covariates to which the constraint shall be applied.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

See Also

Other Constraint-Constructors: group_equal(), group_sparsity(), include(), k_max(), linear(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence(), upper()

Examples

```
dat <- rhglm(100, c(1, 2, 3, 4, 5, 6))
constraints <- group_inout(c("x1", "x2", "x3"))
hglm(y ~ ., constraints = constraints, data = dat)
```

group_sparsity *Group Sparsity Constraint*

Description

Constraint which restricts the number of covariates selected from a specific group.

Usage

```
group_sparsity(vars, k = 1L)
```

Arguments

vars	a vector specifying the indices or names of the covariates to which the group constraint shall be applied.
k	an integer giving the maximum number of covariates to be included in the model from the specified group.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

See Also

Other Constraint-Constructors: group_equal(), group_inout(), include(), k_max(), linear(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence(), upper()

Examples

```
dat <- rhglm(100, c(1, 2, 0, 4, 5, 0))
constraints <- group_sparsity(c("x1", "x2", "x5"), 1L)
hglm(y ~ ., constraints = constraints, data = dat)
```

hglm *Fitting Holistic Generalized Linear Models*

Description

Fit a generalized linear model under holistic constraints.

Usage

```
hglm(
  formula,
  family = gaussian(),
  data,
  constraints = NULL,
  weights = NULL,
 scaler = c("auto", "center_standardization", "center_minmax", "standardization",
    "minmax", "off"),
  scale_response = NULL,
  big_m = 100,
  solver = "auto",
  control = list(),
  dry_run = FALSE,
  object_size = c("normal", "big")
)

holiglm(
  formula,
  family = gaussian(),
  data,
  constraints = NULL,
  weights = NULL,
 scaler = c("auto", "center_standardization", "center_minmax", "standardization",
    "minmax", "off"),
  scale_response = NULL,
  big_m = 100,
  solver = "auto",
  control = list(),
  dry_run = FALSE,
  object_size = c("normal", "big")
)

hglm_seq(
  k_seq,
  formula,
  family = gaussian(),
  data,
  constraints = NULL,
  weights = NULL,
 scaler = c("auto", "center_standardization", "center_minmax", "standardization",
    "minmax", "off"),
  big_m = 100,
  solver = "auto",
  control = list(),
  object_size = c("normal", "big"),
  parallel = FALSE
)
```

Arguments

formula	an object of class "formula" giving the symbolic description of the model to be fitted.
family	a description of the error distribution and link function to be used in the model.
data	a data.frame or matrix giving the data for the estimation.
constraints	a list of 'HGLM' constraints stored in a list of class "lohglmc". Use NULL to turn off constraints.
weights	an optional vector of 'prior weights' to be used for the estimation.
scaler	a character string giving the name of the scaling function (default is "auto") to be employed for the covariates. This typically does not need to be changed.
scale_response	a boolean whether the response shall be standardized or not. Can only be used with family gaussian(). Default is TRUE for family gaussian() and FALSE for other families.
big_m	an upper bound for the coefficients, needed for the big-M constraint. Required to inherit from "hglmc". Currently constraints created by group_sparsity(), group_inout(), include() and group_equal() use the big-M value specified here.
solver	a character string giving the name of the solver to be used for the estimation.
control	a list of control parameters passed to ROI_solve.
dry_run	a logical; if TRUE the model is not fit but only constructed.
object_size	a character string giving the object size, allowed values are "normal" and "big". If "big" is choosen, also the **ROI** solution and the "hglm_model" object are returned.
k_seq	an integer vector giving the values of k_max for which the model should be estimated.
parallel	whether estimation of sequence shall be parallelized

Details

In the case of binding linear constraints the standard errors are corrected, more information about the correction can be found in Schwendinger, Schwendinger and Vana (2022).

Value

An object of class "hglm" inheriting from "glm".

References

Schwendinger B., Schwendinger F., Vana L. (2022). Holistic Generalized Linear Models doi:10.48550/arXiv.2205.15447

Bertsimas, D., & King, A. (2016). OR Forum-An Algorithmic Approach to Linear Regression Operations Research 64(1):2-16. doi:10.1287/opre.2015.1436

McCullagh, P., & Nelder, J. A. (2019). Generalized Linear Models (2nd ed.) Routledge. doi:10.1201/9780203753736.

Dobson, A. J., & Barnett, A. G. (2018). An Introduction to Generalized Linear Models (4th ed.) Chapman and Hall/CRC. doi:10.1201/9781315182780

Chares, Robert. (2009). "Cones and Interior-Point Algorithms for Structured Convex Optimization involving Powers and Exponentials."

Chen, J., & Chen, Z. (2008). Extended Bayesian information criteria for model selection with large model spaces. Biometrika, 95 (3): 759–771. Oxford University Press. doi:10.1093/biomet/asn034

Zhu, J., Wen, C., Zhu, J., Zhang, H., & Wang, X. (2020). A polynomial algorithm for best-subset selection problem. Proceedings of the National Academy of Sciences, 117 (52): 33117–33123. doi:10.1073/pnas.2014241117

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
hglm(y ~ ., constraints = NULL, data = dat)
# estimation without constraints
hglm(y ~ ., constraints = NULL, data = dat)
# estimation with an upper bound on the number of coefficients to be selected
hglm(y ~ ., constraints = k_max(3), data = dat)
# estimation without intercept
hglm(y ~ . - 1, data = dat)
```

---

hglmc *Generic Functions for* hglmc *Objects*

Description

Generic functions for holistic 'GLM' constraints.

Usage

```
## S3 method for class 'hglmc'
c(...)

is.hglmc(x)
```

Arguments

...	multiple objects inheriting from "hglmc" to be combined.
x	an R object.

Details

The 'HGLM' constraints are all of class "hglmc" and can be combined with the typical combine function c(). To verify that an object is a 'HGLM' constraint, the function is.hglmc can be used.

Value

The combine function `c()` returns an object of class `"hglmc"`. The `is.hglmc` function returns `TRUE` if the object inherits from class `"hglmc"` otherwise `FALSE`.

Examples

```
constraints <- c(k_max(7), pairwise_sign_coherence())
is.hglmc(constraints)
```

`hglm_fit` *Fitting Holistic Generalized Linear Models*

Description

Fit a generalized linear model under constraints.

Usage

```
hglm_fit(
  model,
  constraints = NULL,
  big_m,
  solver = "auto",
  control = list(),
  dry_run = FALSE,
  object_size = c("normal", "big")
)
```

Arguments

`model`	a 'HGLM' model (object of class `"hglm_model"`).
`constraints`	a list of 'HGLM' constraints stored in a list of class `"lohglmc"`.
`big_m`	an upper bound for the coefficients, needed for the big-M constraint. Required to inherit from `"hglmc"`. Currently constraints created by `group_sparsity()`, `group_inout()`, `include()` and `group_equal()` use the big-M set here.
`solver`	a character string giving the name of the solver to be used for the estimation.
`control`	a list of control parameters passed to `ROI_solve`.
`dry_run`	a logical if `TRUE` the model is not fit but only constructed.
`object_size`	a character string giving the object size, allowed values are `"normal"` and `"big"`. If `"big"` is choosen, also the **ROI** solution and the `"hglm_model"` object are returned.

Value

an object of class `"hglm.fit"` inheriting from `"glm"`.

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
x <- model.matrix(y ~ ., data = dat)
model <- hglm_model(x, y = dat[["y"]])
fit <- hglm_fit(model, constraints = k_max(3))
```

`hglm_model` *Create a HGLM Model*

Description

Create a HGLM model object.

Usage

```
hglm_model(
  x,
  y,
  family = gaussian(),
  weights = NULL,
  frame = NULL,
  solver = "auto"
)
```

Arguments

`x`	a numeric matrix giving the design matrix.
`y`	a vector giving the response variables.
`family`	a description of the error distribution and link function to be used in the model.
`weights`	an optional vector of 'prior weights' to be used for the estimation.
`frame`	an optional model frame object.
`solver`	a character string giving the name of the solver to be used for the estimation.

Details

No standardization prior to fitting the model takes place. If a x or y standardization is wanted, the user has to do this beforehand.

Value

An object of class `"hglm_model"`.

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
x <- model.matrix(y ~ ., data = dat)
hglm_model(x, y = dat[["y"]])
```

include *Include Constraint*

Description

Ensures that all covariates specified by `vars` coefficients are active.

Usage

```
include(vars)
```

Arguments

`vars`	an integer vector specifying the indices for covariates which have to be in the model.

Value

A holistic generalized model constraint, object inheriting from class `"hglmc"`.

See Also

Other Constraint-Constructors: `group_equal()`, `group_inout()`, `group_sparsity()`, `k_max()`, `linear()`, `lower()`, `pairwise_sign_coherence()`, `rho_max()`, `sign_coherence()`, `upper()`

Examples

```
dat <- rhglm(100, c(1, 2, 3, 4, 5, 6))
constraints <- include(vars = c("x1", "x3"))
hglm(y ~ ., constraints = constraints, data = dat)
```

k_max *Constraint on the Number of Covariates*

Description

Constraint on the maximum number of covariates to be used in the model.

Usage

```
k_max(k)
```

Arguments

`k`	an positive integer with $k \leq k_{max}$ giving the maximum number of covariates to be used in the model.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

Note

- If an intercept is used, the upper bound on $k_{max} + 1$ is given by number of columns of the model matrix.
- If no intercept is used, the upper bound on k_{max} is given by number of columns of the model matrix.

See Also

Other Constraint-Constructors: group_equal(), group_inout(), group_sparsity(), include(), linear(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence(), upper()

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
hglm(y ~ ., constraints = k_max(3), data = dat)
```

---

linear *Linear Constraint*

Description

Linear Constraint

Usage

```
linear(L, dir, rhs, on_big_m = FALSE)
```

Arguments

L	a named vector or matrix defining the linear constraints on the coefficients of the covariates.
dir	a character vector giving the direction of the linear constraints.
rhs	a numeric vector giving the right hand side of the linear constraint.
on_big_m	a logical indicating if the constraint should be imposed on the big-M related binary variables.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

References

Lawson, C. L., & Hanson, R. J. (1995). Solving least squares problems. Society for Industrial and Applied Mathematics. Society for Industrial and Applied Mathematics. doi:10.1137/1.9781611971217

See Also

Other Constraint-Constructors: group_equal(), group_inout(), group_sparsity(), include(), k_max(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence(), upper()

Examples

```
# vector constraint
beta <- c(1, -2, 3)
dat <- rhglm(100, beta)
constraints <- c(linear(c(x1 = 2, x2 = 1), "==", 0), rho_max(1))
hglm(y ~ ., data = dat, constraints = constraints)

# matrix constraint
dat <- rhglm(100, c(1, -2, 3, 4, 5, 6, 7))
mat <- diag(2)
colnames(mat) <- c("x1", "x5")
constraints <- c(linear(mat, c("==", "=="), c(-1, 3)), rho_max(1))
hglm(y ~ ., data = dat, constraints = constraints)
```

---

lower *Lower Bound*

Description

Set a lower bound on the coefficients of specific covariates.

Usage

```
lower(kvars)
```

Arguments

kvars	a named vector giving the lower bounds. The names should correspond to the names of the covariates in the model matrix.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

References

McDonald, J. W., & Diamond, I. D. (1990). On the Fitting of Generalized Linear Models with Nonnegativity Parameter Constraints. Biometrics, 46 (1): 201–206. doi:10.2307/2531643

Slawski, M., & Hein, M. (2013). Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. Electronic Journal of Statistics, 7: 3004-3056. doi:10.1214/13EJS868

See Also

Other Constraint-Constructors: `group_equal()`, `group_inout()`, `group_sparsity()`, `include()`, `k_max()`, `linear()`, `pairwise_sign_coherence()`, `rho_max()`, `sign_coherence()`, `upper()`

Examples

```
set.seed(0)
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
constraints <- lower(c(x2 = 0, x5 = 1))
hglm(y ~ ., constraints = constraints, data = dat)

# non-negative least squares
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
constraints <- lower(setNames(double(5), paste0("x", 1:5)))
hglm(y ~ ., constraints = constraints, data = dat)
```

`pairwise_sign_coherence` *Pairwise Sign Coherence*

Description

Ensures that coefficients of covariates which exhibit strong pairwise correlation have a coherent sign.

Usage

```
pairwise_sign_coherence(
  rho = 0.8,
  exclude = "(Intercept)",
  big_m = 100,
  eps = 1e-06,
  use = c("everything", "all.obs", "complete.obs", "na.or.complete",
    "pairwise.complete.obs"),
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

rho	a value in the range [0,1] specifying the maximum allowed collinearity between pairs of covariates.
exclude	a character vector giving the names of the covariates to be excluded from the constraint (default is "(Intercept)").
big_m	a double giving the big-M parameter.
eps	a double giving the epsilon for the equal sign constraint. Since most numerical solvers can only handle constraints up to some epsilon, e.g., the constraint $Ax \geq b$ is typically transformed to $\|Ax-b\| \geq 0$. By providing an eps> 0 and changing the constraint to $\|Ax - b\| \geq$ eps we can ensure $\|Ax - b\| > 0$.
use	an optional character string giving a method for computing covariances in the presence of missing values. The parameter is passed to cor, therefore see cor for more information.
method	a character string indicating which correlation coefficient is to be computed. The parameter is passed to cor, therefore see cor for more information.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

References

Carrizosa, E., Olivares-Nadal, A. V., & Ramírez-Cobo, P. (2020). Integer Constraints for Enhancing Interpretability in Linear Regression. SORT. Statistics and Operations Research Transactions, 44: 67-98. doi:10.2436/20.8080.02.95.

See Also

Other Constraint-Constructors: group_equal(), group_inout(), group_sparsity(), include(), k_max(), linear(), lower(), rho_max(), sign_coherence(), upper()

Examples

```
constraints <- c(k_max(7), pairwise_sign_coherence())
```

---

predict.hglm *Predict Method for HGLM Fits*

Description

Obtains predictions from a fitted holistic generalized linear model object.

Usage

```
## S3 method for class 'hglm'
predict(object, newdata = NULL, type = c("link", "response"), ...)
```

Arguments

object	a fitted object of class inheriting from "hglm".
newdata	an optional data frame containing new observations for which predictions are to be made. If ommitted, the fitted linear predictors are used.
type	the type of predictions to be made. Possible values are "link" (default) or "response". If "link", the predictions are in the link scale; if "response", the predictions are transformed to the response scale.
...	optional arguments currently ignored.

Value

A vector of predicted values. If type = "link", the predicted values are in the link scale; if type = "response", the predicted values are in the response scale.

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
fit <- hglm(y ~ ., constraints = k_max(3), data = dat)
pred <- predict(fit)
pred2 <- predict(fit, newdata=dat)
```

---

rhglm *Random HGLM Data*

Description

A simple data generator for testing and example purposes.

Usage

```
rhglm(
  n,
  beta,
  sigma = diag(length(beta) - 1L),
  family = gaussian(),
  truncate_mu = FALSE,
  as_list = FALSE,
  ...
)
```

Arguments

n	the number of observations to be created.
beta	a numeric vector giving the magnitude of the coefficients (the first element is assumed to be the intercept).

sigma	a positive-definite symmetric matrix giving the covariance structure of the covariates (passed to MASS::mvrnorm).
family	the family of the inverse link.
truncate_mu	a logical giving if mu should be truncated if necessary.
as_list	a logical (default is FALSE), if TRUE a list is returned otherwise a data.frame is returned.
...	addtional optional parameters. The arguments are passed to the random variables generating function of the response.

Value

A data.frame (or list) containing the generated data.

See Also

Other simulation: cov_matrix()

Examples

```
rhglm(10, 1:5)
rhglm(10, 1:5, family = binomial())
```

---

rho_max *Constraint on the Pairwise Correlation of Covariates*

Description

Constraint which ensures that only one covariate out of a pair of covariates with a correlation of at least rho will be included in the final model.

Usage

```
rho_max(
  rho = 0.8,
  exclude = "(Intercept)",
  use = c("everything", "all.obs", "complete.obs", "na.or.complete",
    "pairwise.complete.obs"),
  method = c("pearson", "kendall", "spearman")
)
```

Arguments

rho	a value in the range [0,1] specifying, the maximum allowed collinearity between pairs of covariates.
exclude	variables to be excluded form the pairwise correlation constraints (default is "(Intercept)").

use — an optional character string giving a method for computing co-variances in the presence of missing values. The parameter is passed to cor, therefore see cor for more information.

method — a character string indicating which correlation coefficient is to be computed. See cor for more information.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

See Also

Other Constraint-Constructors: group_equal(), group_inout(), group_sparsity(), include(), k_max(), linear(), lower(), pairwise_sign_coherence(), sign_coherence(), upper()

Examples

```
beta <- 1:3
Sigma <- cov_matrix(k = length(beta) - 1L, 1, 2, 0.9)
dat <- rhglm(100, beta, sigma = Sigma)
hglm(y ~ ., constraints = rho_max(0.8), data = dat)
```

---

scale_constraint_matrix

Scale Linear Constraint Matrix

Description

Auxiliary function to scale the linear constraint matrices to be consistent with the scaled model matrix.

Usage

```
scale_constraint_matrix(L, xs, ys = 1)
```

Arguments

L — a matrix giving the linear constraints.

xs — a vector of length ncol(L) giving the scaling of the model matrix.

ys — a double giving the scaling of the response.

`sign_coherence` *Sign Coherence Constraint*

Description

Constraint which ensures that the coefficients of the specified covariates have a coherent sign.

Usage

```
sign_coherence(vars, big_m = 100, eps = 1e-06)
```

Arguments

`vars`	a character vector giving the names of the covariates the constraint should be applied to.
`big_m`	a double giving the big-M parameter.
`eps`	a double giving the epsilon used to ensure that the constraint holds.

Value

A holistic generalized model constraint, object inheriting from class `"hglmc"`.

References

Carrizosa, E., Olivares-Nadal, A. V., & Ramírez-Cobo, P. (2020). Integer Constraints for Enhancing Interpretability in Linear Regression. SORT. Statistics and Operations Research Transactions, 44: 67-98. doi:10.2436/20.8080.02.95.

See Also

Other Constraint-Constructors: `group_equal()`, `group_inout()`, `group_sparsity()`, `include()`, `k_max()`, `linear()`, `lower()`, `pairwise_sign_coherence()`, `rho_max()`, `upper()`

Examples

```
dat <- rhglm(100, c(1, -2, 3, 4, 5, 6))
constraints <- sign_coherence(c("x1", "x3"))
hglm(y ~ ., constraints = constraints, data = dat)
```

solution.hglm *Extract Solution*

Description

The solution of the underlying optimization problem, can be accessed via the method `'solution'`.

Usage

```
## S3 method for class 'hglm'
solution(
  x,
 type = c("primal", "dual", "aux", "psd", "msg", "objval", "status", "status_code"),
  force = FALSE,
  ...
)
```

Arguments

`x`	an object of type `'hglm'`.
`type`	a character giving the name of the solution to be extracted.
`force`	a logical to control the return value in the case that the status code is equal to 1 (i.e. something went wrong). By default force is `FALSE` and a solution is only provided if the status code is equal to 0 (i.e. success). If force is `TRUE` the status code is ignored and solutions are returned also where the solver signaled an issue.
`...`	further arguments passed to or from other methods.

Value

the extracted solution.

update_objective *Update the Model Object*

Description

Update the Model Object

Usage

```
update_objective(model, op)
```

Arguments

`model`	an object inheriting from `"hglm_model"`.
`op`	an `OP` giving the new objective.

upper *Upper Bound*

Description

Set a upper bound on the coefficient of specific covariates.

Usage

```
upper(kvars)
```

Arguments

kvars	a named vector giving the upper bounds. The names should correspond to the names of the covariates in the model matrix.

Value

A holistic generalized model constraint, object inheriting from class "hglmc".

References

McDonald, J. W., & Diamond, I. D. (1990). On the Fitting of Generalized Linear Models with Nonnegativity Parameter Constraints. Biometrics, 46 (1): 201–206. doi:10.2307/2531643

Slawski, M., & Hein, M. (2013). Non-negative least squares for high-dimensional linear models: Consistency and sparse recovery without regularization. Electronic Journal of Statistics, 7: 3004-3056. doi:10.1214/13EJS868

See Also

Other Constraint-Constructors: group_equal(), group_inout(), group_sparsity(), include(), k_max(), linear(), lower(), pairwise_sign_coherence(), rho_max(), sign_coherence()

Examples

```
dat <- rhglm(100, c(1, 2, -3, 4, 5, -6))
constraints <- upper(c(x1 = 0, x4 = 1))
hglm(y ~ ., constraints = constraints, data = dat)
```

Index

∗ **Constraint-Constructors**
group_equal, 8
group_inout, 9
group_sparsity, 10
include, 16
k_max, 16
linear, 17
lower, 18
pairwise_sign_coherence, 19
rho_max, 22
sign_coherence, 24
upper, 26
∗ **datasets**
bike, 6
∗ **simulation**
cov_matrix, 8
rhglm, 21
_PACKAGE *(*holiglm-package*)*, 2

acoef *(*active_coefficients*)*, 4
active_coefficients, 4, *7*
agg_binomial, 4
as.OP.hglm_model, 5

bike, 6

c.hglmc *(*hglmc*)*, 13
coef.hglm, 7
cor, *20*, *23*
cov_matrix, 8, *22*

group_equal, 8, *9, 10*, *16–20*, *23, 24*, *26*
group_inout, *9*, 9, *10*, *16–20*, *23, 24*, *26*
group_sparsity, *9*, 10, *16–20*, *23, 24*, *26*

hglm, *3*, 10
hglm_fit, 14
hglm_model, *5*, 15
hglm_seq *(*hglm*)*, 10
hglmc, 13
holiglm, *3*
holiglm *(*hglm*)*, 10
holiglm-package, 2

include, *9, 10*, 16, *17–20*, *23, 24*, *26*
is.hglmc *(*hglmc*)*, 13

k_max, *9, 10*, *16*, 16, *18–20*, *23, 24*, *26*

linear, *9, 10*, *16, 17*, 17, *19, 20*, *23, 24*, *26*
lower, *9, 10*, *16–18*, 18, *20*, *23, 24*, *26*

OP, *5*, *25*

pairwise_sign_coherence, *9, 10*, *16–19*, 19, *23, 24*, *26*
predict.hglm, 20

rhglm, *8*, 21
rho_max, *9, 10*, *16–20*, 22, *24*, *26*
ROI_solve, *5*

scale_constraint_matrix, 23
sign_coherence, *9, 10*, *16–20*, *23*, 24, *26*
simple_triplet_matrix, *8*
solution.hglm, 25

update_objective, 25
upper, *9, 10*, *16–20*, *23, 24*, 26