Package 'SCGLR'

March 26, 2025

Type Package

Title Supervised Component Generalized Linear Regression

Description An extension of the Fisher Scoring Algorithm to combine PLS regression with GLM estimation in the multivariate context. Covariates can also be grouped in themes.

Version 3.1.0

Date 2025-03-24

URL `https://scnext.github.io/SCGLR/`, `https://github.com/scnext/SCGLR`, `https://cran.r-project.org/package=SCGLR`

BugReports `https://github.com/SCnext/SCGLR/issues`

Depends R (>= 3.0.0)

Imports Matrix, Formula, graphics, ggplot2 (>= 3.0.0), grid,pROC (>= 1.9), ade4, rlang, pls

Suggests future, future.apply, progressr

LazyData yes

RoxygenNote 7.3.2

Encoding UTF-8

NeedsCompilation no

Author Guillaume Cornu [aut, cre] (<`https://orcid.org/0000-0002-7523-5176`>),
Frederic Mortier [aut] (<`https://orcid.org/0000-0001-5473-709X`>),
Catherine Trottier [aut],
Xavier Bry [aut],
Jocelyn Chauvet [aut] (<`https://orcid.org/0000-0002-4466-6483`>),
Sylvie Gourlet-Fleury [dtc] (<`https://orcid.org/0000-0002-1136-4307`>, Projet CoForChange <`https://coforchange.cirad.fr`>),
Claude Garcia [dtc] (<`https://orcid.org/0000-0002-7351-0226`>, Projet CoForTips <`https://www.cofortips.org`>),
GAMBAS [fnd] (<`https://gambas.cirad.fr`>)

Maintainer Guillaume Cornu <`gcornu@cirad.fr`>

Repository CRAN

Date/Publication 2025-03-26 16:00:01 UTC

Contents

`critConvergence` *Auxiliary function for controlling SCGLR fitting*

Description

Auxiliary function for `scglr` fitting used to construct a convergence control argument.

Usage

```
critConvergence(tol = 1e-06, maxit = 50)
```

Arguments

`tol`	positive convergence threshold.
`maxit`	integer, maximum number of iterations.

Value

a list containing elements named as the arguments.

`customize`

Plot customization

Description

Parameters used to choose what to plot and how. These parameters are given to `plot.SCGLR` and `pairs.SCGLR`.

Details

Parameter name can be abbreviated (e.g. `pred.col` will be understood as predictors.color).
Options can be set globally using `options("plot.SCGLR")`. It will then provide default values that can be further overriden by giving explicit parameter value.

parameter name	***type (default value). Description.***
`title`	string (NULL). Main title of plot (override built-in).
`labels.auto`	logical (TRUE). Should covariate or predictor labels be aligned with arrows.
`labels.offset`	numeric (0.01). Offset by which labels should be moved from tip of arrows.
`labels.size`	numeric (1). Relative size for labels. Use it to globally alter label size.
`expand`	numeric (1). Expand factor for windows size. Use it for example to make room for clipped lab
`threshold`	numeric. All covariates and/or predictors whose sum of square correlations with the two comp
`observations`	logical (FALSE). Should we draw observations.
`observations.size`	numeric (1). Point size.
`observations.color`	character ("black"). Point color.
`observations.alpha`	numeric (1). Point transparency.
`observations.factor`	logical (FALSE). Paint observations according to factor (specify factor).
`predictors`	logical or array of characters or comma separated string (FALSE). Should we draw predictors
`predictors.color`	string ("red"). Base color used to draw predictors.
`predictors.alpha`	numeric (1). Overall transparency for predictors (0 is transparent, 1 is opaque).
`predictors.arrows`	logical (TRUE). Should we draw arrows for predictors.
`predictors.arrows.color`	string (predictors.color). Specific color for predictor arrows.
`predictors.arrows.alpha`	numeric (predictors.alpha). Transparency for predictor arrows.
`predictors.labels`	logical (TRUE). Should we draw labels for predictors.
`predictors.labels.color`	string (predictors.color). Specific color for predictor labels.
`predictors.labels.alpha`	numeric (predictors.alpha). Transparency for predictor labels.
`predictors.labels.size`	numeric (labels.size). Specific size for predictor labels.
`predictors.labels.auto`	logical (labels.auto). Should predictor labels be aligned with arrows.
`covariates`	logical or array of characters or comma separated string (TRUE). Should we draw covariates a
`covariates.color`	string ("black"). Base color used to draw covariates.
`covariates.alpha`	numeric (1). Overall transparency for covariates (0 is transparent, 1 is opaque).
`covariates.arrows`	logical (TRUE). Should we draw arrows for covariates.
`covariates.arrows.color`	string (covariates.color). Specific color for covariate arrows.
`covariates.arrows.alpha`	numeric (covariates.alpha). Transparency for covariate arrows.
`covariates.labels`	logical (TRUE). Should we draw labels for predictors.
`covariates.labels.color`	string (covariates.color). Specific color for predictor labels.
`covariates.labels.alpha`	numeric (covariates.alpha). Transparency for covariate labels.
`covariates.labels.size`	numeric (labels.size). Specific size for covariate labels.

`covariates.labels.auto`	logical (labels.auto). Should covariate labels be aligned with arrows.
`factor`	logical or character (FALSE). Should we draw a factor chosen among additionnal variables (T
`factor.points`	logical (TRUE). Should symbol be drawn for factors.
`factor.points.size`	numeric (4). Symbol size.
`factor.points.shape`	numeric (13). Point shape.
`factor.labels`	logical (TRUE). Should factor labels be drawn.
`factor.labels.color`	string ("black"). Color used to draw labels.
`factor.labels.size`	numeric (labels.size). Specific size for factor labels.

Examples

```
## Not run:
# setting parameters
plot(genus.scglr)
plot(genus.scglr, covariates=c("evi_1","pluvio_11"))
plot(genus.scglr, covariates="evi_1,pluvio_11")
plot(genus.scglr, predictors=TRUE)
plot(genus.scglr, predictors=TRUE, pred.arrows=FALSE)

# setting global style
options(plot.SCGLR=list(predictors=TRUE, pred.arrows=FALSE))
plot(genus.scglr)

# setting custom style
myStyle <- list(predictors=TRUE, pred.arrows=FALSE)
plot(genus.scglr, style=myStyle)

## End(Not run)
```

`dataGen` *Sample dataset of abundance of genera in tropical moist forest*

Description

dataGen gives the abundance of 8 common tree genera in the tropical moist forest of the Congo-Basin and 58 geo-referenced variables on 2615 8-by-8 km plots (observations). Each plot's data was obtained by aggregating the data measured on a variable number of previously sampled 0.5 ha subplots. Geo-referenced environmental variables were used to describe the physical factors as well as vegetation characteristics. On each plot, 34 physical factors were used pertaining the description of topography, geology, rainfall... Vegetation is characterized through 16-days enhanced vegetation index (EVI) data.

Format

`Y`	matrix giving the abundance of 8 common genera (matrix size = 2615*8).
`X`	matrix of 56 geo-referenced environmental variables (matrix size = 2615*56).
`AX`	matrix of 2 additionnal explanatory variables (geology and anthropic interference).

`offset`	sampled area.
`random`	forest concession id number.

Note

The use of this dataset for publication must make reference to the CoForChange project.

Author(s)

CoForChange project

References

S. Gourlet-Fleury et al. (2009–2014) CoForChange project: https://coforchange.cirad.fr/

C. Garcia et al. (2013–2015) CoForTips project: https://www.cofortips.org/

`genus` *Sample dataset of abundance of genera in tropical moist forest*

Description

Genus gives the abundance of 27 common tree genera in the tropical moist forest of the Congo-Basin and 40 geo-referenced environmental variables on one thousand 8 by 8 km plots (observations). Each plot's data was obtained by aggregating the data measured on a variable number of previously sampled 0.5 ha sub-plots. Geo-referenced environmental variables were used to describe the physical factors as well as vegetation characteristics. 14 physical factors were used pertaining the description of topography, geology and rainfall of each plot. Vegetation is characterized through 16-days enhanced vegetation index (EVI) data.

Format

`gen1` to `gen27`	abundance of the 27 common genera.
`altitude`	above-sea level in meters.
`pluvio_yr`	mean annual rainfall.
`forest`	classified into seven classes.
`pluvio_1` to `pluvio_12`	monthly rainfalls.
`geology`	5-level geological substrate.
`evi_1` to `evi_23`	16-days enhanced vegetation indexes.
`lon` and `lat`	position of the plot centers.
`surface`	sampled area.

Note

The use of this dataset for publication must make reference to the CoForChange project.

Author(s)

CoForChange project

References

S. Gourlet-Fleury et al. (2009–2014) CoForChange project: https://coforchange.cirad.fr/

C. Garcia et al. (2013–2015) CoForTips project: https://www.cofortips.org/

`genus2` *Sample dataset of abundance of genera in tropical moist forest*

Description

genus2 gives the abundance of 15 common tree genera in the tropical moist forest of the Congo-Basin and 46 geo-referenced environmental variables on one thousand 8 by 8 km plots (observations). Each plot's data was obtained by aggregating the data measured on a variable number of previously sampled 0.5 ha sub-plots. Geo-referenced environmental variables were used to describe the physical factors as well as vegetation characteristics. 23 physical factors were used pertaining the description of topography, geology and rainfall of each plot. Vegetation is characterized through 16-days enhanced vegetation index (EVI) data.

Format

`gen1` to `gen15`	abundance of 15 common genera.
`evi_1` to `evi_23`	16-days enhanced vegetation indexes.
`MIR` and `NIR`	Middle-Infrared and Near-Infrared channels.
`pluvio_an`	mean annual rainfall.
`pluvio_1` to `pluvio_12`	monthly rainfalls.
`altitude`	above-sea level in meters.
`mois_sec_50` and `mois_sec_50`	???
`CWD`, `awd` and `mcwd`	???
`wetness`	???
`center_x` and `center_y`	longitude and latitude of the plot centers.
`geology`	5-level geological substrate.
`inventory`	forest concession id number.
`surface`	sampled area.

Note

The use of this dataset for publication must make reference to the CoForChange project.

Author(s)

CoForChange project

References

S. Gourlet-Fleury et al. (2009–2014) CoForChange project: https://coforchange.cirad.fr/

C. Garcia et al. (2013–2015) CoForTips project: https://www.cofortips.org/

`kCompRand` *Function that fits the mixed-SCGLR model*

Description

Calculates the components to predict all the response variables.

Usage

```
kCompRand(
  Y,
  family,
  size = NULL,
  X,
  AX = NULL,
  random,
  loffset = NULL,
  k,
  init.sigma = rep(1, ncol(Y)),
  init.comp = c("pca", "pls"),
 method = methodSR("vpi", l = 4, s = 1/2, maxiter = 1000, epsilon = 10^-6, bailout =
    1000)
)
```

Arguments

`Y`	the matrix of random responses
`family`	a vector of character of the same length as the number of response variables: "bernoulli", "binomial", "poisson" or "gaussian" is allowed.
`size`	describes the number of trials for the binomial dependent variables: a (number of observations * number of binomial response variables) matrix is expected.
`X`	the matrix of the standardized explanatory variables
`AX`	the matrix of the additional explanatory variables
`random`	the vector giving the group of each unit (factor)
`loffset`	a matrix of size (number of observations * number of Poisson response variables) giving the log of the offset associated with each observation

`k`	number of components, default is one
`init.sigma`	a vector giving the initial values of the variance components, default is rep(1, ncol(Y))
`init.comp`	a character describing how the components (loadings-vectors) are initialized in the PING algorithm: "pca" or "pls" is allowed.
`method`	Regularization criterion type: object of class "method.SCGLR" built by function `methodSR`.

Value

an object of the SCGLR class.

Examples

```
## Not run:
library(SCGLR)
# load sample data
data(dataGen)
k.opt=4
s.opt=0.1
l.opt=10
withRandom.opt=kCompRand(Y=dataGen$Y, family=rep("poisson", ncol(dataGen$Y)),
                        X=dataGen$X, AX=dataGen$AX,
                        random=dataGen$random, loffset=log(dataGen$offset), k=k.opt,
                        init.sigma = rep(1, ncol(dataGen$Y)), init.comp = "pca",
                        method=methodSR("vpi", l=l.opt, s=s.opt,
                                        maxiter=1000, epsilon=10^-6, bailout=1000))
plot(withRandom.opt, pred=TRUE, plane=c(1,2), title="Component plane (1,2)",
     threshold=0.7, covariates.alpha=0.4, predictors.labels.size=6)

## End(Not run)
```

`methodSR` *Regularization criterion types*

Description

Regularization criterion types

Usage

```
methodSR(
  phi = "vpi",
  l = 1,
  s = 1/2,
  maxiter = 1000,
  epsilon = 1e-06,
  bailout = 10
)
```

Arguments

`phi`	character string describing structural relevance used in the regularization process. Allowed values are "vpi" for Variable Powered Inertia and "cv" for Component Variance. Default to "vpi".
`l`	is an integer argument (>1) tuning the importance of variable bundle locality.
`s`	is a numeric argument (in [0,1]) tuning the strength of structural relevance with respect to goodness of fit.
`maxiter`	integer for maximum number of iterations of `SR` function
`epsilon`	positive convergence threshold
`bailout`	integer argument

Value

an object of class `method.SCGLR`

`multivariateFormula` *Formula construction*

Description

Helper function for building multivariate scglr formula.

NOTE: Interactions involving factors are not allowed for now. For interactions between two quantitative variables, use `I(x*y)` as usual.

Usage

```
multivariateFormula(Y, X = NULL, ..., A = NULL, additional = NULL, data = NULL)
```

Arguments

`Y`	a formula or a vector of character containing the names of the dependent variables.
`X`	a vector of character containing the names of the covariates (X) involved in the components or a list of it.
`...`	additional groups of covariates (theme)
`A`	a vector of character containing the names of the additional covariates.
`additional`	logical (if A is not provided, should we consider last X to be additional covariates)
`data`	a data frame against which formula's variable will be checked

Details

If Y is given as a formula, groups of covariates must be separated by `|` (pipes). To declare last group as additional covariates, one can use `||` (double pipes) as last group separator or set `additional` parameter as `TRUE`.

Value

an object of class `MultivariateFormula`, `Formula`, `formula` with additional attributes: Y, X, A, X_vars, Y_vars,A_vars,XA_vars, YXA_vars, additional

Examples

```
## Not run:
# build multivariate formula
ny <- c("y1","y2")
nx1 <- c("x11","x12")
nx2 <- c("x21","x22")
nadd <- c("add1","add2")
form <- multivariateFormula(ny,nx1,nx2,nadd,additional=T)
form2 <- multivariateFormula(ny,list(nx1,nx2,nadd),additional=T)
form3 <- multivariateFormula(ny,list(nx1,nx2),A=nadd)
form4 <- multivariateFormula(y1+y2~x11+x12|x21+x22||add1+add2)
# Print formulas
form
form2
form3

## End(Not run)
```

`pairs.SCGLR` *Pairwise scglr plot on components*

Description

Pairwise scglr plot on components

Usage

```
## S3 method for class 'SCGLR'
pairs(x, ..., nrow = NULL, ncol = NULL, components = NULL)
```

Arguments

`x`	object of class 'SCGLR', usually a result of running `scglr`.
`...`	optionally, further arguments forwarded to `plot.SCGLR`.
`nrow`	number of rows of the grid layout.
`ncol`	number of columns of the grid layout.
`components`	vector of integers selecting components to plot (default is all components).

Value

an object of class `ggplot`.

See Also

For pairs application see examples in `plot.SCGLR`

`plot.SCGLR` *SCGLR generic plot*

Description

SCGLR generic plot

Usage

```
## S3 method for class 'SCGLR'
plot(x, ..., style = getOption("plot.SCGLR"), plane = c(1, 2))
```

Arguments

`x`	an object from SCGLR class.
`...`	optional arguments (see customize).
`style`	named list of values used to customize the plot (see customize)
`plane`	a size-2 vector (or string with separator) indicating which components are plotted (eg: c(1,2) or "1,2" or "1/2").

Value

an object of class `ggplot`.

Examples

```
## Not run:
library(SCGLR)

# load sample data
data(genus)

# get variable names from dataset
n <- names(genus)
ny <- n[grep("^gen",n)]    # Y <- names that begins with "gen"
nx <- n[-grep("^gen",n)]   # X <- remaining names

# remove "geology" and "surface" from nx
# as surface is offset and we want to use geology as additional covariate
nx <-nx[!nx%in%c("geology","surface")]

# build multivariate formula
# we also add "lat*lon" as computed covariate
form <- multivariateFormula(ny,c(nx,"I(lat*lon)"),c("geology"))
```

```
# define family
fam <- rep("poisson",length(ny))

genus.scglr <- scglr(formula=form,data = genus,family=fam, K=4,
 offset=genus$surface)

summary(genus.scglr)

barplot(genus.scglr)

plot(genus.scglr)

plot(genus.scglr, predictors=TRUE, factor=TRUE)

pairs(genus.scglr)


## End(Not run)
```

`plot.SCGLRTHM` *SCGLRTHM generic plot*

Description

SCGLR generic plot for themes

Usage

```
## S3 method for class 'SCGLRTHM'
plot(x, ...)
```

Arguments

`x`	object of class 'SCGLRTHM', usually a result of running `scglrTheme`.
`...`	see SCGLR plot method

Value

an object of class `ggplot`.

`scglr` *Function that fits the scglr model*

Description

Calculates the components to predict all the dependent variables.

Usage

```
scglr(
  formula,
  data,
  family,
  K = 1,
  size = NULL,
  weights = NULL,
  offset = NULL,
  subset = NULL,
  na.action = na.omit,
  crit = list(),
  method = methodSR()
)
```

Arguments

`formula`	an object of class `MultivariateFormula` (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	a data frame to be modeled.
`family`	a vector of character of the same length as the number of dependent variables: "bernoulli", "binomial", "poisson" or "gaussian" is allowed.
`K`	number of components, default is one.
`size`	describes the number of trials for the binomial dependent variables. A (number of statistical units * number of binomial dependent variables) matrix is expected.
`weights`	weights on individuals (not available for now)
`offset`	used for the poisson dependent variables. A vector or a matrix of size: number of observations * number of Poisson dependent variables is expected.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set to `na.omit`.
`crit`	a list of two elements : maxit and tol, describing respectively the maximum number of iterations and the tolerance convergence criterion for the Fisher scoring algorithm. Default is set to 50 and 10e-6 respectively.
`method`	structural relevance criterion. Object of class "method.SCGLR" built by `methodSR` for Structural Relevance.

Value

an object of the SCGLR class.

The function `summary` (i.e., `summary.SCGLR`) can be used to obtain or print a summary of the results.

An object of class `"SCGLR"` is a list containing following components:

`u`	matrix of size (number of regressors * number of components), contains the component-loadings, i.e. the coefficients of the regressors in the linear combination giving each component.
`comp`	matrix of size (number of statistical units * number of components) having the components as column vectors.
`compr`	matrix of size (number of statistical units * number of components) having the standardized components as column vectors.
`gamma`	list of length number of dependant variables. Each element is a matrix of coefficients, standard errors, z-values and p-values.
`beta`	matrix of size (number of regressors + 1 (intercept) * number of dependent variables), contains the coefficients of the regression on the original regressors X.
`lin.pred`	data.frame of size (number of statistical units * number of dependent variables), the fitted linear predictor.
`xFactors`	data.frame containing the nominal regressors.
`xNumeric`	data.frame containing the quantitative regressors.
`inertia`	matrix of size (number of components * 2), contains the percentage and cumulative percentage of the overall regressors' variance, captured by each component.
`logLik`	vector of length (number of dependent variables), gives the likelihood of the model of each y_k's GLM on the components.
`deviance.null`	vector of length (number of dependent variables), gives the deviance of the null model of each y_k's GLM on the components.
`deviance.residual`	vector of length (number of dependent variables), gives the deviance of the model of each y_k's GLM on the components.

References

Bry X., Trottier C., Verron T. and Mortier F. (2013) Supervised Component Generalized Linear Regression using a PLS-extension of the Fisher scoring algorithm. *Journal of Multivariate Analysis*, 119, 47-60.

Examples

```
## Not run:
library(SCGLR)

# load sample data
data(genus)

# get variable names from dataset
```

```
n <- names(genus)
ny <- n[grep("^gen",n)]    # Y <- names that begins with "gen"
nx <- n[-grep("^gen",n)]   # X <- remaining names

# remove "geology" and "surface" from nx
# as surface is offset and we want to use geology as additional covariate
nx <-nx[!nx%in%c("geology","surface")]

# build multivariate formula
# we also add "lat*lon" as computed covariate
form <- multivariateFormula(ny,c(nx,"I(lat*lon)"),A=c("geology"))

# define family
fam <- rep("poisson",length(ny))

genus.scglr <- scglr(formula=form,data = genus,family=fam, K=4,
 offset=genus$surface)

summary(genus.scglr)

## End(Not run)
```

`scglrCrossVal` *Function that fits and selects the number of component by cross-validation.*

Description

Function that fits and selects the number of component by cross-validation.

Usage

```
scglrCrossVal(
  formula,
  data,
  family,
  K = 1,
  folds = 10,
  type = "mspe",
  size = NULL,
  offset = NULL,
  na.action = na.omit,
  crit = list(),
  method = methodSR(),
  nfolds,
  mc.cores
)
```

Arguments

`formula`	an object of class "Formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
`data`	the data frame to be modeled.
`family`	a vector of character of length q specifying the distributions of the responses. Bernoulli, binomial, poisson and gaussian are allowed.
`K`	number of components, default is one.
`folds`	number of folds, default is 10. Although folds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. folds can also be provided as a vector (same length as data) of fold identifiers.
`type`	loss function to use for cross-validation. Currently six options are available depending on whether the responses are of the same distribution family. If the responses are all bernoulli distributed, then the prediction performance may be measured through the area under the ROC curve: type = "auc" In any other case one can choose among the following five options ("likelihood","aic","aicc","bic","mspe").
`size`	specifies the number of trials of the binomial variables included in the model. A (n*qb) matrix is expected for qb binomial variables.
`offset`	used for the poisson dependent variables. A vector or a matrix of size: number of observations * number of Poisson dependent variables is expected.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set to the `na.omit`.
`crit`	a list of two elements : maxit and tol, describing respectively the maximum number of iterations and the tolerance convergence criterion for the Fisher scoring algorithm. Default is set to 50 and 10e-6 respectively.
`method`	Regularization criterion type. Object of class "method.SCGLR" built by `methodSR` for Structural Relevance.
`nfolds`	deprecated. Use `fold` parameter instead.
`mc.cores`	deprecated

Value

a matrix containing the criterion values for each response (rows) and each number of components (columns).

References

Bry X., Trottier C., Verron T. and Mortier F. (2013) Supervised Component Generalized Linear Regression using a PLS-extension of the Fisher scoring algorithm. *Journal of Multivariate Analysis*, 119, 47-60.

Examples

```
## Not run:
library(SCGLR)
```

```
# load sample data
data(genus)

# get variable names from dataset
n <- names(genus)
ny <- n[grep("^gen",n)]    # Y <- names that begins with "gen"
nx <- n[-grep("^gen",n)]   # X <- remaining names

# remove "geology" and "surface" from nx
# as surface is offset and we want to use geology as additional covariate
nx <-nx[!nx%in%c("geology","surface")]

# build multivariate formula
# we also add "lat*lon" as computed covariate
form <- multivariateFormula(ny,c(nx,"I(lat*lon)"),A=c("geology"))

# define family
fam <- rep("poisson",length(ny))

# cross validation
genus.cv <- scglrCrossVal(formula=form, data=genus, family=fam, K=12,
 offset=genus$surface)

# find best K
mean.crit <- colMeans(log(genus.cv))

#plot(mean.crit, type="l")

## End(Not run)
```

`scglrTheme` *Function that fits the theme model*

Description

Calculates the components to predict all the dependent variables.

Usage

```
scglrTheme(
  formula,
  data,
  H,
  family,
  size = NULL,
  weights = NULL,
  offset = NULL,
  subset = NULL,
  na.action = na.omit,
```

```
  crit = list(),
  method = methodSR(),
  st = FALSE
)
```

Arguments

`formula`	an object of class "`MultivariateFormula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
`data`	data frame.
`H`	vector of R integer. Number of components to keep for each theme
`family`	a vector of character of the same length as the number of dependent variables: "bernoulli", "binomial", "poisson" or "gaussian" is allowed.
`size`	describes the number of trials for the binomial dependent variables. A (number of statistical units * number of binomial dependent variables) matrix is expected.
`weights`	weights on individuals (not available for now)
`offset`	used for the poisson dependent variables. A vector or a matrix of size: number of observations * number of Poisson dependent variables is expected.
`subset`	an optional vector specifying a subset of observations to be used in the fitting process.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set to `na.omit`.
`crit`	a list of two elements : maxit and tol, describing respectively the maximum number of iterations and the tolerance convergence criterion for the Fisher scoring algorithm. Default is set to 50 and 10e-6 respectively.
`method`	structural relevance criterion. Object of class "method.SCGLR" built by `methodSR` for Structural Relevance.
`st`	logical (FALSE) theme build and fit order. TRUE means random, FALSE means sequential (T1, ..., Tr)

Details

Models for theme are specified symbolically.

A model as the form `response ~ terms` where `response` is the numeric response vector and terms is a series of R themes composed of predictors.

Themes are separated by "|" (pipe) and are composed. ... Y1+Y2+... ~ X11+X12+...+X1_ | X21+X22+... | ...+X1_+... | A1+A2+...

See `multivariateFormula`.

Value

a list of SCGLRTHM class. Each element is a SCGLR object

Examples

```
## Not run:
library(SCGLR)

# load sample data
data(genus)

# get variable names from dataset
n <- names(genus)
n <-n[!n%in%c("geology","surface","lon","lat","forest","altitude")]
ny <- n[grep("^gen",n)]    # Y <- names that begins with "gen"
nx1 <- n[grep("^evi",n)]   # X <- remaining names
nx2 <- n[-c(grep("^evi",n),grep("^gen",n))]


form <- multivariateFormula(ny,nx1,nx2,A=c("geology"))
fam <- rep("poisson",length(ny))
testthm <-scglrTheme(form,data=genus,H=c(2,2),family=fam,offset = genus$surface)
plot(testthm)

## End(Not run)
```

`scglrThemeBackward` *Theme Backward selection*

Description

Perform component selection by cross-validation backward approach

Usage

```
scglrThemeBackward(
  formula,
  data,
  H,
  family,
  size = NULL,
  weights = NULL,
  offset = NULL,
  na.action = na.omit,
  crit = list(),
  method = methodSR(),
  folds = 10,
  type = "mspe",
  st = FALSE
)
```

Arguments

`formula`	an object of class "`Formula`" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given under Details.
`data`	data frame.
`H`	vector of R integer. Number of components to keep for each theme
`family`	a vector of character of the same length as the number of dependent variables: "bernoulli", "binomial", "poisson" or "gaussian" is allowed.
`size`	describes the number of trials for the binomial dependent variables. A (number of statistical units * number of binomial dependent variables) matrix is expected.
`weights`	weights on individuals (not available for now)
`offset`	used for the poisson dependent variables. A vector or a matrix of size: number of observations * number of Poisson dependent variables is expected.
`na.action`	a function which indicates what should happen when the data contain NAs. The default is set to `na.omit`.
`crit`	a list of two elements : maxit and tol, describing respectively the maximum number of iterations and the tolerance convergence criterion for the Fisher scoring algorithm. Default is set to 50 and 10e-6 respectively.
`method`	structural relevance criterion. Object of class "method.SCGLR" built by `methodSR` for Structural Relevance.
`folds`	number of folds - default is 10. Although folds can be as large as the sample size (leave-one-out CV), it is not recommended for large datasets. Smallest value allowable is folds=2. folds can also be provided as a vector (same length as data) of fold identifiers.
`type`	loss function to use for cross-validation. Currently six options are available depending on whether the responses are of the same distribution family. If the responses are all bernoulli distributed, then the prediction performance may be measured through the area under the ROC curve: type = "auc" In any other case one can choose among the following five options ("likelihood","aic","aicc","bic","mspe").
`st`	logical (FALSE) theme build and fit order. TRUE means random, FALSE means sequential (T1, ..., Tr)

Details

Models for theme are specified symbolically.

A model as the form `response ~ terms` where `response` is the numeric response vector and terms is a series of R themes composed of predictors.

Themes are separated by "|" (pipe) and are composed.
y1 + y2 + . . . ~ x11 + x12 + . . . + x1_ | x21 + x22 + . . . | . . . + x1_ + . . . | a1 + a2 + . . .

See `multivariateFormula`.

Value

a list containing the path followed along the selection process, the associated mean square predictor error and the best configuration.

Examples

```
## Not run:
library(SCGLR)

# load sample data
data(genus)

# get variable names from dataset
n <- names(genus)
n <- n[!n %in% c("geology","surface","lon","lat","forest","altitude")]
ny <- n[grep("^gen",n)]    # Y <- names that begins with "gen"
nx1 <- n[grep("^evi",n)]   # X <- remaining names
nx2 <- n[-c(grep("^evi",n),grep("^gen",n))]


form <- multivariateFormula(ny,nx1,nx2,A=c("geology"))
fam <- rep("poisson",length(ny))
testcv <- scglrThemeBackward(form,data=genus,H=c(2,2),family=fam,offset = genus$surface,folds=3)

# Cross-validation pathway
testcv$H_path

# Plot criterion
plot(testcv$cv_path)

# Best combination
testcv$H_best

## End(Not run)
```

`screeplot.SCGLR` *Screeplot of percent of overall X variance captured by component*

Description

Screeplot of percent of overall X variance captured by component

Usage

```
## S3 method for class 'SCGLR'
screeplot(x, ...)
```

Arguments

`x`	object of class 'SCGLR', usually a result of running `scglr`.
`...`	optional arguments.

Value

an object of class `ggplot`.

See Also

For screeplot application see examples in `plot.SCGLR`.

`screeplot.SCGLRTHM` *Screeplot of percent of overall X variance captured by component*

Description

Screeplot of percent of overall X variance captured by component by theme

Usage

```
## S3 method for class 'SCGLRTHM'
screeplot(x, ...)
```

Arguments

`x`	object of class 'SCGLRTHM', usually a result of running `scglrTheme`.
`...`	optional arguments.

Value

an object of class `ggplot`.

Index