Package ‘MLGL’

January 16, 2020

Type Package
Title Multi-Layer Group-Lasso
Version 0.6.1
Date 2019-12-30
Copyright Inria
Description It implements a new procedure of variable selection in the context of redundancy between explanatory variables, which holds true with high dimensional data (Grimonprez et al. (2018) <https://hal.inria.fr/hal-01857242>).
License GPL (>= 2)
Imports gglasso, MASS, Matrix, fastcluster, FactoMineR, parallelDist
RoxygenNote 7.0.2
Encoding UTF-8
Author Quentin Grimonprez [aut, cre],
    Samuel Blanck [ctb],
    Alain Celisse [ths],
    Guillemette Marot [ths],
    Yi Yang [ctb],
    Hui Zou [ctb]
Maintainer Quentin Grimonprez <quentin.grimonprez@inria.fr>
Repository CRAN
Repository/R-Forge/Project hcglasso
Repository/R-Forge/Revision 48
Repository/R-Forge/DateTimestamp 2019-12-30 16:01:10
Date/Publication 2020-01-16 08:50:03 UTC
NeedsCompilation no

R topics documented:

MLGL-package

bootstrapHclust
This package presents a method combining Hierarchical Clustering and Group-lasso. Usually, a single partition of the covariates is used in the group-lasso. Here, we provide several partition from the hierarchical tree.

A post-treatment method based on statistical test (with FWER and FDR control) for selecting the regularization parameter and the optimal group for this value is provided. This method can be applied for the classical group-lasso and our method.
Details

The function `MLGL` performs the hierarchical clustering and the group-lasso. The post-treatment method can be performed with `hierarchicalFWER` and `selFWER` functions. The whole process can be run with the `fullProcess` function.

Author(s)

Quentin Grimonprez

Maintainer: Quentin Grimonprez <quentin.grimonprez@inria.fr>

References

"MLGL: An R package implementing correlated variable selection by hierarchical clustering and group-Lasso.", Quentin Grimonprez, Samuel Blanck, Alain Celisse, Guillemette Marot (2018). [https://hal.inria.fr/hal-01857242](https://hal.inria.fr/hal-01857242)

See Also

`MLGL`, `cv.MLGL`, `fullProcess`, `hierarchicalFWER`

Examples

```r
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- MLGL(X, y)
```

---

### bootstrapHclust

**Hierarchical Clustering with distance matrix computed using bootstrap replicates**

**Description**

Hierarchical Clustering with distance matrix computed using bootstrap replicates

**Usage**

```r
bootstrapHclust(X, frac = 1, B = 50, method = "ward.D2", nCore = NULL)
```
coef.cv.MLGL

 Arguments

 X          data
 frac       fraction of sample used at each replicate
 B          number of replicates
 method     desired method: "single", "complete", "average", "mcquitty", "ward.D", "ward.D2",
            "centroid", "median".
 nCore      number of cores

 Value

 An object of class hclust

 Examples

 hc <- bootstrapHclust(USArrests, nCore = 1)

 coef.cv.MLGL Get coefficients from a cv.MLGL object

 Description

 Get coefficients from a cv.MLGL object

 Usage

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

 Arguments

 object  cv.MLGL object
 s       Either "lambda.1se" or "lambda.min"
 ...     Not used. Other arguments to predict.

 Value

 A matrix with estimated coefficients for given values of s.

 Author(s)

 Quentin Grimonprez

 See Also

 cv.MLGL, predict.cv.MLGL
**Description**

Get coefficients from a MLGL object

**Usage**

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

**Arguments**

- `object`: MLGL object
- `s`: values of lambda. If NULL, use values from object
- `...`: Not used. Other arguments to predict.

**Value**

A matrix with estimated coefficients for given values of `s`.

**Author(s)**

Quentin Grimonprez

**See Also**

MLGL, predict.MLGL

---

**computeGroupSizeWeight**

*Compute the group size weight vector with an authorized maximal size*

**Description**

Compute the group size weight vector with an authorized maximal size

**Usage**

```r
computeGroupSizeWeight(hc, sizeMax = NULL)
```

**Arguments**

- `hc`: output of hclust
- `sizeMax`: maximum size of cluster to consider
Value
the weight vector

Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# use 20 as the maximal number of group
hc <- hclust(dist(t(X)))
w <- computeGroupSizeWeight(hc, sizeMax = 20)
# Apply MLGL method
res <- MLGL(X, y, hc = hc, weightSizeGroup = w)
```

cv.MLGL

**cv.MLGL**

Multi-Layer Group-Lasso with cross V-fold validation

---

Description

V-fold cross validation for MLGL function

Usage

```r
cv.MLGL(
  X,
  y,
  nfolds = 5,
  lambda = NULL,
  hc = NULL,
  weightLevel = NULL,
  weightSizeGroup = NULL,
  loss = c("ls", "logit"),
  intercept = TRUE,
  sizeMaxGroup = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

- **X**: matrix of size n*p
- **y**: vector of size n. If loss = "logit", elements of y must be in -1,1
- **nfolds**: number of folds
cv.MLGL

lambda values for group lasso. If not provided, the function generates its own values of lambda.

hc output of hclust function. If not provided, hclust is run with ward.D2 method.

weightLevel a vector of size p for each level of the hierarchy. A zero indicates that the level will be ignored. If not provided, use 1/(height between 2 successive levels).

weightSizeGroup a vector

loss a character string specifying the loss function to use, valid options are: "ls" least squares loss (regression) and "logit" logistic loss (classification).

intercept should an intercept be included in the model?

sizeMaxGroup maximum size of selected groups. If NULL, no restriction

verbose print some informations

... Others parameters for cv.gglasso function

Details

Hierarchical clustering is performed with all the variables. Then, the partitions from the different levels of the hierarchy are used in the different run of MLGL for cross validation.

Value

a cv.MLGL object containing :

lambda values of lambda.

cvm the mean cross-validated error.

cvse estimate of standard error of cvm

cvupper upper curve = cvm+cvse

cvlower lower curve = cvm-cvse

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.

time computation time

Author(s)

Quentin Grimonprez

See Also

MLGL, stability.MLGL, predict.cv.gglasso, coef.cv.MLGL, plot.cv.MLGL
Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[, c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply cv.MLGL method
res <- cv.MLGL(X, y)
```

---

**Ftest**

**F-test**

Description

Perform a F-test

Usage

```r
Ftest(X, y, varToTest)
```

Arguments

- **X**: design matrix of size n*p
- **y**: response vector of length n
- **varToTest**: vector containing the index of the column of X to test

Details

\[ y = X \beta + \epsilon \]

null hypothesis : \( \beta[\text{varToTest}] = 0 \) alternative hypothesis : it exists an index \( k \) in \( \text{varToTest} \) such that \( \beta[k] \neq 0 \)

The test statistic is based on a full and a reduced model. full : \( y = X \beta[\text{varToTest}] + \epsilon \) reduced : the null model

Value

a vector of the same length as \( \text{varToTest} \) containing the p-values of the test.

See Also

`partialFtest`
Description

Run hierarchical clustering following by a group-lasso on all the different partition and a hierarchical testing procedure. Only for linear regression problem.

Usage

fullProcess(
  X,
  y,
  control = c("FWER", "FDR"),
  alpha = 0.05,
  test = partialFtest,
  hc = NULL,
  fractionSampleMLGL = 1/2,
  BHclust = 50,
  nCore = NULL,
  ...
)

fullProcess.formula(
  formula,
  data,
  control = c("FWER", "FDR"),
  alpha = 0.05,
  test = partialFtest,
  hc = NULL,
  fractionSampleMLGL = 1/2,
  ...
)

Arguments

X  matrix of size n*p
y  vector of size n.
control  either "FDR" or "FWER"
alpha  control level for testing procedure
test  test used in the testing procedure. Default is partialFtest
hc  output of hclust function. If not provided, hclust is run with ward.D2 method.
User can also provide the desired method: "single", "complete", "average", "mcquitty", "ward.D", "ward.D2", "centroid", "median".
fractionSampleMLGL

a real between 0 and 1: the fraction of individuals to use in the sample for MLGL (see Details).

BHclust

number of replicates for computing the distance matrix for the hierarchical clustering tree

nCore

number of cores used for distance computation. Use all cores by default.

... Others parameters for MLGL

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

an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula)

Details

Divide the n individuals in two samples. Then the three following steps are done: 1) Hierarchical CLustering of the variables of X based on the first sample of individuals 2) MLGL on the second sample of individuals 3) Hierarchical testing procedure on the first sample of individuals.

Value

a list containing:

res output of MLGL function

lambdaOpt lambda values maximizing the number of rejects

var A vector containing the index of selected variables for the first lambdaOpt value

group A vector containing the values index of selected groups for the first lambdaOpt value

selectedGroups Selected groups for the first lambdaOpt value

reject Selected groups for all lambda values

alpha Control level

test Test used in the testing procedure

control "FDR" or "FWER"

time Elapsed time

Author(s)

Quentin Grimonprez

See Also

MLGL, hierarchicalFDR, hierarchicalFWER, selFDR, selFWER
Examples

```r
# least square loss
set.seed(42)
X <- simuBlockGaussian(50, 12, 5, 0.7)
y <- X[,c(2,7,12)] %*% c(2,2,-2) + rnorm(50, 0, 0.5)
res <- fullProcess(X, y)
```

hierarchicalFDR

Hierarchical testing with FDR control

Description

Apply hierarchical test for each hierarchy, and test external variables for FDR control at level alpha.

Usage

```r
hierarchicalFDR(X, y, group, var, test = partialFtest)
```

Arguments

- `X`: original data
- `y`: associated response
- `group`: vector with index of groups. group[i] contains the index of the group of the variable var[i].
- `var`: vector with the variables contained in each group. group[i] contains the index of the group of the variable var[i].
- `test`: function for testing the nullity of a group of coefficients in linear regression. 3 parameters: X, response, and varToTest: vector of variables to test; return a pvalue

Details

Version of the hierarchical testing procedure of Yekutieli for MLGL output. You can use the selFDR function to select groups at a desired level alpha.

Value

A list containing:

- `pvalues`: pvalues of the different test (without correction)
- `adjPvalues`: adjusted pvalues
- `groupId`: Index of the group
- `hierMatrix`: Matrix describing the hierarchical tree.
References

See Also
selFDR, hierarchicalFWER

Examples
set.seed(42)
X = simuBlockGaussian(50,12,5,0.7)
y = X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
res = MLGL(X,y)
test = hierarchicalFDR(X, y, res$group[[20]], res$var[[20]])

hierarchicalFWER Hierarchical testing with FWER control

Description
Apply hierarchical test for each hierarchy, and test external variables for FWER control at level alpha

Usage
hierarchicalFWER(X, y, group, var, test = partialFtest, Shaffer = FALSE)

Arguments
X original data
y associated response
group vector with index of groups. group[i] contains the index of the group of the variable var[i].
var vector with the variables contained in each group. group[i] contains the index of the group of the variable var[i].
test function for testing the nullity of a group of coefficients in linear regression. 3 parameters : X : design matrix, y response and varToTest : vector of variables to test; return a pvalue
Shaffer boolean, if TRUE, a shaffer correction is performed

Details
Version of the hierarchical testing procedure of Meinshausen for MLGL output. You can use the selFWER function to select groups at a desired level alpha
**Value**

- **pvalues**  pvalues of the different test (without correction)
- **adjPvalues**  adjusted pvalues
- **groupId**  Index of the group
- **hierMatrix**  Matrix describing the hierarchical tree.

**References**


**See Also**

selFWER, hierarchicalFDR

**Examples**

```r
set.seed(42)
X = simuBlockGaussian(50,12,5,0.7)
y = X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
res = MLGL(X,y)
test = hierarchicalFWER(X, y, res$group[[20]], res$var[[20]])
```

---

**HMT**  
Hierarchical Multiple Testing procedure  
Apply Hierarchical Multiple Testing procedure on a `MLGL` object

**Description**

Hierarchical Multiple Testing procedure  
Apply Hierarchical Multiple Testing procedure on a `MLGL` object

**Usage**

```r
HMT(
  res,
  X,
  y,
  control = c("FWER", "FDR"),
  alpha = 0.05,
  test = partialFtest,
  ...
)
```
Arguments

- **res**: MLGL object
- **X**: matrix of size n*p
- **y**: vector of size n.
- **control**: either "FDR" or "FWER"
- **alpha**: control level for testing procedure
- **test**: test used in the testing procedure. Default is partialFtest
- **...**: extra parameters for selFDR

Value

a list containing:

- **lambdaOpt**: lambda values maximizing the number of rejects
- **var**: A vector containing the index of selected variables for the first lambdaOpt value
- **group**: A vector containing the values index of selected groups for the first lambdaOpt value
- **selectedGroups**: Selected groups for the first lambdaOpt value
- **reject**: Selected groups for all lambda values
- **alpha**: Control level
- **test**: Test used in the testing procedure
- **control**: "FDR" or "FWER"
- **time**: Elapsed time
- **hierTest**: list containing the output of the testing function for each lambda. Each element can be used with the selFWER or selFDR functions.
- **lambda**: lambda path
- **nGroup**: Number of groups before testing
- **nSelectedGroup**: Numer of groups after testing

See Also

hierarchicalFWER hierarchicalFDR selFWER selFDR

Examples

```r
set.seed(42)
X <- simuBlockGaussian(50, 12, 5, 0.7)
y <- X[,c(2,7,12)] %*% c(2,2,-2) + rnorm(50, 0, 0.5)
res <- MLGL(X, y)

# perform hierarchical testing with FWER control
out <- HMT(res, X, y, alpha = 0.05)

# test a new value of alpha for a specific lambda
selFWER(out$hierTest[[60]], alpha = 0.1)
```
listToMatrix

Obtain a sparse matrix of the coefficients of the path

Description

Obtain a sparse matrix of the coefficients of the path

Usage

listToMatrix(x, row = c("covariates", "lambda"))

Arguments

x MLGL object
row "lambda" or "covariates". If row="covariates", each row of the output matrix represents a covariates else if row="lambda", it represents a value of lambda.

Details

This functions can be used with a MLGL object to obtain a matrix with all estimated coefficients for the p original variables. In case of overlapping groups, coefficients from repeated variables are summed.

Value

a sparse matrix containing the estimated coefficients for different values of lambda

See Also

MLGL, overlapglasso

Examples

# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2, 7, 12)]%*%c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- MLGL(X, y)
# Convert output in sparse matrix format
beta <- listToMatrix(res)
**Multi-Layer Group-Lasso**

**Description**

Run hierarchical clustering following by a group-lasso on all the different partitions.

**Usage**

```r
MLGL(
  X,
  y,
  hc = NULL,
  lambda = NULL,
  weightLevel = NULL,
  weightSizeGroup = NULL,
  intercept = TRUE,
  loss = c("ls", "logit"),
  sizeMaxGroup = NULL,
  verbose = FALSE,
  ...
)

MLGL.formula(
  formula,
  data,
  hc = NULL,
  lambda = NULL,
  weightLevel = NULL,
  weightSizeGroup = NULL,
  intercept = TRUE,
  loss = c("ls", "logit"),
  verbose = FALSE,
  ...
)
```

**Arguments**

- **X**: matrix of size n*p
- **y**: vector of size n. If loss = "logit", elements of y must be in -1,1
- **hc**: output of `hclust` function. If not provided, `hclust` is run with `ward.D2` method. User can also provide the desired method: "single", "complete", "average", "mcquitty", "ward.D", "ward.D2", "centroid", "median".
- **lambda**: lambda values for group lasso. If not provided, the function generates its own values of lambda
weightLevel  a vector of size p for each level of the hierarchy. A zero indicates that the level will be ignored. If not provided, use 1/(height between 2 successive levels). Only if hc is provided

weightSizeGroup  a vector of size 2*p-1 containing the weight for each group. Default is the square root of the size of each group. Only if hc is provided

intercept  should an intercept be included in the model ?

loss  a character string specifying the loss function to use, valid options are: "ls" least squares loss (regression) and "logit" logistic loss (classification)

sizeMaxGroup  maximum size of selected groups. If NULL, no restriction

verbose  print some information

...  Others parameters for gglasso function

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  an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula)

Value

a MLGL object containing :

lambda  lambda values

b0  intercept values for lambda

beta  A list containing the values of estimated coefficients for each values of lambda

var  A list containing the index of selected variables for each values of lambda

group  A list containing the values index of selected groups for each values of lambda

nVar  A vector containing the number of non zero coefficients for each values of lambda

nGroup  A vector containing the number of non zero groups for each values of lambda

structure  A list containing 3 vectors. var : all variables used. group : associated groups. weight : weight associated with the different groups. level : for each group, the corresponding level of the hierarchy where it appears and disappears. 3 indicates the level with a partition of 3 groups.

time  computation time

dim  dimension of X

hc  Output of hierarchical clustering

call  Code executed by user

Author(s)

Quentin Grimonprez

See Also

cv.MLGL, stability.MLGL, listToMatrix, predict.MLGL, coef.MLGL, plot.cv.MLGL
Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[, c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- MLGL(X, y)
```

overlapgglasso  
Group-lasso with overlapping groups

Description

Group-lasso with overlapping groups

Usage

```r
overlapgglasso(
  X,
  y,
  var,
  group,
  lambda = NULL,
  weight = NULL,
  loss = c("ls", "logit"),
  intercept = TRUE,
  ...
)
```

Arguments

- **X**: matrix of size n*p
- **y**: vector of size n. If loss = "logit", elements of y must be in -1,1
- **var**: vector containing the variable to use
- **group**: vector containing the associated groups
- **lambda**: lambda values for group lasso. If not provided, the function generates its own values of lambda
- **weight**: a vector the weight for each group. Default is the square root of the size of each group
- **loss**: a character string specifying the loss function to use, valid options are: "ls" least squares loss (regression) and "logit" logistic loss (classification)
- **intercept**: should an intercept be included in the model?
- **...**: Others parameters for gglasso function
Details

Use a group-lasso algorithm (see \texttt{gglasso}) to solve a group-lasso with overlapping groups. Each variable \( j \) of the original matrix \( X \) is paste \( k(j) \) times in a new dataset with \( k(j) \) the number of different groups containing the variable \( j \). The new dataset is used to solve the group-lasso with overlapping groups running a group-lasso algorithm.

Value

a MLGL object containing :

- \( \texttt{lambda} \) lambda values
- \( \texttt{b0} \) intercept values for \( \texttt{lambda} \)
- \( \texttt{beta} \) A list containing the values of estimated coefficients for each values of \( \texttt{lambda} \)
- \( \texttt{var} \) A list containing the index of selected variables for each values of \( \texttt{lambda} \)
- \( \texttt{group} \) A list containing the values index of selected groups for each values of \( \texttt{lambda} \)
- \( \texttt{nVar} \) A vector containing the number of non zero coefficients for each values of \( \texttt{lambda} \)
- \( \texttt{nGroup} \) A vector containing the number of non zero groups for each values of \( \texttt{lambda} \)
- \( \texttt{structure} \) A list containing 3 vectors. \( \texttt{var} \) : all variables used. \( \texttt{group} \) : associated groups. \( \texttt{weight} \) : weight associated with the different groups.
- \( \texttt{time} \) computation time
- \( \texttt{dim} \) dimension of \( X \)

Source


See Also

\texttt{listToMatrix}

Examples

```r
# Least square loss
set.seed(42)
X <- simuBlockGaussian(50, 12, 5, 0.7)
y <- X[, c(2, 7, 12)]%*%c(2, 2, -2) + rnorm(50, 0, 0.5)
var <- c(1:60, 1:8, 7:15)
group <- c(rep(1:12, each = 5), rep(13, 8), rep(14, 9))
res <- overlapgglasso(X, y, var, group)

# Logistic loss
y <- 2*(rowSums(X[, 1:4])>0)-1
var <- c(1:60, 1:8, 7:15)
group <- c(rep(1:12, each = 5), rep(13, 8), rep(14, 9))
res <- overlapgglasso(X, y, var, group, loss = "logit")
```
partialFtest  Partial F-test

Description
Performs a partial F-test

Usage
partialFtest(X, y, varToTest)

Arguments
X  design matrix of size n*p
y  response vector of length n
varToTest  vector containing the index of the column of X to test

Details
y = X * beta + epsilon
null hypothesis : beta[varToTest] = 0 alternative hypothesis : it exists an index k in varToTest such that beta[k] != 0
The test statistic is based on a full and a reduced model. full : y = X * beta + epsilon reduced : y = X * beta[-varToTest] + epsilon

Value
a vector of the same length as varToTest containing the p-values of the test.

See Also
Ftest
plot.cv.MLGL  
Plot the cross-validation obtained from cv.MLGL function

Description
Plot the cross-validation obtained from cv.MLGL function

Usage
## S3 method for class 'cv.MLGL'
plot(x, log.lambda = FALSE, ...)

Arguments
x cv.MLGL object
log.lambda If TRUE, use log(lambda) instead of lambda in abscissa
... Other parameters for plot function

See Also
cv.MLGL

Examples
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
# Apply cv.MLGL method
res <- cv.MLGL(X,y)
# Plot the cv error curve
plot(res)

plot.fullProcess  
Plot the path obtained from fullProcess function

Description
Plot the path obtained from fullProcess function
Usage

```r
## S3 method for class 'fullProcess'
plot(
  x,
  log.lambda = FALSE,
  lambda.lines = FALSE,
  lambda.opt = c("min", "max", "both"),
  ...
)
```

Arguments

- `x`: `fullProcess` object
- `log.lambda`: If TRUE, use log(lambda) instead of lambda in abscissa
- `lambda.lines`: If TRUE, add vertical lines at lambda values
- `lambda.opt`: If there is several optimal lambdas, which one to print "min", "max" or "both"
- `...`: Other parameters for plot function

See Also

- `fullProcess`

Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2, 7, 12)]%*%c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- fullProcess(X, y)
# Plot the solution path
plot(res)
```

Description

Plot the path obtained from `HMT` function
Description

Plot the path obtained from \texttt{MLGL} function

Usage

\begin{verbatim}
## S3 method for class 'MLGL'
plot(x, log.lambda = FALSE, lambda.lines = FALSE, lambda.opt = c("min", "max", "both"), ...
)
\end{verbatim}
Arguments

- **x**: MLGL object
- **log.lambda**: If TRUE, use log(lambda) instead of lambda in abscissa
- **lambda.lines**: if TRUE, add vertical lines at lambda values
- **...**: Other parameters for plot function

See Also

MLGL

Examples

```r
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
set.seed(42)
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[, c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- MLGL(X, y)
# Plot the solution path
plot(res)
```
Value

A list containing:

- **var** Index of selected variables for the given threshold.
- **group** Index of the associated group.
- **threshold** Value of threshold

See Also

- stability.MLGL

Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)

# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50, 0, 0.5)

# Apply stability.MLGL method
res <- stability.MLGL(X, y)
selected <- plot(res)
print(selected)
```

---

**predict.cv.MLGL**

`Predict fitted values from a cv.MLGL object`

**Description**

Predict fitted values from a `cv.MLGL` object

**Usage**

```r
## S3 method for class 'cv.MLGL'
predict(
  object,
  newx = NULL,
  s = c("lambda.1se", "lambda.min"),
  type = c("fit", "coefficients"),
  ... )
```
Arguments

object: cv.MLGL object
newx: matrix with new individuals for prediction. If type="coefficients", the parameter has to be NULL
s: Either "lambda.1se" or "lambda.min"
type: if "fit", return the fitted values for each values of s, if "coefficients", return the estimated coefficients for each s
... Not used. Other arguments to predict.

Value

A matrix with fitted values or estimated coefficients for given values of s.

Author(s)

Quentin Grimonprez

See Also

cv.MLGL

Description

Predict fitted values from a MLGL object

Usage

## S3 method for class 'MLGL'
predict(object, newx = NULL, s = NULL, type = c("fit", "coefficients"), ...)

Arguments

object: MLGL object
newx: matrix with new individuals for prediction. If type="coefficients", the parameter has to be NULL
s: values of lambda. If NULL, use values from object
type: if "fit", return the fitted values for each values of s, if "coefficients", return the estimated coefficients for each s
... Not used. Other arguments to predict.

Value

A matrix with fitted values or estimated coefficients for given values of s.
Author(s)

original code from gglasso package Author: Yi Yang <yiyang@umn.edu>, Hui Zou <hzou@stat.umn.edu>
function inspired from predict function from gglasso package by Yi Yang and Hui Zou.

See Also

MLGL

Description

Print Values
Print a fullProcess object

Usage

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

Arguments

x fullProcess object
...
Not used.

See Also

fullProcess summary.fullProcess

Examples

set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
# Apply MLGL method
res <- fullProcess(X, y)
print(res)
print.hmt

Description
Print Values
Print a HMT object

Usage
## S3 method for class 'HMT'
print(x, ...)

Arguments
x HMT object
...
Not used.

See Also
HMT summary.HMT

Examples
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
# Apply MLGL method
res <- MLGL(X, y)
out <- HMT(res, X, y)
print(out)

print.mlgl

Description
Print Values
Print a MLGL object

Usage
## S3 method for class 'MLGL'
print(x, ...)


Arguments

- `x` : MLGL object
- ... : Not used.

See Also

MLGL summary.MLGL

Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[, c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- MLGL(X, y)
print(res)
```

Description

Select groups from hierarchical testing procedure with FDR control (hierarchicalFDR)

Usage

```r
selFDR(out, alpha = 0.05, global = TRUE, outer = TRUE)
```

Arguments

- `out` : output of hierarchicalFDR function
- `alpha` : control level for test
- `global` : if FALSE the provided alpha is the desired level control for each family.
- `outer` : if TRUE, the FDR is controlled only on outer node (rejected groups without rejected children). If FALSE, it is controlled on the full tree.

Details

See the reference for mode details about the method.

If each family is controlled at a level alpha, we have the following control: FDR control of full tree: \( \alpha \times \text{delta} \times 2 \) (\( \text{delta} = 1.44 \)) FDR control of outer node: \( \alpha \times L \times \text{delta} \times 2 \) (\( \text{delta} = 1.44 \))
Value

a list containing:

- `toSel` vector of boolean. TRUE if the group is selected
- `groupId` Names of groups
- `local.alpha` control level for each family of hypothesis
- `global.alpha` control level for the tree (full tree or outer node)

References


See Also

- `hierarchicalFDR`

Examples

```r
set.seed(42)
X = simuBlockGaussian(50,12,5,0.7)
y = X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
res = MLGL(X,y)
test = hierarchicalFDR(X, y, res$group[[20]], res$var[[20]])
sel = selFDR (test, alpha = 0.05)
```

Description

Select groups from hierarchical testing procedure with FWER control (`hierarchicalFWER`)

Usage

```r
selFWER(out, alpha = 0.05)
```

Arguments

- `out` output of `hierarchicalFDR` function
- `alpha` control level for test

Details

Only outer nodes (rejected groups without rejected children) are returned as TRUE.


Value

- `toSel` vector of boolean. TRUE if the group is selected
- `groupId` Names of groups

References


See Also

- `hierarchicalFWER`

Examples

```r
set.seed(42)
X = simuBlockGaussian(50,12,5,0.7)
y = X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
res = MLGL(X,y)
test = hierarchicalFWER(X, y, res$group[[20]], res$var[[20]])
sel = selFWER (test, alpha = 0.05)
```

---

**simuBlockGaussian**  
_Simulate multivariate Gaussian samples with block diagonal variance matrix_

**Description**

Simulate `n` samples from a gaussian multivariate law with 0 vector mean and block diagonal variance matrix with diagonal 1 and block of rho.

**Usage**

```r
simuBlockGaussian(n, nBlock, sizeBlock, rho)
```

**Arguments**

- `n` number of samples to simulate
- `nBlock` number of blocks
- `sizeBlock` size of blocks
- `rho` correlation within each block
stability.MLGL

Value

a matrix of size n *(nBlock*sizeBlock) containing the samples

Author(s)

Quentin Grimonprez

Examples

\[
X = \text{simuBlockGaussian}(50, 12, 5, 0.7)
\]

---

**stability.MLGL**  
*Stability Selection for Multi-Layer Group-lasso*

Description

Stability selection for **MLGL**

Usage

```r
stability.MLGL(
  X,
  y,
  B = 50,
  fraction = 0.5,
  hc = NULL,
  lambda = NULL,
  weightLevel = NULL,
  weightSizeGroup = NULL,
  loss = c("ls", "logit"),
  intercept = TRUE,
  verbose = FALSE,
  ...
)
```

Arguments

- **X**: matrix of size n*p
- **y**: vector of size n. If loss = "logit", elements of y must be in -1,1
- **B**: number of bootstrap sample
- **fraction**: Fraction of data used at each of the B sub-samples
- **hc**: output of `hclust` function. If not provided, `hclust` is run with ward.D2 method
- **lambda**: lambda values for group lasso. If not provided, the function generates its own values of lambda
- **weightLevel**: a vector of size p for each level of the hierarchy. A zero indicates that the level will be ignored. If not provided, use 1/(height between 2 successive levels)
weightSizeGroup

loss

intercept

verbose

... Others parameters for gglasso function

Details
Hierarchical clustering is performed with all the variables. Then, the partitions from the different levels of the hierarchy are used in the different run of MLGL for estimating the probability of selection of each group.

Value
a stability.MLGL object containing:

- lambda  sequence of lambda.
- B  Number of bootstrap samples.
- stability  A matrix of size length(lambda)*number of groups containing the probability of selection of each group
- var  vector containing the index of covariates
- group  vector containing the index of associated groups of covariates
- time  computation time

Author(s)
Quentin Grimonprez

References

See Also
cv.MLGL, MLGL

Examples

set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)

# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50, 0, 0.5)
# Apply stability.MLGL method
res <- stability.MLGL(X, y)

summary.fullProcess  

## S3 method for class 'fullProcess'
summary(object, ...)

### Arguments

- **object**  
  fullProcess object

- **...**  
  Not used.

### Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[, c(2, 7, 12)] %*% c(2, 2, -2) + rnorm(50, 0, 0.5)
# Apply MLGL method
res <- fullProcess(X, y)
summary(res)
```
Description
Object Summaries
Summary of a HMT object

Usage

```r
## S3 method for class 'HMT'
summary(object, ...)
```

Arguments

- **object**  
  HMT object

- **...**  
  Not used.

See Also

- `HMT`
- `print.HMT`

Examples

```r
set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
# Apply MLGL method
res <- MLGL(X, y)
out <- HMT(res, X, y)
summary(out)
```

Description
Object Summaries
Summary of a MLGL object

Usage

```r
## S3 method for class 'MLGL'
summary(object, ...)
```
Arguments

object MLGL object
... Not used.

See Also

MLGL print.MLGL

Examples

set.seed(42)
# Simulate gaussian data with block-diagonal variance matrix containing 12 blocks of size 5
X <- simuBlockGaussian(50, 12, 5, 0.7)
# Generate a response variable
y <- X[,c(2,7,12)]%*%c(2,2,-2) + rnorm(50,0,0.5)
# Apply MLGL method
res <- MLGL(X,y)
summary(res)

uniqueGroupHclust  Find all unique groups in hclust results

Description

Find all unique groups in hclust results

Usage

uniqueGroupHclust(hc)

Arguments

hc output of hclust function

Value

A list containing:

indexGroup Vector containing the index of variables.
varGroup Vector containing the index of the group of each variable.

Author(s)

Quentin Grimonprez
Examples

hc <- hclust(dist(USArrests), "ave")
res <- uniqueGroupHclust(hc)
Index

+Topic package
  MLGL-package, 2

bootstrapHclust, 3

test.cv.MLGL, 4, 7
test.MLGL, 5, 17
testGroupSizeWeight, 5
cv.gglasso, 7
cv.MLGL, 3, 4, 6, 17, 21, 25, 26, 33

Ftest, 8, 20
fullProcess, 3, 9, 21–23, 27, 34

glasso, 17–19, 33

test, 7, 9, 16, 32, 36
hierarchicalFDR, 10, 11, 13, 14, 29, 30
hierarchicalFWER, 3, 10, 12, 12, 14, 30, 31
HMT, 13, 22, 23, 28, 35

listToMatrix, 15, 17, 19

MLGL, 3, 5–7, 10, 13–15, 16, 23, 24, 26–29, 32,
33, 35, 36
MLGL-package, 2

overlapglasso, 15, 18

partialFtest, 8, 9, 14, 20
plot.cv.MLGL, 7, 17, 21
plot.fullProcess, 21
plot.HMT, 22
plot.MLGL, 23
plot.stability.MLGL, 24
predict.cv.gglasso, 7
predict.cv.MLGL, 4, 25
predict.MLGL, 5, 17, 26
print.fullProcess, 27, 34
print.HMT, 28, 35
print.MLGL, 28, 36

selFDR, 10–12, 14, 29
selFWER, 3, 10, 12–14, 30
simuBlockGaussian, 31
stability.MLGL, 7, 17, 24, 25, 32
summary.fullProcess, 27, 34
summary.HMT, 28, 35
summary.MLGL, 29, 35
uniqueGroupHclust, 36