Package ‘CorReg’

April 23, 2018

Type Package

Title Linear Regression Based on Linear Structure Between Variables

Version 1.2.8

Date 2018-04-23

Description Linear regression based on a recursive structural equation model
(explicit multiples correlations) found by a M.C.M.C. algorithm. It permits to face
highly correlated variables. Variable selection is included (by lasso,
elastic net, etc.). It also provides some graphical tools for basic
statistics.

License CeCILL

Copyright ArcelorMittal

URL http://www.correg.org

Depends R(>= 3.0)

Imports Rcpp(>= 0.11.0), lars(>= 1.2), Rmixmod(>= 2.0.1),
elasticnet(>= 1.1), corplot(>= 0.73), Matrix(>= 1.1), rpart(>= 4.1-5), glmnet(>= 2.0-2), MASS(>= 7.3-30), mvtnorm(>= 0.9),
mclust(>= 4.2), methods, graphics, grDevices, utils, stats

LinkingTo Rcpp, RcppEigen

Suggests clere(>= 1.1.2), spikeslab(>= 1.1.5), parcor(>= 0.2), missMDA
(>= 1.7.3), tuneR, knitr, rmarkdown

VignetteBuilder knitr

RoxygenNote 6.0.1

NeedsCompilation yes

Author Clement THERY [aut, cre],
Christophe BIERNACKI [ctb],
Gaetan LORIDANT [ctb],
Florian WATRIN [ctb],
Quentin GRIMONPREZ [ctb],
Vincent KUBICKI [ctb],
Samuel BLANCK [ctb],
Jeremie KELLNER [ctb]
**CorReg-package**

**Maintainer** Clement THERY <corregeous@correg.org>

**Repository** CRAN

**Date/Publication** 2018-04-23 18:09:06 UTC

---

### R topics documented:

- CorReg-package ........................................... 2
- BicZ .................................................. 4
- BoxPlot .................................................. 5
- cleanZ .................................................. 6
- cleanZtest ............................................... 7
- compare_struct .......................................... 7
- Conan ................................................... 8
- correg .................................................. 9
- CVMSE .................................................. 11
- density_estimation ....................................... 11
- Hist ..................................................... 13
- matplot_zone ............................................ 13
- mixture_generator ....................................... 15
- MSE_loc ................................................ 17
- naive_model ............................................. 18
- Numeric_Only .......................................... 18
- ProbaZ .................................................. 19
- purge_values ........................................... 19
- readZ ................................................... 20
- recursive_tree .......................................... 21
- report_MSE ............................................. 22
- showdata ............................................... 22
- structureFinder ......................................... 23
- Terminator .............................................. 26

---

## Index

<table>
<thead>
<tr>
<th>CorReg-package</th>
<th>Quick tutorial for CorReg package</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>28</td>
</tr>
</tbody>
</table>

---

### Description

Sequential linear regression based on a structural equation model (explicit correlations). It permits to face highly correlated datasets. We first search for an explicit model of correlations within the covariates by linear regression, then this structure is interpreted and used to reduce dimension and correlations for the main regression on the response variable.

### Details

CorReg: see www.correg.org for article and Phd Thesis about CorReg.
Author(s)
Maintainer: Clement THERY <clement.thery@arcelormittal.com>

References

Examples
```r
tr<-require(CorReg)
#dataset generation
base<mixture_generator(n=15,p=10, ratio=0.4, tp1=1, tp2=1, tp3=1, positive=0.5, R2y=0.8, R2x=0.9, scale=TRUE, max_comp=3, lambda=1)
X_appr<base$x_appr #learning sample
Y_appr<base$y_appr #response variable for the learning sample
Y_test<base$y_test #response variable for the validation sample
X_test<base$x_test #validation sample

truez<base$z #True generative structure (binary adjacency matrix)
#Z_{ij}=1 means that X_j linearly depends on X_i

density<density_estimation(X=X_appr, nbclustmax=10, detailed=TRUE)
bic_null_vect<density$bic_vect # vector of the bic found (1 value per covariate)

#MCMC to find the structure
res<structurefinder(X=X_appr, verbose=0, reject=0, Maxiter=900, nbini=20, candidates=-1, Bic_null_vect=Bic_null_vect, star=TRUE, p1max=15, clean=TRUE)
hatz<res$z_opt #found structure (adjacency matrix)
hatbic<res$bic_opt #associated BIC

bicopt_vect<Bic(Z=X_appr, z=hatz, Bic_null_vect=Bic_null_vect)
bicopt_vrai<Bic(Z=X_appr, z=truez, Bic_null_vect=Bic_null_vect)
sum(bicopt_vect);sum(bicopt_vrai)

#Structure comparison
comp<compare_struct(truez=truez, algo=hatz) #qualitative comparison

#interpretation of found and true structure ordered by increasing R2
readZ(z=hatz, crit="R2", order=1)
readZ(z=truez, crit="R2", order=1)

#Regression coefficients estimation
select=NULL #without variable selection (otherwise, choose "lar" for example)
resY<correg(X=X_appr, Y=Y_appr, z=hatz, compl=TRUE, expl=TRUE, pred=TRUE, select=select, K=10)

#MSE computation
```
MSE_complete=MSE_loc(Y=Y_test,X=X_test,A=resY$compl$A)#classical model on X
MSE_marginal=MSE_loc(Y=Y_test,X=X_test,A=resY$expl$A)#reduced model without correlations
MSE_plugin=MSE_loc(Y=Y_test,X=X_test,A=resY$pred$A)#plug-in model
MSE_true=MSE_loc(Y=Y_test,X=X_test,A=base$A)# True model

# MSE comparison
MSE=data.frame(MSE_complete,MSE_marginal,MSE_plugin,MSE_true)
MSE#estimated structure
compZ$true_left;compZ$false_left
barplot(as.matrix(MSE),main="MSE on validation dataset", sub=paste("select=" ,select))
abline(h=MSE_complete,col="red")

## End(Not run)

---

**BicZ**

*Compute the BIC of a given structure*

**Description**

Compute the BIC of a given structure

**Usage**

```r
BicZ(X = X, Z = Z, Bic_null_vect = NULL, Bic_old = NULL, methode = 1,
    Zold = NULL, star = FALSE)
```

**Arguments**

- `X` the dataset
- `Z` binary adjacency matrix of the structure (size p)
- `Bic_null_vect` the BIC of the null hypothesis (used for independent variables)
- `Bic_old` BIC (vector) associated to Zold
- `methode` parameter for OLS (matrix inversion) methode_BIC parameter for OLS (matrix inversion) 1:householderQr, 2:colPivHouseholderQr
- `Zold` another structure with some common parts with Z (allows to compute only the differences, to be faster)
- `star` boolean defining whether classical BIC or BIC* (over-penalized by a hierarchical uniform assumption to avoid over-learning) is computed

**Value**

The vector of the BICs associated to each covariate (conditional distribution) according to the sub-regression structure.
**Examples**

```r
## Not run:
require(CorReg)
data=mixture_generator(n=15,p=5,valid=0)#dataset generation
Z=data$Z #binary adjacency matrix that describes correlations within the dataset
X=data$x_appr
Bic_null_vect=density_estimation(X=X)$BIC_vect
#Computes the BIC associated to each covariate (optional, BicZ can do it if not given as an input)
#computes the BIC associated to the structure
res=BicZ(X = X,Z = Z,Bic_null_vect=Bic_null_vect)
```

## End(Not run)

---

**BoxPlot**

*Boxplot with confidence interval and ANOVA on the plot.*

---

**Description**

classical boxplot function improved with integrated confidence level on the mean for each group plotted on the graph and also ANOVA with p-value and its interpretation given in the legend.

**Usage**

```r
BoxPlot(num, grp = NULL, data = NULL, AnoVa = TRUE, risk = 0.05, 
lang = c("en", "fr"), verbose = TRUE, ...)
```

**Arguments**

- `num` a numeric vector to plot boxplot(num~grp). Represents the value that will be compared between the groups.
- `grp` a qualitative vector (factor) to plot boxplot(num~grp). Represents the groups we will compare.
- `data` a data.frame (or list) from which the variables in formula should be taken.
- `AnoVa` boolean to compute or not anova (when multiple groups) to see if they differ in mean. If false the Kruskal-Wallis Rank Sum test is computed instead.
- `risk` the risk value used for confidence intervals.
- `lang` linguistic parameter to specify the language of the legend
- `verbose` boolean to make a test and print the result in the subtitle
- `...` Other graphical parameters
cleanZ

### Not run:
```r
require(CorReg)
repart=c(20,40,40)
X=data.frame(num=c(rnorm(repart[1],10,1),rnorm(repart[2],11,1),rnorm(repart[3],10,1)),
grp=c(rep("A",times=repart[1]),rep("B",times=repart[2]),rep("C",times=repart[3])))
BoxPlot(X$num,X$grp,data=X,ylab="num",main="boxplot with confidence intervals")
#Confidence interval in red with mean in blue.
```

### End(Not run)

cleanZ

**clean the structure of correlations Z (if BIC improved)**

### Description

clean the structure of correlations Z (if BIC improved)

### Usage

```r
cleanZ(X = X, Z = Z, Bic_null_vect = Bic_null_vect, methode = 1,
       plot = F, verbose = 1, star = FALSE)
```

### Arguments

- **X**: the dataset
- **Z**: binary adjacency matrix of the structure (size p)
- **Bic_null_vect**: the BIC of the null hypothesis (used for independent variables)
- **methode**: parameter for OLS (matrix inversion) methode_BIC parameter for OLS (matrix inversion) 1:householderQr, 2:colPivHouseholderQr
- **plot**: if TRUE returns the vector of BIC for each step
- **verbose**: 0:none, 1:BIC,step and complexity when best BIC found 2:BIC, step, complexity, nb candidates and best candidate when best BIC found
- **star**: boolean defining wether classical BIC or BIC* is computed
cleanZtest

Clean Z’s columns based on p-values (coefficients or global)

Description
This function cleans the structure of correlations by setting to 0 the coefficients in the sub-regressions that are associated to a p-value below the "pvalmin" threshold.

Usage
```r
cleanZtest(Z = Z, X = X, pvalmin = 0.05, global = F, bonferroni = F)
```

Arguments
- `Z` the binary matrix describing the sub-regression structure (as given by the "structureFinder" function).
- `X` the dataset on which we have the sub-regression structure Z.
- `pvalmin` the threshold on coefficients p-values to clean the structure.
- `global` boolean. If TRUE the threshold is only on the F statistic for each sub-regression, not on each coefficients. So it will only remove entire sub-regressions.
- `bonferroni` boolean to use bonferroni correction on the pvalmin parameter to avoid multiple testing issues.

compare_struct

To compare sub-regression structures

Description
Compares two sub-regression structures, considering one of them as the "true one".

Usage
```r
compare_struct(trueZ = trueZ, Zalgo = Zalgo, all = TRUE, mode = "NULL")
```

Arguments
- `trueZ` first structure (binary adjacency matrix)
- `Zalgo` second structure (binary adjacency matrix)
- `all` (boolean) Also compute the ratio for each stat.
- `mode` how to modify the structures before comparison. mode=c("NULL","hybrid","clique","sym") It allows to compare groups instead of exact sub-regressions. Does nothing by default.
Conan

Value

true1  Number of links that exist in both matrices
false1 Number of links that exist only in Zalgo
false0 Number of links that exist only in trueZ
deltadr Number of sub-regressions in trueZ - Number of sub-regressions in Zalgo (i.e.: negative if too much sub-regressions in Zalgo)
true_left Number of variables redundant in both matrices
false_left Number of variables redundant in Zalgo but not in trueZ
ratio_true1 ratio of links in trueZ that exist also in Zalgo
ratio_true0 ratio of links not in trueZ that don’t exist in Zalgo.

Conan

Removes missing values (rows and column to obtain a large full matrix)

Description

Removes missing values (rows and column alternatively) to obtain a large full matrix

Usage

Conan(X = X, nbstep = Inf, std = FALSE, verbose = FALSE,
coercing = NULL, Xout = TRUE)

Arguments

X the dataset (matrix) with missing values
nbstep number of cutting steps (may remove several rows or columns at each step)
std (boolean) remove constant covariates
verbose (boolean) to print the result
coercing vector of the covariates to keep (names or index)
Xout (boolean) to export or not the reduced matrix (if not, indices are sufficient)

Value

individus_restants Index of remaining individuals
variables_restantes Index of remaining variables
X If Xout=TRUE, the reduced dataset without missing values
**correg**  

**Examples**

```r
## Not run:
data<-mtcars
require(CorReg)
datamiss=Terminator(target = data,wrath=0.05)#5% of missing values
datamiss
showdata(datamiss)#plot positions of the missing values
reduced=Conan(X=datamiss)
reduced

## End(Not run)
```

**correg**  

*Linear regression using CorReg’s method, with variable selection.*

**Description**

Computes three regression models: Complete (regression on the whole dataset X), marginal (regression using only independent covariates: $X[,\text{colSums}(Z)==0]$) and plug-in (sequential regression based on the marginal model and then use redundant covariates by plug-in, with a regression on the residuals of the marginal model by the residuals of the sub-regressions). Each regression can be computed with variable selection (for example the lasso).

**Usage**

```r
correg(X = NULL, y = NULL, Z = NULL, B = NULL, compl = TRUE,
expl = FALSE, pred = FALSE, select = "lar", criterion = c("MSE",
"BIC"), X_test = NULL, Y_test = NULL, intercept = TRUE, K = 10,
groupe = NULL, Amax = NULL, lambda = 1, alpha = NULL, g = 5)
```

**Arguments**

- **X**  
The data matrix (covariates) without the intercept
- **Y**  
The response variable vector
- **Z**  
The structure (adjacency matrix) between the covariates
- **B**  
The (d+1)x(d) matrix associated to Z and that contains the parameters of the sub-regressions
- **compl**  
(boolean) to decide if the complete model is computed
- **expl**  
(boolean) to decide if the explicative model is in the output
- **pred**  
(boolean) to decide if the predictive model is computed
- **select**  
selection method in ("lar","lasso","forward.stagewise","stepwise", "elasticnet", "NULL","ridge","adalasso","clere","spikeslab")
- **criterion**  
the criterion used to compare the models
- **X_test**  
validation sample
Y_test response for the validation sample
intercept boolean. If FALSE intercept will be set to 0 in each model.
K the number of clusters for cross-validation
groupe a vector of integer to define the groups used for cross-validation (to obtain a reproducible result)
Amax the maximum number of non-zero coefficients in the final model
lambda (optional) parameter for elasticnet or ridge (quadratic penalty) if select="elasticnet" or "ridge".
alpha Coefficients of the explicative model to coerce the predictive step. if not NULL explicative step is not computed.
g (optional) number of group of variables for clere if select="clere"

Value

a list that contains:
compl Results associated to the regression on X
expl Results associated to the marginal regression on explicative covariates (defined by colSums(Z)==0)
pred Results associated to the plug-in regression model.
compl$A Vector of the regression coefficients (the first is the intercept).(also have expl$A and pred$A)
compl$BIC BIC criterion associated to the model (also have expl$A and pred$A)
compl$AIC AIC criterion associated to the model (also have expl$A)
compl$CVMSE Cross-validated MSE associated to the model (also have expl$A)

Examples

## Not run:
require(CorReg)
#dataset generation
base=mixture_generator(n=15,p=10,ratio=0.4,tp1=1,tp2=1,tp3=1,positive=0.5, R2Y=0.8,R2Z=0.9, scale=TRUE,max_compl=3, lambda=1)
X_appr=base$X_appr #learning sample
Y_appr=base$Y_appr #response variable for the learning sample
Y_test=base$Y_test #response variable for the validation sample
X_test=base$X_test #validation sample
TrueZ=base$Z#True generative structure (binary adjacency matrix)

#Regression coefficients estimation
select="lar"#variable selection with lasso (using lar algorithm)
resY=correg(X=X_appr,Y=Y_appr,Z=base$Z,compl=TRUE,expl=TRUE,pred=TRUE,
           select=select,K=10)

#MSE computation
MSE_complete=MSE_loc(Y=Y_test,X=X_test,A=resY$compl$A)#classical model on X
MSE_marginal=MSE_loc(Y=Y_test,X=X_test,A=resY$expl$A)#reduced model without correlations
CVMSE

Cross validation

Description

Cross validation

Usage

\[
\text{CVMSE}(X = X, \ Y = Y, \ K = K, \ \text{intercept} = \text{TRUE}, \ \text{methode} = 1, \\
\text{groupe} = \text{NULL})
\]

Arguments

- **X**
  - covariates matrix (double)
- **Y**
  - response variable
- **K**
  - number of classes
- **intercept**
  - (boolean) with or without an intercept
- **methode**
  - the method used by OLS.
- **groupe**
  - a vector to define the groups used for cross-validation (to obtain a reproducible result)

Density_estimation

*BIC of estimated marginal gaussian mixture densities*

Description

Estimates the density of each covariates with gaussian mixture models and then gives the associated BIC.
density_estimation

Usage

density_estimation(X = X, nbclustmax = 10, nbclustmin = 1,
               verbose = FALSE, detailed = FALSE, max = TRUE, package = c("mclust",
               "Rmixmod"), nbini = 20, matshape = FALSE, ...)

Arguments

X the dataset (matrix)
nbclustmax max number of clusters in the gaussian mixtures
nbclustmin min number of clusters in the gaussian mixtures
verbose verbose or not
detailed boolean to give the details of the mixtures found
max boolean. Use an heuristic to shrink nbclustmax according to the number of
        individuals in the dataset
package package to use (Rmixmod,mclust)
nbini number of initial points for Rmixmod
matshape boolean to give the detail in matricial shape
... additional parameters

Value

a list that contains:

BIC_vect vector of the BIC (one per variable)
BIC global value of the BIC (=sum(BIC_vect))
nbclust vector of the numbers of components
details list of matrices that describe each Gaussian Mixture (proportions, means and
         variances)

Examples

## Not run:
rm(list=ls())#clean the workspace

require(CorReg)
#dataset generation
base=mixture_generator(n=150,p=10,valid=0, ratio=0.4, tp1=1, tp2=1, tp3=1, positive=0.5,
    R2Y=0.8,R2=0.9, scale=TRUE, max_compl=3, lambda=1)
X_appr=base$x_appr #learning sample
density=density_estimation(X = X_appr, detailed = TRUE)#estimation of the marginal densities
density$BIC_vect #vector of the BIC (one per variable)
density$BIC #global value of the BIC (sum of the BICs)
density$nbclust #vector of the numbers of components.
density$details #matrices that describe each Gaussian Mixture (proportions, means and variances)

## End(Not run)
Hist

Histos with clusters

Description

Histos with clusters

Usage

Hist(x, classes = NULL, plot = TRUE, col = 2:10, mode = c("classical", "cumsum", "density"), breaks = "Sturges", ...)

Arguments

x a vector, matrix or data.frame on which the histogram will be computed and plotted.
classes vector of classes to color
plot if FALSE, the histogram is only computed with no graphical output.
col numeric color
mode one of c("classical", "cumsum", "density")
breaks by default: "Sturges"
... further arguments and graphical parameters passed to plot.histogram and thence to title and axis.

Examples

## Not run:
require(CorReg)
x <- c(rnorm(50, 0, 1), rnorm(50, 1, 1))
classes <- rep(1:2, each = 50)
Hist(x, classes)

## End(Not run)

matplot_zone

Matplot with curves comparison by background colors.

Description

Plot the columns of one matrix against the columns of another, with conditionnal background for easier comparison of curves.
**Usage**

```r
matplot_zone(x = x, y = y, col = 1:6, alpha = 0.2, what = which.min,
    ylim = NULL, xlim = NULL, type = "p", xlab = NULL, ylab = NULL, ...)
```

**Arguments**

- `x`        the abscisses
- `y`        matrix of the curves (columns)
- `col`      list of colors (like in matplot)
- `alpha`    parameter for transparency of the background
- `what`     a function to choose a winner. Takes y as an input and must return a vector of colors (can be positive integers) of size length(x)
- `ylim`     ranges of y axe
- `xlim`     ranges of x axe
- `type`     character string (length 1 vector) or vector of 1-character strings indicating the type of plot for each column of y. The first character of type defines the first plot, the second character the second, etc. Characters in type are cycled through; e.g., "pl" alternately plots points and lines.
- `xlab`     title for x axe
- `ylab`     title for y axe
- `...`      Other graphical parameters

**Examples**

```r
## Not run:
require(CorReg)
n=15
x=1:n
y=cbind(c(rnorm(5,0,1),rnorm(5,1,1),rnorm(5,2,1)),
c(rnorm(5,0,1),rnorm(5,1,1),rnorm(5,4,1)),
c(rnorm(5,1,3),rnorm(5,1,2),rnorm(5,1,1)))
matplot_zone(x,y,type="1",what=which.max,main="Highest curve")
#background color follows color of the highest curve
matplot_zone(x,y,type="1",what=which.min,main="Lowest curve")
#background color follows color of the lowest curve

## End(Not run)
```
mixture_generator

Gaussian mixtures dataset generator with regression between the co-
variates

Description

Generates a dataset (with an additional validation sample) made of Gaussian mixtures with some of
them generated by sub-regressions on others. A response variable is then added by linear regression.
This function is used to generate datasets for simulations using CorReg, or just with Gaussian
Mixtures.

Usage

mixture_generator(n = 130, p = 100, ratio = 0.4, max_compl = 1,
valid = 1000, positive = 0.6, sigma_Y = 10, sigma_X = NULL,
R2 = NULL, R2Y = 0.4, meanvar = NULL, sigmavar = NULL, lambda = 3,
Amax = NULL, lambdaposis = 10, gamma = FALSE, gammadashape = 1,
gammadascaler = 0.5, tp1 = 1, tp2 = 1, tp3 = 1, nonlin = 0,
nonlin = 2, scale = TRUE, Z = NULL)

Arguments

n the number of individuals in the learning dataset
p the number of covariates (without the response)
ratio the ratio of covariates generated by sub-regressions on others
max_compl the number of covariates in each sub-regression
valid the number of individuals in the validation sample
positive the ratio of positive coefficients in both the regression and the sub-regressions
sigma_Y the standard deviation for the noise of the regression
sigma_X the standard deviation for the noise of the sub-regressions (all). ignored if
gamma=TRUE or if R2 is not NULL
R2 the strength of the sub-regressions (coefficients will be chosen to obtain this
value).
R2Y the strength of the main regression (coefficients will be chosen to obtain this
value).
meanvar vector of means for the covariates.
sigmavar standard deviation of the covariates.
lambda parameter of the Poisson’s law that defines the number of components in Gaussian
Mixtures models
Amax the maximum number of covariates with non-zero coefficients in the regression
lambdaposis parameter used to generate the coefficient in the subregressions. Poisson’s dis-
tribution.
gamma  (boolean) to generate a p-sized vector sigma_X gamma-distributed

gammashape  shape parameter of the gamma distribution (if needed)

gammascale  scale parameter of the gamma distribution (if needed)

tp1  the ratio of right-side (explicative) covariates allowed to have a non-zero coefficient in the regression

tp2  the ratio of left-side (redundant) covariates allowed to have a non-zero coefficient in the regression

tp3  the ratio of strictly independent covariates allowed to have a non-zero coefficient in the regression

nonlin  to use non linear structure (squared or log). If not null, it is the proba to use power pnonlin instead of log. The type is drawn for each link between covariates

pnonlin  the power used if non linear structure

scale  (boolean) to scale X before computing Y

Z  the binary squared adjacency matrix (size p) to obtain. If NULL it is randomly generated, based on ratio and max_compl parameters.

Value

a list that contains:

\(X_{\text{appr}}\)  matrix of the learning set. p covariates following Gaussian Mixtures with some of them generated by sub-regressions on others.

\(Y_{\text{appr}}\)  Response variable vector (size n) generated by linear regression on \(X_{\text{appr}}\) with coefficients \(A\) and residual standard deviation \(\sigma_Y\).

\(A\)  vector of the of the regression generating \(Y_{\text{appr}}\)

\(B\)  Matrix of the coefficients of sub-regressions (first line : the intercepts) then \(B[i-1,j]\) is the coefficient associated to \(X_{\text{appr}}[,i]\) in the sub-regression that generates \(X_{\text{appr}}[,j]\)

\(Z\)  Binary squared adjacency matrix of size p that describes the structure of sub-regressions. \(Z[i,j]=1\) if \(X_{\text{appr}}[,i]\) explains \(X_{\text{appr}}[,j]\)

\(X_{\text{test}}\)  validation sample generated the same way as \(X_{\text{appr}}\), with valid individuals.

\(Y_{\text{test}}\)  Response vector associated to the validation sample

\(\text{sigma}_X\)  Vector of the standard deviations of the residuals of the sub-regressions (one value for each sub-regression)

\(\text{sigma}_Y\)  Standard deviation of the residual of the regression that generates \(Y_{\text{appr}}\) and \(Y_{\text{test}}\).

\(\text{nbcomp}\)  vector of the number of components for covariates that are not explained by others.
**Examples**

```r
## Not run:
require(CorReg)
#dataset generation
base=mixture_generator(n=1500,p=10,valid=0)
X_appr=base$X_appr #learning sample
Y_appr=base$Y_appr #response variable
for(i in 1:ncol(X_appr)){
  hist(X_appr[,i])
}

## End(Not run)
```

---

### Description

This function computes the MSE (Mean Squared Error) of prediction associated to a vector of coefficients $A$ used to predict a response variable $Y$ by linear regression on $X$, with an intercept or not.

### Usage

```r
MSE_loc(Y = Y, X = X, A = A, intercept = T)
```

### Arguments

- **Y** the response variable (vector)
- **X** the dataset (matrix of covariates)
- **A** the vector of coefficients
- **intercept** (boolean) to add a column of 1 to $X$ if $A$ contains an intercept and $X$ doesn’t.

### Value

the Mean Squared Error observed on $X$ when using $A$ coefficients to predict $Y$. 

```r
@examples require(CorReg) #dataset generation base=mixture_generator(n=15,p=5,valid=100,_scale=TRUE)
X_appr=base$X_appr #learning sample
Y_appr=base$Y_appr #response variable
X_test=base$X_test #validation sample
Y_test=base$Y_test #response variable
A=lm(Y_appr~X_appr)$coefficients
MSE_loc(Y=Y_appr,X=X_appr,A=A)#MSE on the learning dataset
MSE_loc(Y=Y_test,X=X_test,A=A)#MSE on the validation sample
```
naive_model

How would it be if we were naive?

Description

Describe the result of a naive binary discriminant model

Usage

naive_model(proba_1 = 0.8, effectif = 100, nb_1 = NULL)

Arguments

proba_1 | The ratio of 1 in the population (if nb_1 is NULL)
effectif | The global effective of the population
nb_1 | The number of 1 in the population. If not NULL proba_1 is not read. default=NULL

Numeric_Only

To clean non numeric values in a vector

Description

Replace all non numeric values in a vector by NA’s and change the vector format to be numeric

Usage

Numeric_Only(x = NULL)

Arguments

x | The vector to clean
ProbaZ

Probability of Z without knowing the dataset. It also gives the exact number of binary nilpotent matrices of size p.

Description

Probability of Z without knowing the dataset. It also gives the exact number of binary nilpotent matrices of size p.

Usage

ProbaZ(Z = NULL, p = NULL, proba = FALSE, star = TRUE)

Arguments

- **Z**: binary adjacency matrix of the structure (size p)
- **p**: the number of covariates
- **proba**: gives the proba under the uniform law for Z, if FALSE and star=FALSE it gives the number of p-sized binary nilpotent matrices
- **star**: gives the log proba under uniform law for p

purge_values

Replaces unwanted values by NAs

Description

Find values in a dataframe and replace them by NAs. Also give the list of the variables implied. Beware of the factors. The variables stays as factors and the level is still in memory.

Usage

purge_values(base, value)

Arguments

- **base**: the dataframe to clean
- **value**: the value or vector of value to find and remove. if "space" it removes the blank thousands separator.
readZ

read the structure and explain it

Description

This function describes the structure of sub-regression given by an adjacency matrix. It computes the associated regression coefficients and R-squared for each sub-regression.

Usage

readZ(Z = Z, B = NULL, crit = c("none", "R2", "F", "sigmaX"),
    varnames = NULL, output = c("index", "names", "all"), X = NULL,
    order = 1)

Arguments

- **Z** binary adjacency matrix of the structure (size p)
- **B** is the complete structure (Z with sub-regression coefficients instead of 1 and an additional first line for the intercepts)
- **crit** define the criterion to use: c("none","R2","F","sigmaX")
- **varnames** the names of the variables (size p)
- **output** indicates the content of the output: c("index","names","all")
- **X** is a data frame or matrix containing the dataset
- **order** define the order used (0: none, -1: decreasing, 1: growing) for printing

Value

a list containing the sub-regressions details. Each item of the list represents a subregression. First element is the R-square. Second element is the variable that is regressed by others. Then comes the list of the explicative variables in the subgression and the associated coefficients (in the first column).

Examples

```r
## Not run:

data<-mtcars
#we first search a sub-regression structure
res=structureFinder(X = data,nbini = 30,verbose=0)
#then we can try to interpret it
readZ(Z = res$Z_opt,crit = "R2",output = "all",X = data)
#each component is a sub-regression
#First line : The adjusted R-squared is given
#Second line : the name of the covariate that is regressed by others
#other lines : Coefficients of sub-regression and name of the associated covariate

## End(Not run)
```
**Description**

decision tree in a recursive way

**Usage**

```r
recursive_tree(data = data, Y = "Y", modele = NULL, kill = NULL, index = NULL, print = TRUE, plot = TRUE, main = NULL, sub = NULL, lang = c("en", "fr"), all = FALSE, digits = getOption("digits") - 3)
```

**Arguments**

data: the dataset including the response  
Y: the name of the response  
modele: (optional) vector of names of covariates allowed in the tree  
kill: vector of the names to kill (variables won’t be used in the tree)  
index: to give a number to the plot  
print: boolean to print the tree parameters  
plot: boolean to plot the tree  
main: the main title if plot=TRUE  
sub: the subtitle (if NULL it is automatically added)  
lang: the language for the automatic subtitle in the plot  
all: Logical. If TRUE, all nodes are labeled, otherwise just terminal nodes.  
digits: number of digits for legend of the leaves

**Value**

returns the tree as an "rpart" object and the modele as a vector of the names of the covariates the tree could have used (to give as an input of the function).

modele: vector of the names of the covariates the tree could have used  
tree: the regression tree as an "rpart" object

**Examples**

```r
# Not run:  
data<-mtcars  
require(CorReg)  
main="Regression tree of cars consumption (in mpg)"  
mytree=recursive_tree(data = data,Y ="mpg",main=main)  
#want to try without cylinder and disp  
mytree2=recursive_tree(data = data,Y ="mpg",kill=c("cyl","disp"),modele=mytree$modele,main=main)
```
## report_MSE

Quickly reports some MSE

### Description
Some MSE reporting to have an overview of the predictive quality of a model.

### Usage
```
report_MSE(real, prediction)
```

### Arguments
- **real**: a numeric vector that contains the true values
- **prediction**: a numeric vector that contains the predicted values.

### Value
A list containing the MSE values. RMSE is the root MSE (square root of the mean square error). Relative is the mean of the relative errors (Root errors divided by the real values). Standard Deviation is the standard deviation of the root errors.

## showdata

To show the missing values of a dataset

### Description
Plot the dataset with marks where there are missing values. It allows to have a quick idea of the structure of missing values (Missing at Random or not for example).

### Usage
```
showdata(X = X, what = c("miss", "correl"), pch = 7)
```

### Arguments
- **X**: the matrix to analyse (matrix with missing values or correlations matrix)
- **what**: indicates what to plot. If what="correl" and X is a correlation matrix then the plot is a correlation plot. Else it shows the missing values positions in the dataset.
- **pch**: for missing, symbol to plot (can set pch="." for large datasets)
structureFinder

MCMC algorithm to find a structure between the covariates

Description

This function computes a random walk based on a full generative model on the dataset. We optimize a BIC-like criterion to find a model of sub-regressions within the covariates. If marginal density are unknown, Gaussian Mixture models are used automatically. We obtain the best structure as an adjacency matrix (binary squared matrix) that corresponds to the Directed Acyclic Graph of dependencies within the covariates.

Usage

structureFinder(X = X, Z = NULL, Bic_null_vect = NULL, candidates = -1, reject = 0, methode = 1, p1max = 5, p2max = NULL, Maxiter = 1, plot = FALSE, best = TRUE, better = FALSE, random = TRUE, verbose = 1, nb_opt_max = NULL, exact = TRUE, nbini = NULL, star = TRUE, clean = TRUE, ...)

Arguments

X  
the matrix of the dataset containing $p$ correlated covariates (with $n$ individuals)

Z  
(optional) initial structure. Binary adjacency matrix of size $p$. if NULL zero matrix is used

Bic_null_vect  
p-sized vector of the BIC values of the null hypothesis (used for independent variables). If NULL then it would be computed based on Gaussian Mixture hypothesis.

Examples

```r
## Not run:
data<-mtcars
require(CorReg)
datamiss=Terminator(target = data,wrath=0.05)#5% of missing values
showdata(datamiss)#plot positions of the missing values

#missing values with a structure
datamiss=Terminator(target = data,diag=1)#diag of missing values
showdata(datamiss)#plot positions of the missing values (no full individuals, no full variable)

opar=par(no.readonly = TRUE)
showdata(X=cor(data),what="correl")
par(opar)

## End(Not run)
```
candidates strategy to define a neighbourhood (list of candidates). Each new candidate is a modification of the current model as an adjacency matrix. So candidates are defined by the position that will be modified in Z. One modification for each candidate. We have then several neighbourhood to propose. 0: row and column (randomly chosen), -1: column only (randomly chosen), int>0: random int candidates at each step, -2: all (but the diagonal) so \( p^2 - p \) candidates, -3: non-zeros (at each step we test all possible link removal). Each strategy gives a distinct number of candidates at each step.

reject 0: constraint relaxation (if a candidate is not feasible then we modify it to make it feasible by deleting not compatible links), 1: reject mode (if a candidate is not feasible, we don’t look at it).

methode parameter for OLS (matrix inversion in Ordinary Least Squares) 1: householdQr, 2: colPivHouseholderQr

p1max maximum complexity (number of explaining covariates) for a sub-regression (positive integer)

p2max maximum number of sub-regressions (positive integer)

Maxiter number of steps (positive integer)

plot (boolean) TRUE: returns for each step the type of move, complexity and BIC. If nbini>1 then it returns the values associated to the chain that found the best BIC.

best (boolean) TRUE: systematically jumps to the best BIC seen ever when seen (it is stored even if best=FALSE)

better (boolean) TRUE: systematically jumps to the best candidate if better than stationarity (random jump weighted by the BIC otherwise)

random (boolean) if FALSE: moves only to improve and only to the best. Otherwise random jump weighted by the BIC if no deterministic jump due to parameters best and/or better.

verbose level of printed informations during the walk. 0: none, 1: BIC, step and complexity when best BIC found 2: BIC, step, complexity, nb candidates and best candidate when best BIC found.

nb_opt_max stop criterion defining how many times the chain can walk (or stay) on the max found

exact (boolean) If exact sub-regression is found it gives its content (another verbose mode). One of the covariates can then be deleted manually by the user without loss of information.

nbini Number of initialisations (using initialisation based on correlation matrix if Z is NULL). if NULL and Z is NULL: only one chain starting with zero matrix (model without any sub-regression)

star (boolean) to compute \( \text{BIC}^* \) instead of BIC (stronger penalization of the complexity based on a hierarchical uniform hypothesis on the probability of each structure). WARNING: star=TRUE implies \( p2max <= p/2 \).

clean (boolean) if TRUE then we add cleaning steps at the end of the walk (testing each remaining 1 for removal). So it is only few additional steps with candidates=-3

... optional parameters to be passed (for initialization).
Details

At each step we compare several candidates that are the local structure modified at one place (one coefficient of the adjacency matrix). Knowing the local structure a candidate is then just defined by the index of the position we modify in this local structure. So strategies are just choices of list of indices.

To avoid local extrema we allow constraints relaxation. If a modification is not feasible (it generates cycles for example) then the candidate is not rejected but modified. In fact, if we want to modify $Z[i,j]$ then we modify $Z$ at each position that makes the modification of $Z[i,j]$ not feasible. It is like allowing several steps in one, a kind of simulated anhealing but without parameter to tune.

Value

a list that contains:

- $Z$ The local structure of the last step (adjacency matrix)
- $Z_{opt}$ The best structure seen during the walk in terms of the BIC-like criterion.
- $bic_{opt}$ Value of the global BIC-like criterion associated to $Z_{opt}$
- $step_{opt}$ The index of the step where $Z_{opt}$ was found
- $bic_{null\_vect}$ $p$-sized vector of the BIC values associated to the model without sub-regressions. For use in a later search.
- $bic_{step}$ if plot=TRUE, vector of the BIC at each step
- $complexity_{step}$ if plot=TRUE, vector of the complexities at each step ($\sum_H z[I]$)
- $step$ if plot=TRUE, vector of the type of modification at each step. 0: delete, 1: add, 2: stationarity

Examples

```r
# Not run:
rm(list=ls()) # clean the workspace

require(CorReg)

# dataset generation
base=mixture_generator(n=15,p=10,ratio=0.4,tp1=1,tp2=1,tp3=1,positive=0.5,
R2Y=0.8,R2=0.9,scale=TRUE,max_compl=3,lambda=1)
X_appr=base$X_appr # learning sample
Y_appr=base$Y_appr # response variable for the learning sample
Y_test=base$Y_test # response variable for the validation sample
X_test=base$X_test # validation sample
TrueZ=base$Z # True generative structure (binary adjacency matrix)

density=density_estimation(X=X_appr,nbclustmax=10,detailed=TRUE)
Bic_null_vect=density$BIC_vect # vector of the BIC found (1 value per covariate)

# MCMC to find the structure
res=structureFinder(X=X_appr,verbose=0,reject=0,Maxiter=900,plot=TRUE,
```
Destructing values to have missing ones

**Description**

Destructing values to have missing ones

**Usage**

Terminator(target = NULL, wrath = 0.1, diag = 0, Z = NULL)

**Arguments**

- **target**: the dataset (matrix or data.frame) in which missing values will be made
- **wrath**: the ratio of missing values in the output
- **diag**: if = 1 it creates a diagonal band of missing values (no complete line, no complete column, but not too much missing values)
- **Z**: adjacency matrix to coerce a maximum of 1 missing value per sub-regression for each individual
### Value

the matrix with missing values.

### Examples

```r
## Not run:
rm(list=ls()) # clean the workspace

data<-mtcars
require(CorReg)
datamiss=Terminator(target = data, wrath=0.05) # 5% of missing values
showdata(datamiss) # plot positions of the missing values

datamiss=Terminator(target = data, diag=1) # diag of missing values
showdata(datamiss) # plot positions of the missing values (no full individuals, no full variable)

## End(Not run)
```
Index

*Topic package
   CorReg-package, 2

BicZ, 4
BoxPlot, 5

cleanZ, 6
cleanZtest, 7
cmpare_struct, 7
Conan, 8
correg, 9
CorReg-package, 2
CVMSE, 11

density_estimation, 11
Hist, 13

matplot_zone, 13
mixture_generator, 15
MSE_loc, 17

naive_model, 18
Numeric_Only, 18

ProbaZ, 19
purge_values, 19

readZ, 20
recursive_tree, 21
report_MSE, 22

showdata, 22
structureFinder, 23

Terminator, 26