Package ‘LorenzRegression’

February 28, 2023

Type Package

Title Lorenz and Penalized Lorenz Regressions

Version 1.0.0

Description
Inference for the Lorenz and penalized Lorenz regressions. More broadly, the package proposes functions to assess inequality and graphically represent it. The Lorenz Regression procedure is introduced in Heuchenne and Jacquemain (2022) <doi:10.1016/j.csda.2021.107347>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.3.1)

LazyData true

Imports stats, ggplot2, parallel, doParallel, foreach, MASS, GA, locpol, Rearrangement, Rcpp (>= 0.11.0), knitr

RoxygenNote 7.2.2

Suggests rmarkdown

LinkingTo Rcpp, RcppArmadillo

NeedsCompilation yes

Author Alexandre Jacquemain [aut, cre] (<`https://orcid.org/0000-0001-9349-780X`>), Xingjie Shi [ctb] (Author of an R implementation of the FABS algorithm available at https://github.com/shuanggema/Fabs, of which function Lorenz.FABS is derived)

Maintainer Alexandre Jacquemain <`aljacquemain@gmail.com`>

Repository CRAN

Date/Publication 2023-02-28 17:32:34 UTC

R topics documented:

.Fitness_cpp 2
boot.confint 3

coef.LR 3
coef.PLR 4
confint.LR 5
confint.PLR 6
Data.Incomes 7
Gini.coef 8
Lorenz.boot 9
Lorenz.curve 11
Lorenz.FABS 12
Lorenz.GA 14
Lorenz.graphs 16
Lorenz.Population 17
Lorenz.Reg 17
Lorenz.SCADFABS 21
LorenzRegression 23
plot.LR 23
plot.PLR 24
PLR.BIC 25
PLR.CV 26
PLR.normalize 28
PLR.wrap 28
print.LR 30
print.PLR 31
Rearrangement.estimation 31
summary.LR 33
summary.PLR 33

Index **35**

`.Fitness_cpp` *Computes the fitness used in the GA*

Description

Computes the fitness of a candidate in the genetic algorithm displayed in function Lorenz.GA.cpp

Usage

```
.Fitness_cpp(x, Y, X, Z, pi)
```

Arguments

`x`	vector of size (p-1) giving the proposed candidate, where p is the number of covariates
`Y`	vector of size n gathering the response, where n is the sample size
`X`	matrix of dimension (n*p) gathering the covariates
`Z`	vector of size n gathering iid repetitions of a U[0,1]
`pi`	vector of size n gathering the observation weights (notice that sum(pi)=1)

Value

Fitness of candidate x

`boot.confint` *Bootstrap confidence intervals*

Description

`boot.confint` computes bootstrap confidence intervals given an estimation on the original sample and on the bootstrap samples

Usage

```
boot.confint(x.hat, x.star, alpha, boot.method)
```

Arguments

`x.hat`	estimator on the original sample.
`x.star`	vector gathering the estimation on the bootstrapped sample.
`alpha`	1-level of the confidence interval
`boot.method`	bootstrap method.

Value

A vector of dimension two with the desired confidence interval

---

`coef.LR` *Estimated coefficients for the Lorenz Regression*

Description

`coef.LR` provides the estimated coefficients for an object of class `LR`.

Usage

```
## S3 method for class 'LR'
coef(object, ...)
```

Arguments

`object`	Output of a call to `Lorenz.Reg`, where `penalty="none"`.
`...`	Additional arguments.

Value

a vector gathering the estimated coefficients

See Also

Lorenz.Reg

Examples

```
data(Data.Incomes)
NPLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "none")
coef(NPLR)
```

---

coef.PLR *Estimated coefficients for the Penalized Lorenz Regression*

Description

`coef.PLR` provides the estimated coefficients for an object of class `PLR`.

Usage

```
## S3 method for class 'PLR'
coef(object, renormalize = TRUE, ...)
```

Arguments

`object`	Output of a call to `Lorenz.Reg`, where `penalty!="none"`.
`renormalize`	whether the coefficient vector should be re-normalized to match the representation where the first category of each categorical variable is omitted. Default value is TRUE
`...`	Additional arguments

Value

If the PLR was fitted with only one selection method, the output is a vector gathering the estimated coefficients. If several selection methods were selected, it outputs a list of vectors, where each element of the list corresponds to a different selection method.

See Also

Lorenz.Reg

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  h.grid = nrow(Data.Incomes)^(-1/5.5), sel.choice = c("BIC","CV"),
                  eps = 0.01, seed.CV = 123, nfolds = 5)
coef(PLR)
```

---

confint.LR *Confidence intervals for the Lorenz Regression*

Description

confint.LR provides confidence intervals for the explained Gini coefficient, Lorenz-R2 and theta vector for an object of class LR.

Usage

```
## S3 method for class 'LR'
confint(
  object,
  parm = c("Gini", "LR2", "theta"),
  level = 0.95,
  boot.method = c("Param", "Basic", "Perc"),
  ...
)
```

Arguments

object	Output of a call to Lorenz.Reg, where penalty="none" and Boot.inference=TRUE.
parm	Determines whether the confidence interval is computed for the explained Gini coefficient, for the Lorenz-R2 or for the vector of theta coefficients. Possible values are "Gini" (default, for the explained Gini),"LR2" (for the Lorenz-R2) and "theta" (for the vector theta).
level	level of the confidence interval
boot.method	What bootstrap method is used to construct the confidence interval. Default value is "Param", which exploits the asymptotic normality and only bootstraps the variance. Other possible values are "Perc" (percentile bootstrap) and "Basic" (basic bootstrap). Percentile bootstrap directly plugs the quantiles of the bootstrap distribution. Basic bootstrap is based on bootstrapping the whole distribution of the estimator.
...	Additional arguments.

Details

Use this function only if Boot.inference was set to TRUE in the call to Lorenz.Reg. Otherwise, bootstrap was not computed and the confidence intervals cannot be determined.

Value

The desired confidence interval. If parm is set to either "Gini" or "LR2", the output is a vector. If parm is set to "theta", it is a matrix where each row corresponds to a different coefficient.

See Also

`Lorenz.Reg`

Examples

```
# The following piece of code might take several minutes
data(Data.Incomes)
set.seed(123)
Data <- Data.Incomes[sample(1:nrow(Data.Incomes),50),]
NPLR <- Lorenz.Reg(Income ~ ., data = Data, penalty = "none",
                   seed.boot = 123, B = 40, Boot.inference = TRUE)
confint(NPLR)
```

`confint.PLR` *Confidence intervals for the Penalized Lorenz Regression*

Description

`confint.PLR` provides confidence intervals for the explained Gini coefficient and Lorenz-R2 for an parm of class PLR.

Usage

```
## S3 method for class 'PLR'
confint(
  object,
  parm = c("Gini", "LR2"),
  level = 0.95,
  boot.method = c("Param", "Basic", "Perc"),
  which.pars = NULL,
  ...
)
```

Arguments

`object`	Output of a call to `Lorenz.Reg`, where `penalty!="none"` and `Boot.inference=TRUE`.
`parm`	Determines whether the confidence interval is computed for the explained Gini coefficient or for the Lorenz-R2. Possible values are "Gini" (default, for the explained Gini) and "LR2" (for the Lorenz-R2).

`level`	level of the confidence interval
`boot.method`	What bootstrap method is used to construct the confidence interval. Default value is "Param", which exploits the asymptotic normality and only bootstraps the variance. Other possible values are "Perc" (percentile bootstrap) and "Basic" (basic bootstrap). Percentile bootstrap directly plugs the quantiles of the bootstrap distribution. Basic bootstrap is based on bootstrapping the whole distribution of the estimator.
`which.pars`	Which values of the bandwidth h and the penalty parameter lambda should be used. Default is NULL, in which case the optimal values are used.
`...`	Additional arguments.

Details

Use this function only if Boot.inference was set to TRUE in the call to `Lorenz.Reg`. Otherwise, bootstrap was not computed and the confidence intervals cannot be determined.

Value

A matrix gathering the desired confidence intervals. Each row corresponds to a different selection method for the pair (h,lambda).

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
set.seed(123)
Data <- Data.Incomes[sample(1:nrow(Data.Incomes),50),]
PLR <- Lorenz.Reg(Income ~ ., data = Data, h.grid = nrow(Data)^(-1/5.5),
            penalty = "SCAD", eps = 0.02, seed.boot = 123, B = 40, Boot.inference = TRUE)
confint(PLR)
```

`Data.Incomes` *Simulated income data*

Description

Fictitious cross-sectional dataset used to illustrate the Lorenz regression methodology. It covers 7 variables for 200 individuals aged between 25 and 30 years.

Usage

```
data(Data.Incomes)
```

Format

A data frame with 200 rows and 7 columns:

Income Individual’s labor income

Sex Sex (0=Female, 1=Male)

Health.level Variable ranging from 0 to 10 indicating the individual health’s level (0 is worst, 10 is best)

Age Individual’s age in years, ranging from 25 to 30

Work.Hours Individual’s weekly work hours

Education Individual’s highest grade completed in years

Seniority Length of service in years with the individual’s employer

---

`Gini.coef` *Concentration index of* y *wrt* x

Description

`Gini.coef` computes the concentration index of a vector *y* with respect to another vector *x*. If *y* and *x* are identical, the obtained concentration index boils down to the Gini coefficient.

Usage

```
Gini.coef(
  y,
  x = y,
  na.rm = TRUE,
  ties.method = c("mean", "random"),
  seed = NULL,
  weights = NULL
)
```

Arguments

`y`	variable of interest.
`x`	variable to use for the ranking. By default $x = y$, and the obtained concentration index is the Gini coefficient of *y*.
`na.rm`	should missing values be deleted. Default value is `TRUE`. If `FALSE` is selected, missing values generate an error message
`ties.method`	What method should be used to break the ties in the rank index. Possible values are "mean" (default value) or "random". If "random" is selected, the ties are broken by further ranking in terms of a uniformly distributed random variable. If "mean" is selected, the average rank method is used.
`seed`	fixes what seed is imposed for the generation of the vector of uniform random variables used to break the ties. Default is NULL, in which case no seed is imposed.
`weights`	vector of sample weights. By default, each observation is given the same weight.

Value

The value of the concentration index (or Gini coefficient)

See Also

`Lorenz.curve`, `Lorenz.graphs`

Examples

```
data(Data.Incomes)
# We first compute the Gini coefficient of Income
Y <- Data.Incomes$Income
Gini.coef(y = Y)
# Then we compute the concentration index of Income with respect to Age
X <- Data.Incomes$Age
Gini.coef(y = Y, x = X)
```

`Lorenz.boot` *Produces bootstrap-based inference for (penalized) Lorenz regression*

Description

`Lorenz.boot` determines bootstrap estimators for the weight vector, explained Gini coefficient and Lorenz-R^2 and, if applies, selects the regularization parameter.

Usage

```
Lorenz.boot(
  formula,
  data,
  standardize = TRUE,
  weights = NULL,
  LR.est = NULL,
  penalty = c("none", "SCAD", "LASSO"),
  h = NULL,
  eps = 0.005,
  B = 500,
  bootID = NULL,
  seed.boot = NULL,
  parallel = FALSE,
  ...
)
```

Arguments

`formula`	A formula object of the form *response* ~ *other_variables*.
`data`	A data frame containing the variables displayed in the formula.
`standardize`	Should the variables be standardized before the estimation process? Default value is TRUE.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`LR.est`	Estimation on the original sample. Output of a call to `Lorenz.GA` or `PLR.wrap`.
`penalty`	should the regression include a penalty on the coefficients size. If "none" is chosen, a non-penalized Lorenz regression is computed using function `Lorenz.GA`. If "SCAD" is chosen, a penalized Lorenz regression with SCAD penalty is computed using function `Lorenz.SCADFABS`. IF "LASSO" is chosen, a penalized Lorenz regression with LASSO penalty is computed using function `Lorenz.FABS`.
`h`	Only used if penalty="SCAD" or penalty="LASSO". Bandwidth of the kernel, determining the smoothness of the approximation of the indicator function. Default value is NULL (unpenalized case) but has to be specified if penalty="LASSO" or penalty="SCAD".
`eps`	Only used if penalty="SCAD" or penalty="LASSO". Step size in the FABS or SCADFABS algorithm. Default value is 0.005.
`B`	Number of bootstrap resamples. Default is 500.
`bootID`	matrix where each row provides the ID of the observations selected in each bootstrap resample. Default is NULL, in which case these are defined internally.
`seed.boot`	Should a specific seed be used in the definition of the folds. Default value is NULL in which case no seed is imposed.
`parallel`	Whether parallel computing should be used to distribute the `B` computations on different CPUs. Either a logical value determining whether parallel computing is used (TRUE) or not (FALSE, the default value). Or a numerical value determining the number of cores to use.
`...`	Additional parameters corresponding to arguments passed in `Lorenz.GA`, `Lorenz.SCADFABS` or `Lorenz.FABS` depending on the argument chosen in penalty.

Value

A list with several components:

`LR.est` Estimation on the original sample.

`Gi.star` In the unpenalized case, a vector gathering the bootstrap estimators of the explained Gini coefficient. In the penalized case, it becomes a list of vectors. Each element of the list corresponds to a different value of the penalization parameter

`LR2.star` In the unpenalized case, a vector gathering the bootstrap estimators of the Lorenz-R^2. In the penalized case, it becomes a list of vectors.

`theta.star` In the unpenalized case, a matrix gathering the bootstrap estimators of theta (rows correspond to bootstrap iterations and columns refer to the different coefficients). In the penalized case, it becomes a list of matrices.

`OOB.total` In the penalized case only. Vector gathering the OOB-score for each lambda value.

`OOB.best` In the penalized case only. index of the lambda value attaining the highest OOB-score.

References

Heuchenne, C. and A. Jacquemain (2022). Inference for monotone single-index conditional means: A Lorenz regression approach. *Computational Statistics & Data Analysis 167(C)*. Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient.

See Also

`Lorenz.Reg`, `Lorenz.GA`, `Lorenz.SCADFABS`, `Lorenz.FABS`, `PLR.wrap`

Examples

```
data(Data.Incomes)
set.seed(123)
Data <- Data.Incomes[sample(1:nrow(Data.Incomes),50),]
Lorenz.boot(Income ~ ., data = Data,
            penalty = "SCAD", h = nrow(Data)^(-1/5.5),
            eps = 0.02, B = 40, seed.boot = 123)
```

`Lorenz.curve` *Concentration curve of* y *with respect to* x

Description

`Lorenz.curve` computes the concentration curve index of a vector y with respect to another vector x. If y and x are identical, the obtained concentration curve boils down to the Lorenz curve.

Usage

```
Lorenz.curve(
  y,
  x = y,
  graph = FALSE,
  na.rm = TRUE,
  ties.method = c("mean", "random"),
  seed = NULL,
  weights = NULL
)
```

Arguments

`y`	variable of interest.
`x`	variable to use for the ranking. By default $x = y$, and the obtained concentration curve is the Lorenz curve of y.

`graph`	whether a graph of the obtained concentration curve should be traced. Default value is FALSE.
`na.rm`	should missing values be deleted. Default value is `TRUE`. If `FALSE` is selected, missing values generate an error message
`ties.method`	What method should be used to break the ties in the rank index. Possible values are "mean" (default value) or "random". If "random" is selected, the ties are broken by further ranking in terms of a uniformly distributed random variable. If "mean" is selected, the average rank method is used.
`seed`	seed imposed for the generation of the vector of uniform random variables used to break the ties. Default is NULL, in which case no seed is imposed.
`weights`	vector of sample weights. By default, each observation is given the same weight.

Value

A function corresponding to the estimated Lorenz or concentration curve. If `graph` is TRUE, the curve is also plotted.

See Also

`Lorenz.graphs`, `Gini.coef`

Examples

```
data(Data.Incomes)
# We first compute the Lorenz curve of Income
Y <- Data.Incomes$Income
Lorenz.curve(y = Y, graph = TRUE)
# Then we compute the concentration curve of Income with respect to Age
X <- Data.Incomes$Age
Lorenz.curve(y = Y, x = X, graph = TRUE)
```

`Lorenz.FABS` *Solves the Penalized Lorenz Regression with Lasso penalty*

Description

`Lorenz.FABS` solves the penalized Lorenz regression with (adaptive) Lasso penalty on a grid of lambda values. For each value of lambda, the function returns estimates for the vector of parameters and for the estimated explained Gini coefficient, as well as the Lorenz-R^2 of the regression.

Usage

```
Lorenz.FABS(
  YX_mat,
  weights = NULL,
  h,
  w.adaptive = NULL,
  eps,
  iter = 10^4,
  lambda = "Shi",
  lambda.min = 1e-07,
  gamma = 0.05
)
```

Arguments

`YX_mat`	a matrix with the first column corresponding to the response vector, the remaining ones being the explanatory variables.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`h`	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function.
`w.adaptive`	vector of size equal to the number of covariates where each entry indicates the weight in the adaptive Lasso. By default, each covariate is given the same weight (Lasso).
`eps`	step size in the FABS algorithm.
`iter`	maximum number of iterations. Default value is 10^4.
`lambda`	this parameter relates to the regularization parameter. Several options are available. `grid` If lambda="grid", lambda is defined on a grid, equidistant in the logarithmic scale. `Shi` If lambda="Shi", lambda, is defined within the algorithm, as in Shi et al (2018). `supplied` If the user wants to supply the lambda vector himself
`lambda.min`	lower bound of the penalty parameter. Only used if lambda="Shi".
`gamma`	value of the Lagrange multiplier in the loss function

Details

The regression is solved using the FABS algorithm developed by Shi et al (2018) and adapted to our case. For a comprehensive explanation of the Penalized Lorenz Regression, see Jacquemain et al. In order to ensure identifiability, theta is forced to have a L2-norm equal to one.

Value

A list with several components:

`iter` number of iterations attained by the algorithm.

`direction` vector providing the direction (-1 = backward step, 1 = forward step) for each iteration.

`lambda` value of the regularization parameter for each iteration.

`h` value of the bandwidth.

`theta` matrix where column i provides the estimated parameter vector for iteration i.

`LR2` the Lorenz-R^2 of the regression.

`Gi.expl` the estimated explained Gini coefficient.

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient. Shi, X., Y. Huang, J. Huang, and S. Ma (2018). A Forward and Backward Stagewise Algorithm for Nonconvex Loss Function with Adaptive Lasso, *Computational Statistics & Data Analysis 124*, 235-251.

See Also

`Lorenz.Reg`, `PLR.wrap`, `Lorenz.SCADFABS`

Examples

```
data(Data.Incomes)
YX_mat <- Data.Incomes[,-2]
Lorenz.FABS(YX_mat, h = nrow(Data.Incomes)^(-1/5.5), eps = 0.005)
```

`Lorenz.GA` *Estimates the parameter vector in Lorenz regression using a genetic algorithm*

Description

`Lorenz.GA` estimates the vector of parameters in Lorenz regression using the unit-norm normalization It also returns the Lorenz-R^2 of the regression as well as the estimated explained Gini coefficient.

Usage

```
Lorenz.GA(
  YX_mat,
  standardize = TRUE,
  popSize = 50,
  maxiter = 1500,
  run = 150,
  ties.method = c("random", "mean"),
  ties.Gini = c("random", "mean"),
  seed.random = NULL,
  weights = NULL,
  parallel = FALSE
)
```

Arguments

`YX_mat`	A matrix with the first column corresponding to the response vector, the remaining ones being the explanatory variables.
`standardize`	Should the variables be standardized before the estimation process? Default value is TRUE.
`popSize`	Size of the population of candidates in the genetic algorithm. Default value is 50.
`maxiter`	Maximum number ot iterations in the genetic algorithm. Default value is 1500.
`run`	Number of iterations without improvement in the best fitness necessary for the algorithm to stop. Default value is 150.
`ties.method`	What method should be used to break the ties in optimization program. Possible values are "random" (default value) or "mean". If "random" is selected, the ties are broken by further ranking in terms of a uniformly distributed random variable. If "mean" is selected, the average rank method is used.
`ties.Gini`	what method should be used to break the ties in the computation of the Gini coefficient at the end of the algorithm. Possible values and default choice are the same as above.
`seed.random`	seed.random imposed for the generation of the vector of uniform random variables used to break the ties. Default is NULL, in which case no seed.random is imposed.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`parallel`	Whether parallel computing should be used to distribute the computations in the genetic algorithm. Either a logical value determining whether parallel computing is used (TRUE) or not (FALSE, the default value). Or a numerical value determining the number of cores to use.

Details

The genetic algorithm is solved using function `ga` from the *GA* package. The fitness function is coded in Rcpp to speed up computation time. When discrete covariates are introduced and ties occur in the index, the default option randomly breaks them, as advised in Section 3 of Heuchenne and Jacquemain (2020)

Value

A list with several components:

`theta` the estimated vector of parameters.

`LR2` the Lorenz-R^2 of the regression.

`Gi.expl` the estimated explained Gini coefficient.

`niter` number of iterations attained by the genetic algorithm.

`fit` value attained by the fitness function at the optimum.

References

Heuchenne, C. and A. Jacquemain (2022). Inference for monotone single-index conditional means: A Lorenz regression approach. *Computational Statistics & Data Analysis 167(C).*

See Also

`Lorenz.Reg`, `ga`

Examples

```
data(Data.Incomes)
YX_mat <- cbind(Data.Incomes$Income, Data.Incomes$Age, Data.Incomes$Work.Hours)
Lorenz.GA(YX_mat, popSize = 40)
```

`Lorenz.graphs` *Graphs of concentration curves*

Description

`Lorenz.graphs` traces the Lorenz curve of a response and the concentration curve of the response and each of a series of covariates.

Usage

```
Lorenz.graphs(formula, data, ...)
```

Arguments

`formula`	A formula object of the form *response* ~ *other_variables*.
`data`	A dataframe containing the variables of interest
`...`	other arguments (see Section 'Arguments' in `Lorenz.curve`).

Value

A plot comprising

- The Lorenz curve of *response*
- The concentration curves of *response* with respect to each element of *other_variables*

See Also

`Lorenz.curve`, `Gini.coef`

Examples

```
data(Data.Incomes)
Lorenz.graphs(Income ~ Age + Work.Hours, data = Data.Incomes)
```

`Lorenz.Population` *Defines the population used in the genetic algorithm*

Description

`Lorenz.Population` creates the initial population of the genetic algorithm used to solve the Lorenz regression.

Usage

```
Lorenz.Population(object)
```

Arguments

`object`	An object of class "`ga`", resulting from a call to function `ga`.

Details

Note that this population produces an initial solution ensuring a unit norm.

Value

A matrix of dimension `object@popSize` times the number of explanatory variables minus one, gathering the initial population.

See Also

`Lorenz.GA`

`Lorenz.Reg` *Undertakes a Lorenz regression*

Description

`Lorenz.Reg` performs the Lorenz regression of a response with respect to several covariates.

Usage

```
Lorenz.Reg(
  formula,
  data,
  standardize = TRUE,
  weights = NULL,
  parallel = FALSE,
  penalty = c("none", "SCAD", "LASSO"),
  h.grid = c(0.1, 0.2, 1, 2, 5) * nrow(data)^(-1/5.5),
  eps = 0.005,
  sel.choice = c("BIC", "CV", "Boot")[1],
  nfolds = 10,
  seed.CV = NULL,
  foldID = NULL,
  Boot.inference = FALSE,
  B = 500,
  bootID = NULL,
  seed.boot = NULL,
  LR = NULL,
  LR.boot = NULL,
  ...
)
```

Arguments

`formula`	A formula object of the form *response* ~ *other_variables*.
`data`	A data frame containing the variables displayed in the formula.
`standardize`	Should the variables be standardized before the estimation process? Default value is TRUE.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`parallel`	Whether parallel computing should be used to distribute the computations on different CPUs. Either a logical value determining whether parallel computing is used (TRUE) or not (FALSE, the default value). Or a numerical value determining the number of cores to use.
`penalty`	should the regression include a penalty on the coefficients size. If "none" is chosen, a non-penalized Lorenz regression is computed using function `Lorenz.GA`. If "SCAD" is chosen, a penalized Lorenz regression with SCAD penalty is computed using function `Lorenz.SCADFABS`. IF "LASSO" is chosen, a penalized Lorenz regression with LASSO penalty is computed using function `Lorenz.FABS`.
`h.grid`	Only used if penalty="SCAD" or penalty="LASSO". Grid of values for the bandwidth of the kernel, determining the smoothness of the approximation of the indicator function. Default value is (0.1,0.2,1,2,5)*n^(-1/5.5), where n is sample size.
`eps`	Only used if penalty="SCAD" or penalty="LASSO". Step size in the FABS or SCADFABS algorithm. Default value is 0.005.
`sel.choice`	Only used if penalty="SCAD" or penalty="LASSO". Determines what method is used to determine the optimal regularization parameter. Possibles values are any subvector of c("BIC","CV","Boot"). Default is "BIC". Notice that "Boot" is necessarily added if Boot.inference is set to TRUE.
`nfolds`	Only used if sel.choice contains "CV". Number of folds in the cross-validation.
`seed.CV`	Only used if sel.choice contains "CV". Should a specific seed be used in the definition of the folds. Default value is NULL in which case no seed is imposed.
`foldID`	vector taking value from 1 to nfolds specifying the fold index of each observation. Default value is NULL in which case the folds are defined internally.

`Boot.inference`	should bootstrap inference be produced ? Default is FALSE. It is automatically turned to TRUE if sel.choice contains "Boot".
`B`	Only used if Boot.inference is TRUE. Number of bootstrap resamples. Default is 500.
`bootID`	Only used if Boot.inference is TRUE. matrix where each row provides the ID of the observations selected in each bootstrap resample. Default is NULL, in which case these are defined internally.
`seed.boot`	Only used if Boot.inference is TRUE. Should a specific seed be used in the definition of the folds. Default value is NULL in which case no seed is imposed.
`LR`	Estimation on the original sample. Output of a call to `Lorenz.GA` or `PLR.wrap`.
`LR.boot`	Estimation on the bootstrap resamples. In the non-penalized case, it is the output of a call to `Lorenz.boot`. In the penalized case, it is a list of size length(h.grid), where each element is the output of a call to `Lorenz.boot` and uses a different value of the bandwidth.
`...`	Additional parameters corresponding to arguments passed in `Lorenz.GA`, `Lorenz.SCADFABS` or `Lorenz.FABS` depending on the argument chosen in penalty.

Value

For the Non-penalized Lorenz Regression, a list with the following elements :

`theta` the estimated vector of parameters.

`pval.theta` Only returned if Boot.inference is TRUE. the pvalues associated to each element of the parameter vector.

`summary` a vector including the estimated explained Gini coefficient and the Lorenz-R^2.

`Gi.expl` the estimated explained Gini coefficient

`LR2` the Lorenz-R^2 of the regression.

`MRS` the matrix of estimated marginal rates of substitution. More precisely, if we want the MRS of X1 (numerator) with respect to X2 (denominator), we should look for row corresponding to X1 and column corresponding to X2.

`Fit` A data frame containing the response (first column) and the estimated index (second column).

`Gi.star` Only returned if Boot.inference is TRUE. A vector gathering the bootstrap estimators of the explained Gini coefficient.

`LR2.star` Only returned if Boot.inference is TRUE. A vector gathering the bootstrap estimators of the Lorenz-R^2.

`theta.star` Only returned if Boot.inference is TRUE. A matrix gathering the bootstrap estimators of theta (rows refer to bootstrap iterations and columns refer to the different coefficients).

For the Penalized Lorenz Regression, a list with the following elements.

`path` a list where the different elements correspond to the values of h.grid. Each element is a matrix where the first line displays the path of regularization parameters. The second and third lines display the evolution of the Lorenz-R^2 and explained Gini coefficient along that path. The next lines display the evolution of the scores of the methods chosen in sel.choice. The remaining lines display the evolution of the estimated parameter vector.

`theta` a matrix where the different lines correspond to the methods chosen in sel.choice. Each line provides the estimated vector of parameters at the optimal value of the regularization parameter.

`summary` a matrix where the different lines correspond to the methods chosen in sel.choice. Each line provides the estimated explained Gini coefficient, the Lorenz-R^2, the optimal lambda, the optimal bandwidth, the number of selected variables and the scores at the optimal value of the regularization parameter.

`Gi.expl` a vector providing the estimated explained Gini coefficient at the optimal value of the regularization parameter for each method in sel.choice.

`LR2` a vector providing the Lorenz-R^2 at the optimal value of the regularization parameter for each method in sel.choice.

`MRS` a list where the different elements correspond to a method in sel.choice. Each element is a matrix of estimated marginal rates of substitution for non-zero coefficients at the optimal value of the regularization parameter.

`Fit` A data frame containing the response (first column). The remaining columns give the estimated index at the optimal value of the regularization parameter, for each method chosen in sel.choice.

`which.h` a vector providing the index of the optimal bandwidth for each method in sel.choice.

`which.lambda` a vector providing the index of the optimal lambda for each method in sel.choice.

`Gi.star` Only returned if Boot.inference is TRUE. A list (each element a different value of the bandwidth h) of lists (each element a different value of the penalty parameter) of vectors (each element a bootstrap iteration) gathering the bootstrap estimators of the explained Gini coefficient.

`LR2.star` Only returned if Boot.inference is TRUE. Similarly for the Lorenz-R^2

`theta.star` Only returned if Boot.inference is TRUE. A list (each element a different value of the bandwidth h) of lists (each element a different value of the penalty parameter) of matrices (rows are bootstrap iterations and columns refer to the coefficients) gathering the bootstrap estimators of theta.

In both cases, the list also technical information, namely the formula, data, weights and call.

References

Heuchenne, C. and A. Jacquemain (2022). Inference for monotone single-index conditional means: A Lorenz regression approach. *Computational Statistics & Data Analysis 167(C)*. Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient.

See Also

`Lorenz.GA`, `Lorenz.SCADFABS`, `Lorenz.FABS`, `PLR.wrap`, `Lorenz.boot`

Examples

```
data(Data.Incomes)
set.seed(123)
Data <- Data.Incomes[sample(1:nrow(Data.Incomes),50),]
```

```
# 1. Non-penalized regression
NPLR <- Lorenz.Reg(Income ~ ., data = Data, penalty = "none",
                   popSize = 30)
# 2. Penalized regression
PLR <- Lorenz.Reg(Income ~ ., data = Data, penalty = "SCAD",
                  h.grid = nrow(Data.Incomes)^(-1/5.5),
                  sel.choice = c("BIC","CV"), eps = 0.01, nfolds = 5)
# Comparison
NPLR$theta;PLR$theta
NPLR$summary;PLR$summary
```

Lorenz.SCADFABS *Solves the Penalized Lorenz Regression with SCAD penalty*

Description

Lorenz.SCADFABS solves the penalized Lorenz regression with SCAD penalty on a grid of lambda values. For each value of lambda, the function returns estimates for the vector of parameters and for the estimated explained Gini coefficient, as well as the Lorenz-R^2 of the regression.

Usage

```
Lorenz.SCADFABS(
  YX_mat,
  weights = NULL,
  h,
  eps,
  a = 3.7,
  iter = 10^4,
  lambda = "Shi",
  lambda.min = 1e-07,
  gamma = 0.05
)
```

Arguments

YX_mat	a matrix with the first column corresponding to the response vector, the remaining ones being the explanatory variables.
weights	vector of sample weights. By default, each observation is given the same weight.
h	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function.
eps	step size in the FABS algorithm.
a	parameter of the SCAD penalty. Default value is 3.7.
iter	maximum number of iterations. Default value is 10^4.
lambda	this parameter relates to the regularization parameter. Several options are available.

	`grid` If lambda="grid", lambda is defined on a grid, equidistant in the logarithmic scale.
	`Shi` If lambda="Shi", lambda, is defined within the algorithm, as in Shi et al (2018).
	`supplied` If the user wants to supply the lambda vector himself
`lambda.min`	lower bound of the penalty parameter. Only used if lambda="Shi".
`gamma`	value of the Lagrange multiplier in the loss function

Details

The regression is solved using the SCAD-FABS algorithm developed by Jacquemain et al and adapted to our case. For a comprehensive explanation of the Penalized Lorenz Regression, see Heuchenne et al. In order to ensure identifiability, theta is forced to have a L2-norm equal to one.

Value

A list with several components:

`iter` number of iterations attained by the algorithm.

`direction` vector providing the direction (-1 = backward step, 1 = forward step) for each iteration.

`lambda` value of the regularization parameter for each iteration.

`h` value of the bandwidth.

`theta` matrix where column i provides the non-normalized estimated parameter vector for iteration i.

`LR2` vector where element i provides the Lorenz-R^2 of the regression for iteration i.

`Gi.expl` vector where element i provides the estimated explained Gini coefficient for iteration i.

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient.

See Also

`Lorenz.Reg`, `PLR.wrap`, `Lorenz.FABS`

Examples

```
data(Data.Incomes)
YX_mat <- Data.Incomes[,-2]
Lorenz.SCADFABS(YX_mat, h = nrow(Data.Incomes)^(-1/5.5), eps = 0.005)
```

`LorenzRegression` *LorenzRegression : A package to estimate and interpret Lorenz regressions*

Description

The `LorenzRegression` package proposes a toolbox to estimate, produce inference on and interpret Lorenz regressions. As argued in Heuchenne and Jacquemain (2020), these regressions are used to determine the explanatory power of a set of covariates on the inequality of a response variable. In a nutshell, each variable is given a weight in order to maximize the concentration index of the response with respect to a weighted sum of the covariates. The obtained concentration index is called the explained Gini coefficient. If a single-index model with increasing link function is assumed, the explained Gini boils down to the Gini coefficient of the fitted part of the model. This package rests on two main functions: `Lorenz.Reg` for the estimation process and `Lorenz.boot` for more complete inference (tests and confidence intervals).

Details

We direct the user to Heuchenne and Jacquemain (2020) for a rigorous exposition of the methodology and to the vignette Learning Lorenz regressions with examples for a motivational introduction of the `LorenzRegression` package.

References

Heuchenne, C. and A. Jacquemain (2022). Inference for monotone single-index conditional means: A Lorenz regression approach. *Computational Statistics & Data Analysis 167(C).*

`plot.LR` *Plots for the Unpenalized Lorenz Regression*

Description

`plot.LR` provides plots for an object of class `LR`.

Usage

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

Arguments

`x`	Output of a call to `Lorenz.Reg`, where `penalty=="none"`.
`...`	Additional arguments

Value

The Lorenz curve of the response and concentration curve of the response with respect to the estimated index

See Also

Lorenz.Reg

Examples

```
data(Data.Incomes)
NPLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "none")
plot(NPLR)
```

`plot.PLR` *Plots for the Penalized Lorenz Regression*

Description

`plot.PLR` provides plots for an object of class `PLR`.

Usage

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

Arguments

`x`	Output of a call to `Lorenz.Reg`, where `penalty!="none"`.
`...`	Additional arguments.

Value

Three types of plots The first is the Lorenz curve of the response and concentration curves of the response with respect to the estimated index (obtained with each selection method). In each of the remaining graphs, the horizontal axis is -log(lambda), lambda being the value of the regularization parameter. The second type of plot is a traceplot, where the vertical axis gives the size of the coefficient attached to each covariate. The third type of plot shows the evolution of the score(s) for each of the selection method chosen in the `PLR` object. For comparability reasons, the scores are normalized such that the larger the better and the optimum is attained in 1. Since the whole path depends on the chosen bandwidth for the kernel, and the optimal bandwidth may depend on the selection method, the plots are produced for each selection method used in the PLR object

See Also

Lorenz.Reg

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  sel.choice = c("BIC","CV"), h.grid = nrow(Data.Incomes)^(-1/5.5),
                  eps = 0.01, seed.CV = 123, nfolds = 5)
plot(PLR)
```

PLR.BIC *Determines the regularization parameter (lambda) in a PLR via optimization of an information criterion.*

Description

PLR.BIC takes as input a matrix of estimated parameter vectors, where each row corresponds to a covariate and each column corresponds to a value of lambda, and returns the index of the optimal column by optimizing an information criterion. By default the BIC is used.

Usage

```
PLR.BIC(YX_mat, theta, weights = NULL, IC = c("BIC", "AIC"))
```

Arguments

YX_mat	A matrix with the first column corresponding to the response vector, the remaining ones being the explanatory variables.
theta	matrix gathering the path of estimated parameter vectors. Each row corresponds to a given covariate. Each column corresponds to a given value of lambda
weights	vector of sample weights. By default, each observation is given the same weight.
IC	indicates which information criterion is used. Possibles values are "BIC" (default) or "AIC".

Value

A list with two components

val vector indicating the value attained by the information criterion for each value of lambda.

best index of the value of lambda where the optimum is attained.

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient.

See Also

Lorenz.Reg, PLR.wrap, Lorenz.FABS, Lorenz.SCADFABS

Examples

```
data(Data.Incomes)
YX_mat <- Data.Incomes[,-2]
PLR <- PLR.wrap(YX_mat, h = nrow(YX_mat)^(-1/5.5), eps = 0.005)
PLR.BIC(YX_mat, PLR$theta)
```

PLR.CV *Determines the regularization parameter (lambda) in a PLR via cross-validation*

Description

PLR.CV undertakes k-fold cross-validation for a Penalized Lorenz Regression. It returns the CV-score associated to each value of the regularization parameter and the index of the optimum.

Usage

```
PLR.CV(
  formula,
  data,
  penalty = "SCAD",
  h,
  PLR.est = NULL,
  standardize = TRUE,
  weights = NULL,
  eps,
  nfolds = 10,
  foldID = NULL,
  seed.CV = NULL,
  parallel = FALSE,
  ...
)
```

Arguments

formula	A formula object of the form *response ~ other_variables*.
data	A data frame containing the variables displayed in the formula.
penalty	penalty used in the Penalized Lorenz Regression. Possible values are "SCAD" (default) or "LASSO".
h	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function.
PLR.est	Output of a call to PLR.wrap corresponding to the estimation of the Penalized Lorenz Regression on the full sample. Default value is NULL in which case the estimation on the full sample is run internally.

`standardize`	Should the variables be standardized before the estimation process? Default value is TRUE.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`eps`	Step size in the FABS or SCADFABS algorithm. Default value is 0.005.
`nfolds`	Number of folds. Default value is 10.
`foldID`	vector taking value from 1 to nfolds specifying the fold index of each observation. Default value is NULL in which case the folds are defined internally.
`seed.CV`	Should a specific seed be used in the definition of the folds. Default value is NULL in which case no seed is imposed.
`parallel`	Whether parallel computing should be used to distribute the `nfolds` computations on different CPUs. Either a logical value determining whether parallel computing is used (TRUE) or not (FALSE, the default value). Or a numerical value determining the number of cores to use.
`...`	Additional parameters corresponding to arguments passed in `Lorenz.SCADFABS` or `Lorenz.FABS` depending on the argument chosen in penalty.

Value

A list with two components

`val` vector indicating the CV-score for each value of lambda.

`best` index where the optimum is attained.

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2022). A penalised bootstrap estimation procedure for the explained Gini coefficient.

See Also

`Lorenz.Reg`, `PLR.wrap`, `Lorenz.FABS`, `Lorenz.SCADFABS`

Examples

```
YX_mat <- Data.Incomes[,-2]
PLR <- PLR.wrap(YX_mat, h = nrow(YX_mat)^(-1/5.5), eps=0.01)
PLR.CV(Income ~ ., Data.Incomes, PLR.est = PLR,
       h = nrow(Data.Incomes)^(-1/5.5), eps = 0.01, nfolds = 5)
```

PLR.normalize — *Re-normalizes the estimated coefficients of a penalized Lorenz regression*

Description

`PLR.normalize` transforms the estimated coefficients of a penalized Lorenz regression to match the model where the first category of each categorical variable is omitted.

Usage

```
PLR.normalize(PLR)
```

Arguments

PLR	Output of a call to `Lorenz.Reg`, where `penalty!="none"`.

Value

A matrix of re-normalized coefficients.

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  sel.choice = c("BIC","CV"), h.grid = nrow(Data.Incomes)^(-1/5.5),
                  eps = 0.01, seed.CV = 123, nfolds = 5)
PLR.normalize(PLR)
```

PLR.wrap — *Wrapper for the `Lorenz.SCADFABS` and `Lorenz.FABS` functions*

Description

`PLR.wrap` standardizes the covariates, run the penalized regression and spits out the path of parameter vectors.

Usage

```
PLR.wrap(
  YX_mat,
  standardize = TRUE,
  weights = NULL,
  penalty = c("SCAD", "LASSO"),
  h,
  eps = 0.005,
  ...
)
```

Arguments

`YX_mat`	a matrix with the first column corresponding to the response vector, the remaining ones being the explanatory variables.
`standardize`	Should the variables be standardized before the estimation process? Default value is TRUE.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`penalty`	penalty used in the Penalized Lorenz Regression. Possible values are "SCAD" (default) or "LASSO".
`h`	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function.
`eps`	Only used if penalty="SCAD" or penalty="LASSO". Step size in the FABS or SCADFABS algorithm. Default value is 0.005.
`...`	Additional parameters corresponding to arguments passed in `Lorenz.SCADFABS` or `Lorenz.FABS` depending on the argument chosen in penalty.

Value

A list with several components:

`lambda` vector gathering the different values of the regularization parameter

`theta` matrix where column i provides the normalized estimated parameter vector corresponding to value lambda[i] of the regularization parameter.

`LR2` vector where element i provides the Lorenz-R^2 of the regression related to value lambda[i] of the regularization parameter.

`Gi.expl` vector where element i provides the estimated explained Gini coefficient related to value lambda[i] of the regularization parameter.

See Also

`Lorenz.SCADFABS`, `Lorenz.FABS`

Examples

```
data(Data.Incomes)
YX_mat <- Data.Incomes[,-2]
PLR.wrap(YX_mat, h = nrow(Data.Incomes)^(-1/5.5), eps = 0.005)
```

---

print.LR *Printing method for the Lorenz Regression*

Description

`print.LR` prints the arguments and estimated coefficients of an object of class `LR`.

Usage

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

Arguments

`x`	Output of a call to `Lorenz.Reg`, where `penalty="none"`.
`...`	Additional arguments.

Value

No return value, called for printing an object of class `LR` to the console

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
NPLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "none")
print(NPLR)
```

print.PLR — *Printing method for the Penalized Lorenz Regression*

Description

`print.PLR` prints the arguments and estimated coefficients of an object of class `PLR`.

Usage

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

Arguments

`x`	Output of a call to `Lorenz.Reg`, where `penalty!="none"`.
`...`	Additional arguments.

Value

No return value, called for printing an object of class `PLR` to the console

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  sel.choice = c("BIC","CV"), h.grid = nrow(Data.Incomes)^(-1/5.5),
                  eps = 0.01, seed.CV = 123, nfolds = 5)
print(PLR)
```

Rearrangement.estimation — *Estimates a monotonic regression curve via Chernozhukov et al (2009)*

Description

`Rearrangement.estimation` estimates the increasing link function of a single index model via the methodology proposed in Chernozhukov et al (2009).

Usage

```
Rearrangement.estimation(Y, Index, t = Index, weights = NULL, degree.pol = 1)
```

Arguments

`Y`	The response variable.
`Index`	The estimated index. The user may obtain it using function `Lorenz.Reg`.
`t`	A vector of points over which the link function $H(.)$ should be estimated. Default is the estimated index.
`weights`	vector of sample weights. By default, each observation is given the same weight.
`degree.pol`	degree of the polynomial used in the local polynomial regression. Default value is 1.

Details

A first estimator of the link function, neglecting the assumption of monotonicity, is obtained with function `locpol` from the *locpol* package. The final estimator is obtained through the rearrangement operation explained in Chernozhukov et al (2009). This operation is carried out with function `rearrangement` from package `Rearrangement`.

Value

A list with the following components

`t`	the points over which the estimation has been undertaken.
`H`	the estimated link function evaluated at *t*.

References

Chernozhukov, V., I. Fernández-Val, and A. Galichon (2009). Improving Point and Interval Estimators of Monotone Functions by Rearrangement. *Biometrika 96 (3)*. 559–75.

See Also

`Lorenz.Reg`, `locpol`, `rearrangement`

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  h.grid = nrow(Data.Incomes)^(-1/5.5), eps = 0.01)
Y <- PLR$Fit[,1]
Index <- PLR$Fit[,2]
Rearrangement.estimation(Y = Y, Index = Index)
```

summary.LR *Summary for the Lorenz Regression*

Description

`summary.LR` provides a summary for an object of class `LR`.

Usage

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

Arguments

`object`	Output of a call to `Lorenz.Reg`, where `penalty=="none"`.
`...`	Additional arguments

Value

A summary displaying the explained Gini coefficient, Lorenz-R^2 and a table gathering the estimated coefficients, including p-values if bootstrap was performed.

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
NPLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "none")
summary(NPLR)
```

summary.PLR *Summary for the Penalized Lorenz Regression*

Description

`summary.PLR` provides a summary for an object of class `PLR`.

Usage

```
## S3 method for class 'PLR'
summary(object, renormalize = TRUE, ...)
```

Arguments

`object`	Output of a call to `Lorenz.Reg`, where `penalty!="none"`.
`renormalize`	whether the coefficient vector should be re-normalized to match the representation where the first category of each categorical variable is omitted. Default value is TRUE
`...`	Additional arguments.

Value

A summary displaying two tables: a summary of the model and the estimated coefficients.

See Also

`Lorenz.Reg`

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  sel.choice = c("BIC","CV"), h.grid = nrow(Data.Incomes)^(-1/5.5),
                  eps = 0.01, seed.CV = 123, nfolds = 5)
summary(PLR)
```

Index

∗ **datasets**
    Data.Incomes, 7
.Fitness_cpp, 2

boot.confint, 3

coef.LR, 3
coef.PLR, 4
confint.LR, 5
confint.PLR, 6

Data.Incomes, 7

ga, *15, 16*
Gini.coef, 8, *12*, *16*

locpol, *32*
Lorenz.boot, 9, *19, 20*, *23*
Lorenz.curve, *9*, 11, *16*
Lorenz.FABS, *10, 11*, 12, *18–20*, *22*, *25*, *27–29*
Lorenz.GA, *10, 11*, 14, *17–20*
Lorenz.graphs, *9*, *12*, 16
Lorenz.Population, 17
Lorenz.Reg, *3–7*, *11*, *14*, *16*, 17, *22–25*, *27, 28*, *30–34*
Lorenz.SCADFABS, *10, 11*, *14*, *18–20*, 21, *25*, *27–29*
LorenzRegression, 23

plot.LR, 23
plot.PLR, 24
PLR.BIC, 25
PLR.CV, 26
PLR.normalize, 28
PLR.wrap, *10, 11*, *14*, *19, 20*, *22*, *25–27*, 28
print.LR, 30
print.PLR, 31

rearrangement, *32*
Rearrangement.estimation, 31

summary.LR, 33
summary.PLR, 33