Package 'LorenzRegression'

June 27, 2025

Type Package

Title Lorenz and Penalized Lorenz Regressions

Version 2.2.0

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> and in Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2024) <doi:10.1214/23-EJS2200>.

License GPL-3

Encoding UTF-8

Depends R (>= 3.3.1)

LazyData true

Imports stats, ggplot2, parsnip, boot, rsample, parallel, doParallel, foreach, MASS, GA, Rearrangement, progress, Rcpp (>= 0.11.0)

RoxygenNote 7.3.2

Suggests rmarkdown

LinkingTo Rcpp, RcppArmadillo

URL https://github.com/AlJacq/LorenzRegression

BugReports https://github.com/AlJacq/LorenzRegression/issues

ByteCompile true

NeedsCompilation yes

Author Alexandre Jacquemain [aut, cre] (ORCID: <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 2025-06-27 07:30:12 UTC

Contents

`autoplot.LR` *Plots for the Lorenz regression*

Description

`autoplot` generates a plot for an object of class `"LR"` and returns it as a `ggplot` object. The `plot` method is a wrapper around `autoplot` that directly displays the plot, providing a more familiar interface for users accustomed to base R plotting.

Usage

```
## S3 method for class 'LR'
autoplot(object, type = c("explained", "residuals"), band.level = 0.95, ...)

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

Arguments

object	An object of class "LR".
type	A character string indicating the type of plot. Possible values are "explained" and "residuals". • If "explained" is selected, the graph displays the Lorenz curve of the response and concentration curve of the response with respect to the estimated index. If object inherits from "PLR_boot" and LC_store was set to TRUE in Lorenz.boot, pointwise confidence intervals for the concentration curve are added. Their confidence level is set via the argument band.level. • If "residuals" is selected, the graph displays a scatterplot of residuals with respect to the estimated index. Obtaining residuals entail to estimate the link function of the single-index. This is performed via the function Rearrangement.estimation, as explained in predict.LR.
band.level	Confidence level for the bootstrap confidence intervals.
...	Additional arguments passed either to Lorenz.graphs (for type = "explained") or to fitted.LR and residuals.LR (for type = "residuals").
x	An object of class "LR".

Value

autoplot returns a ggplot object representing the desired graph. plot directly displays this plot.

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg)
```

`autoplot.PLR` *Plots for the penalized Lorenz regression*

Description

`autoplot` generates summary plots for an object of class `"PLR"` and returns them as `ggplot` objects. The `plot` method is a wrapper around `autoplot` that directly displays the plot, providing a more familiar interface for users accustomed to base R plotting.

Usage

```
## S3 method for class 'PLR'
autoplot(
  object,
  type = c("explained", "traceplot", "diagnostic", "residuals"),
  traceplot.which = "BIC",
  pars.idx = "BIC",
  score.df = NULL,
  band.level = 0.95,
  ...
)

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

Arguments

`object`	An object of class `"PLR"`. The object might also have S3 classes `"PLR_boot"` and/or `"PLR_cv"` (both inherit from class `"PLR"`)
`type`	A character string indicating the type of plot. Possible values are `"explained"`, `"traceplot"` and `"diagnostic"`.

- If `"explained"` is selected, the graph displays the Lorenz curve of the response and concentration curve of the response with respect to the estimated index. The grid and penalty parameters used to estimate the index are chosen via the `pars.idx` argument. If `object` inherits from `"PLR_boot"` and `LC_store` was set to `TRUE` in `Lorenz.boot`, pointwise confidence intervals for the concentration curve are added. Their confidence level is set via the argument `band.level`.
- If `"traceplot"` is selected, the graph displays a traceplot, where the horizontal axis is -log(lambda), lambda being the value of the penalty parameter. The vertical axis gives the value of the estimated coefficient attached to each covariate.
- If `"diagnostic"` is selected, the graph displays a faceted plot, where each facet corresponds to a different value of the grid parameter. Each plot shows the evolution of the scores of each available selection method. For comparability reasons, the scores are normalized such that the larger the better and the optimum is attained in 1.

- If "residuals" is selected, the graph displays a scatterplot of residuals with respect to the estimated index. The grid and penalty parameters used for estimation are chosen via the pars.idx argument. Obtaining residuals entail to estimate the link function of the single-index. This is performed via the function Rearrangement.estimation, as explained in predict.LR.

traceplot.which
: This argument indicates the value of the grid parameter for which the traceplot should be produced (see arguments grid.value and grid.arg in function Lorenz.Reg). It can be an integer indicating the index in the grid determined via grid.value. Alternatively, it can be a character string indicating the selection method. In this case the index corresponds to the optimal value according to that selection method.

pars.idx
: What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are:

  - "BIC" (default) - Always available.
  - "Boot" - Available if object inherits from "PLR_boot".
  - "CV" - Available if object inherits from "PLR_cv".

  Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.

score.df
: A data.frame providing the scores to be displayed if type is set to "diagnostic". For internal use only.

band.level
: Confidence level for the bootstrap confidence intervals.

...
: Additional arguments passed either to Lorenz.graphs (for type = "explained") or to fitted.PLR and residuals.PLR (for type = "residuals").

x
: An object of class "PLR". The object might also have S3 classes "PLR_boot" and/or "PLR_cv" (both inherit from class "PLR")

Details

The available selection methods depend on the classes of the object: BIC is always available, bootstrap is available if object inherits from "PLR_boot", cross-validation is available if object inherits from "PLR_cv"

Value

autoplot returns a ggplot object representing the desired graph. plot directly displays this plot.

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

coef.LR *Estimated coefficients for the Lorenz regression*

Description

Provides the estimated coefficients for an object of class "LR".

Usage

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

Arguments

object	An object of S3 class "LR".
...	Additional arguments.

Value

a vector gathering the estimated coefficients

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg)
```

coef.PLR *Estimated coefficients for the penalized Lorenz regression*

Description

Provides the estimated coefficients for an object of class "PLR".

Usage

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

Arguments

`object`	An object of S3 class `"PLR"`. The object might also have S3 classes `"PLR_boot"` and/or `"PLR_cv"` (both inherit from class `"PLR"`)
`renormalize`	A logical value determining 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
`pars.idx`	What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are: • `"BIC"` (default) - Always available. • `"Boot"` - Available if `object` inherits from `"PLR_boot"`. • `"CV"` - Available if `object` inherits from `"PLR_cv"`. Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.
`...`	Additional arguments

Value

a vector gathering the estimated coefficients.

See Also

`Lorenz.Reg`

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

---

confint.LR_boot *Confidence intervals for the Lorenz regression*

Description

Provides bootstrap confidence intervals for the explained Gini coefficient, Lorenz-R2 and theta vector for an object of class `"LR_boot"`.

Usage

```
## S3 method for class 'LR_boot'
confint(
  object,
  parm = c("Gini", "LR2", "theta"),
  level = 0.95,
  type = c("norm", "basic", "perc"),
  bias.corr = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"LR_boot"`. The current implementation requires bootstrap to construct confidence intervals. Hence, it is not sufficient that `object` inherits from `"LR"`.
`parm`	A logical value determining whether the confidence interval is computed for the explained Gini coefficient, for the Lorenz-R^2 or for the vector of coefficients of the single-index model. Possible values are `"Gini"` (default, for the explained Gini),`"LR2"` (for the Lorenz-R^2) and `"theta"` (for the index coefficients).
`level`	A numeric giving the level of the confidence interval. Default value is 0.95.
`type`	A character string specifying the bootstrap method. Possible values are `"norm"`, `"basic"` and `"perc"`. For more information, see the argument `type` of the function `boot.ci` from the *boot* library.
`bias.corr`	A logical determining whether bias correction should be performed. Only used if `type="norm"`. Default is `TRUE`.
`...`	Additional arguments.

Value

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

See Also

`Lorenz.boot`, `boot.ci`

Examples

```
## For examples see example(Lorenz.boot)
```

`confint.PLR_boot` *Confidence intervals for the penalized Lorenz regression*

Description

Provides bootstrap confidence intervals for the explained Gini coefficient and Lorenz-R^2 for an object of class `"PLR_boot"`.

Usage

```
## S3 method for class 'PLR_boot'
confint(
  object,
  parm = c("Gini", "LR2"),
  level = 0.95,
  type = c("norm", "basic", "perc"),
```

```
  pars.idx = "BIC",
  bias.corr = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"PLR_boot"`. The object might also have S3 class `"PLR_cv"`. The current implementation requires bootstrap to construct confidence intervals. Hence, it is not sufficient that `object` inherits from `"PLR"`.
`parm`	A character string determining whether the confidence interval is computed for the explained Gini coefficient or for the Lorenz-R^2. Possible values are `"Gini"` (default, for the explained Gini) and `"LR2"` (for the Lorenz-R^2)
`level`	A numeric giving the level of the confidence interval. Default value is 0.95.
`type`	A character string specifying the bootstrap method. Possible values are `"norm"`, `"basic"` and `"perc"`. For more information, see the argument `type` of the function `boot.ci` from the *boot* library.
`pars.idx`	What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are: • `"BIC"` (default). • `"Boot"`. • `"CV"` - Available if `object` inherits from `"PLR_cv"`. Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.
`bias.corr`	A logical determining whether bias correction should be performed. Only used if `type="norm"`. Default is `TRUE`.
`...`	Additional arguments.

Value

A vector providing the desired confidence interval.

See Also

`Lorenz.boot`, `boot.ci`

Examples

```
## For examples see example(Lorenz.boot)
```

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

diagnostic.PLR *Diagnostic for the penalized Lorenz regression*

Description

`diagnostic.PLR` provides diagnostic information for an object of class `"PLR"` It restricts the path of the PLR to pairs of parameters (grid, lambda) that satisfy a threshold criterion.

Usage

```
diagnostic.PLR(
  object,
  tol = 0.99,
  method = c("union", "intersect", "BIC", "Boot", "CV")
)
```

Arguments

`object`	An object of class `"PLR"`.
`tol`	A numeric threshold value used to restrict the PLR path. More specifically, we restrict to pairs (grid,lambda) whose normalized score exceeds `tol`. Default value is 0.95.
`method`	A character string specifying the method used to evaluate the scores. Options are `"union"`, `"intersect"`, `"BIC"`, `"Boot"`, and `"CV"`.

"BIC" The score is the BIC-score.

"Boot" The score is the OOB-score.

"CV" The score is the CV-score.

"union" The threshold requirement must be met for at least one of the selection methods available.

"intersect" The threshold requirement must be met for all selection methods available.

Value

A list with two elements:

`path` The restricted model path, containing only the values of the pair (grid, lambda) that satisfy the threshold criterion.

`best` The best model. It is obtained by considering the pair (grid, lambda) in the restricted path that leads to the sparsest model. If several pairs yield the same level of sparsity, we consider the pair that maximizes the minimum score across all selection methods available.

See Also

`Lorenz.Reg`

Examples

```
# Continuing the  Lorenz.boot(.) example:
# The out-of-bag score seems to remain relatively flat when lambda is small enough
plot(PLR_boot, type = "diagnostic")
# What is the best pair (grid,penalty) parameter that is close enough to the highest OOB score
diagnostic.PLR(PLR_boot, tol = 0.99, method = "Boot")
# We want the solution to be close to the best, for both the BIC and OOB scores.
diagnostic.PLR(PLR_boot, method = "intersect")
```

Gini.coef *Concentration index of* y *with respect to* 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.

Details

The parameter seed allows for local seed setting to control randomness in the generation of the uniform random variables. The specified seed is applied to the respective part of the computation, and the seed is reverted to its previous state after the operation. This ensures that the seed settings do not interfere with the global random state or other parts of the code.

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)
```

---

ineqExplained *Retrieve a measure of explained inequality from a model*

Description

This generic function extracts a measure of explained inequality, such as the explained Gini coefficient or the Lorenz-R2, from a fitted model object.

Usage

```
ineqExplained(object, type = c("Gini.explained", "Lorenz-R2"), ...)
```

Arguments

object	An object for which the inequality metrics should be extracted.
type	Character string specifying the type of inequality metric to retrieve. Options are "Gini.explained" for the explained Gini coefficient or "Lorenz-R2" for the Lorenz-R^2.
...	Additional arguments passed to specific methods.

Value

The requested inequality metric.

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg)
```

ineqExplained.lm *Explained inequality metrics for (generalized) linear models*

Description

Retrieves the explained Gini coefficient or the Lorenz-R^2 from an object of class `"lm"`.

Usage

```
## S3 method for class 'lm'
ineqExplained(object, type = c("Gini.explained", "Lorenz-R2"), ...)
```

Arguments

`object`	An object of S3 class `"lm"`.
`type`	Character string specifying the type of inequality metric to retrieve. Options are `"Gini.explained"` (for the explained Gini coefficient) and `"Lorenz-R2"` (for the Lorenz-R^2).
`...`	Additional arguments passed to `Gini.coef`.

Value

A numeric value representing the requested inequality metric.

ineqExplained.LR *Explained inequality metrics for the Lorenz regression*

Description

Retrieves the explained Gini coefficient or the Lorenz-R^2 from an object of class `"LR"`.

Usage

```
## S3 method for class 'LR'
ineqExplained(object, type = c("Gini.explained", "Lorenz-R2"), ...)
```

Arguments

`object`	An object of S3 class `"LR"`.
`type`	Character string specifying the type of inequality metric to retrieve. Options are `"Gini.explained"` (for the explained Gini coefficient) and `"Lorenz-R2"` (for the Lorenz-R^2).
`...`	Additional arguments.

Value

A numeric value representing the requested inequality metric.

`ineqExplained.PLR` *Explained inequality metrics for the penalized Lorenz regression*

Description

Retrieves the explained Gini coefficient or the Lorenz-R^2 from an object of class `"PLR"`.

Usage

```
## S3 method for class 'PLR'
ineqExplained(
  object,
  type = c("Gini.explained", "Lorenz-R2"),
  pars.idx = "BIC",
  ...
)
```

Arguments

`object`	An object of S3 class `"PLR"`. The object might also have S3 classes `"PLR_boot"` and/or `"PLR_cv"` (both inherit from class `"PLR"`)
`type`	Character string specifying the type of inequality metric to retrieve. Options are `"Gini.explained"` (for the explained Gini coefficient) and `"Lorenz-R2"` (for the Lorenz-R^2).
`pars.idx`	What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are: • `"BIC"` (default) - Always available. • `"Boot"` - Available if `object` inherits from `"PLR_boot"`. • `"CV"` - Available if `object` inherits from `"PLR_cv"`. Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.
`...`	Additional arguments.

Value

A numeric value representing the requested inequality metric.

`Lorenz.boot` *Bootstrap for the (penalized) Lorenz regression*

Description

`Lorenz.boot` performs bootstrap estimation for the vector of coefficients of the single-index model, the explained Gini coefficient, and the Lorenz-R^2. In the penalized case, it also provides a selection method.

Usage

```
Lorenz.boot(
  object,
  R,
  boot_out_only = FALSE,
  store_LC = FALSE,
  show_progress = TRUE,
  ...
)
```

Arguments

`object`	An object of class `"LR"` or `"PLR"`, i.e., the output of a call to `Lorenz.Reg`.
`R`	An integer specifying the number of bootstrap replicates.
`boot_out_only`	A logical value indicating whether the function should return only the raw bootstrap output. This advanced feature can help save computation time in specific use cases. See Details.
`store_LC`	A logical determining whether explained Lorenz curves ordinates should be stored for each bootstrap sample. The default is `FALSE` since it might require storing large objects. If set to `TRUE`, ordinates are stored and plots of the explained Lorenz curve will include confidence bands, see `plot.LR` and `plot.PLR`.
`show_progress`	A logical. If `TRUE` (default), a progress bar is displayed during bootstrap computation. Set to `FALSE` to disable it. Progress is not shown when parallel computing is used.
`...`	Additional arguments passed to either the bootstrap function `boot` from the **boot** package or the underlying fit functions (`Lorenz.GA`, `Lorenz.FABS`, or `Lorenz.SCADFABS`). By default, the fit function uses the same parameters as in the original call to `Lorenz.Reg`, but these can be overridden by explicitly passing them in `...`.

Details

The function supports parallel computing in two ways:

1. Using the built-in parallelization options of `boot`, which can be controlled via the `...` arguments such as `parallel`, `ncpus`, and `cl`.

2. Running multiple independent instances of `Lorenz.boot()`, each handling a subset of the bootstrap samples. In this case, setting `boot_out_only = TRUE` ensures that the function only returns the raw bootstrap results. These results can be merged using `Lorenz.boot.combine`.

Handling of additional arguments (. . .): The function allows for two types of arguments through . . . :

- Arguments for `boot`, used to control the bootstrap procedure.
- Arguments for the underlying fit functions (`Lorenz.GA`, `Lorenz.FABS`, or `Lorenz.SCADFABS`). By default, the function retrieves these parameters from the original `Lorenz.Reg` call. However, users can override them by explicitly specifying new values in . . . .

Value

An object of class `c("LR_boot", "LR")` or `c("PLR_boot", "PLR")`, depending on whether a non-penalized or penalized regression was fitted.

The methods `confint.LR` and `confint.PLR` can be used on objects of class `"LR_boot"` or `"PLR_boot"` to construct confidence intervals for the model parameters.

For the non-penalized Lorenz regression, the returned object is a list containing:

`theta` The estimated vector of parameters. In the penalized case, this is a matrix where each row corresponds to a different selection method (e.g., BIC, bootstrap, cross-validation).

`Gi.expl` The estimated explained Gini coefficient. In the penalized case, this is a vector, where each element corresponds to a different selection method.

`LR2` The Lorenz-R^2 of the regression. In the penalized case, this is a vector, where each element corresponds to a different selection method.

`boot_out` An object of class `"boot"` containing the raw bootstrap output.

For the penalized Lorenz regression, the returned object includes:

`path` See `Lorenz.Reg` for the original path. The out-of-bag (OOB) score is added.

`lambda.idx` A vector indicating the index of the optimal lambda obtained by each selection method.

`grid.idx` A vector indicating the index of the optimal grid parameter obtained by each selection method.

Note: In the penalized case, the returned object may have additional classes such as `"PLR_cv"` if cross-validation was performed and used for selection.

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 (2024). A penalized bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300*.

See Also

`Lorenz.Reg`, `Lorenz.GA`, `Lorenz.SCADFABS`, `Lorenz.FABS`, `PLR.CV`, `boot`

Examples

```
# Non-penalized regression example (not run due to execution time)
## Not run:
set.seed(123)
NPLR_boot <- Lorenz.boot(NPLR, R = 30)
confint(NPLR_boot) # Confidence intervals
summary(NPLR_boot)

## End(Not run)

# Penalized regression example:
set.seed(123)
PLR_boot <- Lorenz.boot(PLR, R = 20)
print(PLR_boot)
summary(PLR_boot)
coef(PLR_boot, pars.idx = "Boot")
predict(PLR_boot, pars.idx = "Boot")
plot(PLR_boot)
plot(PLR_boot, type = "diagnostic")

# Confidence intervals for different selection methods:
confint(PLR_boot, pars.idx = "BIC")  # Using BIC-selected tuning parameters
confint(PLR_boot, pars.idx = "Boot") # Using bootstrap-selected tuning parameters
```

Lorenz.boot.combine *Combines bootstrap Lorenz regressions*

Description

`Lorenz.boot.combine` combine outputs of different instances of the `Lorenz.boot` function.

Usage

```
Lorenz.boot.combine(boot_list)
```

Arguments

`boot_list`	list of objects, each element being the output of a call to the function `Lorenz.boot`.

Value

An object of class `c("LR_boot", "LR")` or `c("PLR_boot", "PLR")`, depending on whether a non-penalized or penalized regression was fitted.

The method `confint` is used on an object of class `"LR_boot"` or `"PLR_boot"` to obtain bootstrap inference on the model parameters.

For the non-penalized Lorenz regression, the returned object is a list containing the following components:

`theta` The estimated vector of parameters. In the penalized case, it is a matrix where each row corresponds to a different selection method (e.g., BIC, bootstrap, cross-validation).

`Gi.expl` The estimated explained Gini coefficient. In the penalized case, it is a vector, where each element corresponds to a different selection method.

`LR2` The Lorenz-R^2 of the regression. In the penalized case, it is a vector, where each element corresponds to a different selection method.

`boot_out` An object of class `"boot"` containing the output of the bootstrap calculation.

For the penalized Lorenz regression, the returned object is a list containing the following components:

`path` See `Lorenz.Reg` for the original path. To this path is added the out-of-bag (OOB) score.

`lambda.idx` A vector indicating the index of the optimal lambda obtained by each selection method.

`grid.idx` A vector indicating the index of the optimal grid parameter obtained by each selection method.

Note: The returned object may have additional classes such as `"PLR_cv"` if cross-validation was performed and used as a selection method in the penalized case.

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 (2024). A penalised bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300*.

See Also

`Lorenz.boot`

Examples

```
# Continuing the Lorenz.Reg(.) example for the penalized regression:
boot_list <- list()
set.seed(123)
boot_list[[1]] <- Lorenz.boot(PLR, R = 10, boot_out_only = TRUE)
set.seed(456)
boot_list[[2]] <- Lorenz.boot(PLR, R = 10, boot_out_only = TRUE)
PLR_boot <- Lorenz.boot.combine(boot_list)
summary(PLR_boot)
```

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,
  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.
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.

Details

The parameter seed allows for local seed setting to control randomness in the generation of the uniform random variables. The specified seed is applied to the respective part of the computation, and the seed is reverted to its previous state after the operation. This ensures that the seed settings do not interfere with the global random state or other parts of the code.

Value

A function corresponding to the estimated Lorenz or concentration curve.

See Also

Lorenz.graphs, Gini.coef

Examples

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

Lorenz.FABS *Estimates the parameter vector in a 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(
  y,
  x,
  standardize = TRUE,
  weights = NULL,
  kernel = 1,
  h = length(y)^(-1/5.5),
  gamma = 0.05,
  lambda = "Shi",
  w.adaptive = NULL,
  eps = 0.005,
  iter = 10^4,
  lambda.min = 1e-07
)
```

Arguments

y	a vector of responses
x	a matrix of 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.
`kernel`	integer indicating what kernel function to use. The value 1 is the default and implies the use of an Epanechnikov kernel while the value of 2 implies the use of a biweight kernel.
`h`	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function. Default value is n^(-1/5.5) where n is the sample size.
`gamma`	value of the Lagrange multiplier in the loss function
`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
`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. Default value is 0.005.
`iter`	maximum number of iterations. Default value is 10^4.
`lambda.min`	lower bound of the penalty parameter. Only used if `lambda="Shi"`.

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:

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

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

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

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

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2024). A penalised bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300.*

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, Lorenz.SCADFABS

Examples

```
data(Data.Incomes)
y <- Data.Incomes[,1]
x <- as.matrix(Data.Incomes[,-c(1,2)])
Lorenz.FABS(y, x)
```

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

Description

Lorenz.GA estimates the coefficient vector of the single-index model. It also returns the Lorenz-R^2 of the regression as well as the estimated explained Gini coefficient.

Usage

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

Arguments

y	a vector of responses
x	a matrix of 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.

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.
suggestions	Initial guesses used in the genetic algorithm. The default value is NULL, meaning no suggestions are passed. Other possible values are a numeric matrix with at most popSize rows and ncol(x) columns, or a character string "OLS". In the latter case, 0.5*popSize suggestions are created as random perturbations of the OLS solutions.
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	An optional seed for generating the vector of uniform random variables used to break ties in the genetic algorithm. Defaults to NULL, which means no specific seed is set.
seed.Gini	An optional seed for generating the vector of uniform random variables used to break ties in the computation of the Gini coefficient. Defaults to NULL, meaning no specific seed is applied.
seed.GA	An optional seed for ga, used during the fitting of the genetic algorithm. Defaults to NULL, implying that no specific seed is set.
parallel.GA	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 (2022)

The parameters seed.random, seed.Gini, and seed.GA allow for local seed setting to control randomness in specific parts of the function. Each seed is applied to the respective part of the computation, and the seed is reverted to its previous state after the operation. This ensures that the seed settings do not interfere with the global random state or other parts of the code.

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)
y <- Data.Incomes$Income
x <- cbind(Data.Incomes$Age, Data.Incomes$Work.Hours)
Lorenz.GA(y, x, 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, difference = FALSE, ...)
```

Arguments

`formula`	A formula object of the form *response ~ other_variables*. The form *response ~ 1* is used to display only the Lorenz curve of the response.
`data`	A dataframe containing the variables of interest
`difference`	A logical determining whether the vertical axis should be expressed in terms of deviation from perfect equality. Default is `FALSE`.
`...`	Further 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)
# Expressing now the vertical axis as the deviation from perfect equality
Lorenz.graphs(Income ~ Age + Work.Hours, data = Data.Incomes, difference = TRUE)
```

Lorenz.Reg *Fits a Lorenz regression*

Description

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

Usage

```
Lorenz.Reg(
  formula,
  data,
  weights,
  na.action,
  penalty = c("none", "SCAD", "LASSO"),
  grid.arg = c("h", "SCAD.nfwd", "eps", "kernel", "a", "gamma"),
  grid.value = NULL,
  ...
)
```

Arguments

formula	An object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted.
data	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), typically the environment from which Lorenz.Reg is called.

weights	An optional vector of sample weights to be used in the fitting process. Should be NULL or a numeric vector.
na.action	A function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.
penalty	A character string specifying the type of penalty on the size of the estimated coefficients of the single-index model. The default value is "none", in which case a non-penalized Lorenz regression is fitted using Lorenz.GA. Other possible values are "LASSO" and "SCAD", in which case a penalized Lorenz regression is fitted using Lorenz.FABS or Lorenz.SCADFABS respectively.
grid.arg	A character string specifying the tuning parameter for which a grid is to be constructed, see Details.
grid.value	A numeric vector specifying the grid values, see Details.
...	Additional parameters corresponding to arguments passed in Lorenz.GA, Lorenz.FABS or Lorenz.SCADFABS, depending on the argument chosen in penalty.

Details

In the penalized case, the model is fitted for a grid of values of two parameters : the penalty parameter (lambda) and one tuning parameter specified by the arguments grid.arg and grid.value. The possibles values for grid.arg are tuning parameters of the functions Lorenz.FABS and Lorenz.SCADFABS : ''h'' (the default), ''SCAD.nfwd'',''eps'', ''kernel'', ''a'' and ''gamma''. The values for the grid are specified with grid.value. The default is NULL, in which case no grid is constructed

Value

An object of class "LR" for the non-penalized Lorenz regression or of class "PLR" for a penalized Lorenz regression.

Several methods are available for both classes to facilitate model analysis. Use summary.LR or summary.PLR to summarize the model fits. Extract the coefficients of the single-index model using coef.LR or coef.PLR. Measures of explained inequality (Gini coefficient and Lorenz-R^2) are retrieved using ineqExplained.LR or ineqExplained.PLR. Obtain predictions with predict.LR or predict.PLR, and fitted values with fitted.LR or fitted.PLR. For visual representations of explained inequality, use autoplot.LR and plot.LR, or autoplot.PLR and plot.PLR.

The object of class "LR" is a list containing the following components:

theta The estimated vector of parameters.

Gi.expl The estimated explained Gini coefficient.

LR2 The Lorenz-R^2 of the regression.

The object of class "PLR" is a list containing the following components:

path A list where the different elements correspond to the values of the grid parameter. Each element is a matrix where the first line displays the vector of lambda values. The second and third lines display the evolution of the Lorenz-R^2 and explained Gini coefficient along that vector. The next lines display the evolution of the BIC score. The remaining lines display the evolution of the estimated coefficients of the single-index model.

`lambda.idx` the index of the optimal lambda obtained by the BIC method

`grid.idx` the index of the optimal grid parameter obtained by the BIC method.

In both cases, the list also provides technical information, such as the specified `formula`, `weights` and `call`, as well as the design matrix `x` and the response vector `y`.

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 (2024). A penalised bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300*.

See Also

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

Examples

```
data(Data.Incomes)
set.seed(123)
data <- Data.Incomes[sample(1:200,40),]
# 1. Non-penalized regression
NPLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "none", popSize = 15)
# 2. Penalized regression
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes, penalty = "SCAD",
                  eps = 0.06, grid.arg = "h",
                  grid.value=c(0.5,1,2)*nrow(Data.Incomes)^(-1/5.5))
# Print method
print(NPLR)
print(PLR)
# Summary method
summary(NPLR)
summary(PLR)
# Coef method
coef(NPLR)
coef(PLR)
# ineqExplained method
ineqExplained(NPLR)
ineqExplained(PLR)
# Predict method
## One can predict either the index or the response
predict(NPLR,type="response")
predict(PLR,type="response")
# Plot method
## The default displays the explained and observed Lorenz curve.
plot(NPLR)
plot(PLR)
## It is also possible to display a residuals plot.
plot(PLR,type="residuals")
## For PLR only, one can obtain a traceplot of the penalized coefficients
plot(PLR,type="traceplot")
```

`Lorenz.SCADFABS` *Estimates the parameter vector in a 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(
  y,
  x,
  standardize = TRUE,
  weights = NULL,
  kernel = 1,
  h = length(y)^(-1/5.5),
  gamma = 0.05,
  a = 3.7,
  lambda = "Shi",
  eps = 0.005,
  SCAD.nfwd = NULL,
  iter = 10^4,
  lambda.min = 1e-07
)
```

Arguments

`y`	a vector of responses
`x`	a matrix of 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.
`kernel`	integer indicating what kernel function to use. The value 1 is the default and implies the use of an Epanechnikov kernel while the value of 2 implies the use of a biweight kernel.
`h`	bandwidth of the kernel, determining the smoothness of the approximation of the indicator function. Default value is n^(-1/5.5) where n is the sample size.
`gamma`	value of the Lagrange multiplier in the loss function
`a`	parameter of the SCAD penalty. Default value is 3.7.

`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
`eps`	step size in the FABS algorithm. Default value is 0.005.
`SCAD.nfwd`	optional tuning parameter used if penalty="SCAD". Default value is NULL. The larger the value of this parameter, the sooner the path produced by the SCAD will differ from the path produced by the LASSO.
`iter`	maximum number of iterations. Default value is 10^4.
`lambda.min`	lower bound of the penalty parameter. Only used if lambda="Shi".

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:

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

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

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

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

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2024). A penalised bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300.*

See Also

`Lorenz.Reg`, `Lorenz.FABS`

Examples

```
data(Data.Incomes)
y <- Data.Incomes[,1]
x <- as.matrix(Data.Incomes[,-c(1,2)])
Lorenz.SCADFABS(y, x)
```

PLR.CV *Cross-validation for penalized Lorenz regression*

Description

PLR.CV performs k-fold cross-validation to select the grid and penalization parameters of the penalized Lorenz regression.

Usage

```
PLR.CV(object, k, seed.CV = NULL, parallel = FALSE, ...)
```

Arguments

object	An object of class "PLR", i.e., the result of a call to Lorenz.Reg where penalty is either "SCAD" or "LASSO".
k	An integer specifying the number of folds in the k-fold cross-validation.
seed.CV	An optional integer specifying a seed for reproducibility in the creation of the folds. Default is NULL, in which case no seed is imposed.
parallel	A logical or numeric value controlling parallel computation. If TRUE, parallel processing is enabled using the maximum available cores minus one. If a numeric value is provided, it specifies the number of cores to use. Default is FALSE (no parallelization).
...	Additional arguments passed to either the cross-validation function vfold_cv from the **rsample** package or the underlying fit functions (Lorenz.GA, Lorenz.FABS, or Lorenz.SCADFABS). By default, the fit function uses the same parameters as in the original call to Lorenz.Reg, but these can be overridden by explicitly passing them in ....

Details

The parameter seed.CV allows for local seed setting to control randomness in the generation of the folds. The specified seed is applied to the respective part of the computation, and the seed is reverted to its previous state after the operation. This ensures that the seed settings do not interfere with the global random state or other parts of the code.

Value

An object of class c("PLR_cv", "PLR"). The returned list contains the following components:

path See Lorenz.Reg for the original path. The cross-validation score is added.

lambda.idx A vector indicating the index of the optimal lambda obtained by each selection method.

grid.idx A vector indicating the index of the optimal grid parameter obtained by each selection method.

splits A list storing the data splits used for cross-validation, as generated by vfold_cv.

Note: The returned object may have additional classes such as "PLR_boot" if bootstrap was performed.

References

Jacquemain, A., C. Heuchenne, and E. Pircalabelu (2024). A penalised bootstrap estimation procedure for the explained Gini coefficient. *Electronic Journal of Statistics 18(1) 247-300.*

See Also

Lorenz.Reg, Lorenz.SCADFABS, Lorenz.FABS, Lorenz.boot

Examples

```
# Continuing the Lorenz.Reg(.) example:
PLR_CV <- PLR.CV(PLR, k = 5, seed.CV = 123)
# The object now inherits from the class "PLR_cv".
# Hence the methods (also) display the results obtained by cross-validation.
print(PLR_CV)
summary(PLR_CV)
coef(PLR_CV, pars.idx = "CV")
predict(PLR_CV, pars.idx = "CV")
plot(PLR_CV)
plot(PLR_CV, type = "diagnostic") # Plot of the scores depending on the grid and penalty parameters
```

PLR.fit *Penalized Lorenz Regression Fit Function*

Description

PLR.fit fits a penalized Lorenz regression model using either the LASSO or SCAD penalty. It serves as an internal wrapper that applies the fit function over a grid of tuning parameter values.

Usage

```
PLR.fit(y, x, weights = NULL, penalty, grid.arg, grid.value, lambda.list, ...)
```

Arguments

y	A numeric vector representing the response variable.
x	A numeric matrix of covariates.
weights	An optional numeric vector of sample weights. Default is NULL.
penalty	A character string specifying the penalty type. Possible values are "LASSO" and "SCAD".
grid.arg	A character string specifying the tuning parameter for which a grid is constructed.
grid.value	A numeric vector specifying the grid values for grid.arg. If NULL, no grid is constructed.
lambda.list	An optional list specifying penalty values (λ) to be used for each grid value.
...	Additional arguments passed to Lorenz.FABS or Lorenz.SCADFABS, depending on the penalty type.

Details

The function applies either Lorenz.FABS (for LASSO) or Lorenz.SCADFABS (for SCAD) for each grid value. The best model is selected based on the BIC score.

Value

A list containing:

`path` A list of matrices, where each element corresponds to a grid value. Each matrix contains lambda values, Lorenz-R^2, explained Gini coefficients, BIC scores, and estimated coefficients.

`grid.idx` The index of the optimal grid parameter selected by the BIC criterion.

`lambda.idx` The index of the optimal λ selected by the BIC criterion.

`grid.value` The grid values used for `grid.arg`.

`lambda.list` A list of λ values along the solution paths.

`grid.arg` The tuning parameter for which the grid was constructed.

See Also

Lorenz.FABS, Lorenz.SCADFABS, Lorenz.boot, Lorenz.Reg

Examples

```
data(Data.Incomes)
y <- Data.Incomes$Income
x <- as.matrix(Data.Incomes[,-c(1,2)])
PLR.fit(y, x, penalty = "SCAD", grid.arg = "eps", grid.value = c(0.2,0.5), lambda.list = NULL)
```

---

predict.LR *Prediction and fitted values for the Lorenz regression*

Description

`predict` provides predictions for an object of class "LR", while `fitted` extracts the fitted values.

Usage

```
## S3 method for class 'LR'
predict(object, newdata, type = c("index", "response"), ...)

## S3 method for class 'LR'
fitted(object, type = c("index", "response"), ...)
```

Arguments

object	An object of class "LR".
newdata	An optional data frame in which to look for variables with which to predict. If omitted, the original data are used.
type	A character string indicating the type of prediction or fitted values. Possible values are "response" and "index" (the default). In the first case, the prediction estimates the conditional expectation of the response given the covariates. In the second case, the prediction estimates only the index of the single-index model.
...	Additional arguments passed to the function Rearrangement.estimation.

Details

The type argument distinguishes between two types of prediction outputs, aligned with the goals of the Lorenz regression. When type = "index", the function returns the estimated index $X^\top \theta$ of the single-index model. This index captures the full ordering structure of the conditional expectation and is sufficient for computing the explained Gini coefficient, which is the primary focus of the method. Crucially, this estimation does not require recovering the full nonparametric link function. When type = "response", the function estimates the full conditional expectation $\mathbb{E}[Y|X]$ by performing a second-stage estimation of the link function via Rearrangement.estimation. This is useful if fitted or predicted response values are needed for other purposes.

Value

A vector of predictions for predict, or a vector of fitted values for fitted.

See Also

Lorenz.Reg, Rearrangement.estimation

Examples

```
## For examples see example(Lorenz.Reg) and example(Lorenz.boot)
```

---

predict.PLR *Prediction and fitted values for the penalized Lorenz regression*

Description

predict provides predictions for an object of class "PLR", while fitted extracts the fitted values.

Usage

```
## S3 method for class 'PLR'
predict(object, newdata, type = c("index", "response"), pars.idx = "BIC", ...)

## S3 method for class 'PLR'
fitted(object, type = c("index", "response"), pars.idx = "BIC", ...)
```

Arguments

`object`	An object of S3 class `"PLR"`. The object might also have S3 classes `"PLR_boot"` and/or `"PLR_cv"` (both inherit from class `"PLR"`)
`newdata`	An optional data frame in which to look for variables with which to predict. If omitted, the original data are used.
`type`	A character string indicating the type of prediction or fitted values. Possible values are `"index"` (the default) and response. In the first case, only the index of the single-index model is estimated. In the second case, the "full" conditional expectation of the response given the covariates is estimated.
`pars.idx`	What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are: • `"BIC"` (default) - Always available. • `"Boot"` - Available if `object` inherits from `"PLR_boot"`. • `"CV"` - Available if `object` inherits from `"PLR_cv"`. Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.
`...`	Additional arguments passed to the function `Rearrangement.estimation`.

Details

The `type` argument distinguishes between two types of prediction outputs, aligned with the goals of the penalized Lorenz regression. When `type = "index"`, the function returns the estimated index $X^\top \theta$ of the single-index model. This index captures the full ordering structure of the conditional expectation and is sufficient for computing the explained Gini coefficient, which is the primary focus of the method. Crucially, this estimation does not require recovering the full nonparametric link function. When `type = "response"`, the function estimates the full conditional expectation $\mathbb{E}[Y|X]$ by performing a second-stage estimation of the link function via `Rearrangement.estimation`. This is useful if fitted or predicted response values are needed for other purposes.

Value

A vector of predictions for `predict`, or a vector of fitted values for `fitted`.

See Also

`Lorenz.Reg`, `Rearrangement.estimation`

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

print.LR *Printing method for the Lorenz regression*

Description

Prints the arguments, explained Gini coefficient and estimated coefficients of an object of class "LR".

Usage

```
## S3 method for class 'LR'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class "LR".
digits	The number of significant digits to be passed.
...	Additional arguments.

Value

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

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg)
```

print.PLR *Printing method for the penalized Lorenz regression*

Description

Prints the arguments, explained Gini coefficient and estimated coefficients of an object of class "PLR".

Usage

```
## S3 method for class 'PLR'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of S3 class "PLR". The object might also have S3 classes "PLR_boot" and/or "PLR_cv" (both inherit from class "PLR")
digits	The number of significant digits to be passed.
...	Additional arguments.

Details

The explained Gini coefficient and estimated coefficients are returned for each available selection method, depending on the class of x.

Value

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

See Also

Lorenz.Reg

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

---

print.summary.LR *Printing method for the summary of a Lorenz regression*

Description

Provides a printing method for an object of class "summary.LR".

Usage

```
## S3 method for class 'summary.LR'
print(x, digits = max(3L, getOption("digits") - 3L), ...)

## S3 method for class 'summary.LR_boot'
print(
  x,
  digits = max(3L, getOption("digits") - 3L),
  signif.stars = getOption("show.signif.stars"),
  ...
)
```

Arguments

x	An object of class "summary.LR". The object might also have S3 class "summary.LR_boot" (which inherits from class "summary.LR")
digits	Number of significant digits to be passed.
...	Additional arguments passed to the function print.
signif.stars	Logical determining whether p-values should be also encoded visually. See the help of the function printCoefmat for more information. This is only relevant if x inherits from "summary.LR_boot".

Value

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

See Also

summary.LR

Examples

```
## For examples see example(Lorenz.Reg) and example(Lorenz.boot)
```

---

print.summary.PLR *Printing method for the summary of a penalized Lorenz regression*

Description

Provides a printing method for an object of class "summary.PLR".

Usage

```
## S3 method for class 'summary.PLR'
print(x, digits = max(3L, getOption("digits") - 3L), ...)
```

Arguments

x	An object of class "summary.PLR". The object might also have S3 class "summary.PLR_boot" and/or "summary.PLR_cv" (both inherit from class "summary.LR")
digits	Number of significant digits to be passed.
...	Additional arguments passed to the function print.

Value

No return value, called for printing an object of class "summary.PLR" to the console.

See Also

summary.PLR

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

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,
  method = "loess",
  ...
)
```

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	A vector of sample weights. By default, each observation is given the same weight.
method	Either a character string specifying a smoothing method (e.g., "loess", the default), or a list containing two elements: • fit_fun: a function used to fit the model, typically taking arguments y, x, and optionally weights, and returning a fitted object. • predict_fun: a function to generate predictions from the fitted object. It must accept the fitted object and an argument newdata. The specification of a custom method is illustrated in the Examples section. If a character string is provided, a predict method must exist for that function (e.g., predict.loess). If weights are provided but unsupported by the fit function, a warning is issued and the weights are ignored.
...	Additional arguments passed to the fit function defined by method.

Details

A first estimator of the link function, neglecting the assumption of monotonicity, is obtained using the procedure chosen via `method`. 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`, `rearrangement`

Examples

```
data(Data.Incomes)
PLR <- Lorenz.Reg(Income ~ ., data = Data.Incomes,
                  penalty = "SCAD", eps = 0.01)
y <- PLR$y
index <- predict(PLR)
# Default method where the first step is obtained with loess()
Rearrangement.estimation(y = y, index = index, method = "loess")
# Custom method, where the first step is obtained with ksmooth()
# ksmooth() lacks from a separation between fitting and predicting interfaces
ksmooth_method <- list(
  fit_fun = function(y, x, ...) {
    list(y = y, x = x, args = list(...))
  },
  predict_fun = function(fit, newdata) {
    if(missing(newdata)){
      x.points <- fit$x
    } else {
      x.points <- newdata[,1]
    }
    o <- order(order(x.points))
   yhat <- do.call(ksmooth, c(list(x = fit$x, y = fit$y, x.points = x.points), fit$args))$y
    yhat[o]
  }
)
Rearrangement.estimation(y = y, index = index, method = ksmooth_method, bandwidth = 0.1)
```

residuals.LR *Residuals for the Lorenz regression*

Description

`residuals` provides residuals for an object of class `"LR"`.

Usage

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

Arguments

`object`	An object of class `"LR"`.
`...`	Additional arguments passed to the function `Rearrangement.estimation`.

Details

Computing residuals entail to estimate the link function of the single-index model. This is done via the function `Rearrangement.estimation`.

Value

A vector of residuals.

See Also

`Lorenz.Reg`, `Rearrangement.estimation`

Examples

```
## For examples see example(Lorenz.Reg) and example(Lorenz.boot)
```

residuals.PLR *Residuals for the penalized Lorenz regression*

Description

`residuals` provides residuals for an object of class `"PLR"`.

Usage

```
## S3 method for class 'PLR'
residuals(object, pars.idx = "BIC", ...)
```

Arguments

object	An object of class "PLR".
pars.idx	What grid and penalty parameters should be used for parameter selection. Either a character string specifying the selection method, where the possible values are: • "BIC" (default) - Always available. • "Boot" - Available if object inherits from "PLR_boot". • "CV" - Available if object inherits from "PLR_cv". Or a numeric vector of length 2, where the first element is the index of the grid parameter and the second is the index of the penalty parameter.
...	Additional arguments passed to the function Rearrangement.estimation.

Details

Computing residuals entail to estimate the link function of the single-index model. This is done via the function Rearrangement.estimation.

Value

A vector of residuals.

See Also

Lorenz.Reg, Rearrangement.estimation

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

---

summary.LR *Summary for the Lorenz regression*

Description

Provides a summary for an object of class "LR".

Usage

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

Arguments

object	An object of class "LR". The object might also have S3 class "LR_boot" (which inherits from class "PLR").
...	Additional arguments.

Details

The inference provided in the `coefficients` matrix is obtained by using the asymptotic normality and estimating the asymptotic variance via bootstrap.

Value

An object of class `"summary.LR"`, containing the following elements:

`call` The matched call.

`ineq` A matrix with one row and three columns providing information on explained inequality. The first column gives the explained Gini coefficient, the second column gives the Gini coefficient of the response. The third column gives the Lorenz-R^2.

`coefficients` A matrix providing information on the estimated coefficients. The first column gives the estimates. If `object` inherits from `"LR_boot"`, bootstrap inference was performed and the matrix contains further information. The second column is the boostrap standard error. The third column is the z-value. Finally, the last column is the p-value. In this case, the class `"summary.LR_boot"` is added to the output.

See Also

`Lorenz.Reg`, `Lorenz.boot`

Examples

```
## For examples see example(Lorenz.Reg) and example(Lorenz.boot)
```

---

summary.PLR *Summary for the penalized Lorenz regression*

Description

Provides a summary for an object of class `"PLR"`.

Usage

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

Arguments

`object`	An object of class `"PLR"`. The object might also have S3 classes `"PLR_boot"` and/or `"PLR_cv"` (both inherit from class `"PLR"`)
`renormalize`	A logical value determining 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

An object of class `"summary.PLR"`, which contains:

`call` The matched call.

`ineq` A table of explained inequality metrics. The columns display the explained Gini coefficient, the Gini coefficient of the response, and the Lorenz-R2. The first row contains the results obtained by BIC.

`coefficients` A matrix with estimated coefficients, each row corresponding to a specific coefficient. The first column contains the results obtained by BIC.

If the object inherits from `"PLR_boot"`, `ineq` and `coefficients` also include results from bootstrap, and the class `"summary.PLR_boot"` is added to the output. Similarly, if the object inherits from `"PLR_cv"`, `ineq` and `coefficients` also include results from cross-validation, and the class `"summary.PLR_cv"` is added to the output.

See Also

`Lorenz.Reg`, `Lorenz.boot`, `PLR.CV`

Examples

```
## For examples see example(Lorenz.Reg), example(Lorenz.boot) and example(PLR.CV)
```

Index