Package 'relaimpo'

October 4, 2023

Title Relative Importance of Regressors in Linear Models

Version 2.2-7

Date 2023-10-04

Description Provides several metrics for assessing relative importance in linear models. These can be printed, plotted and bootstrapped. The recommended metric is lmg, which provides a decomposition of the model explained variance into non-negative contributions. There is a version of this package available that additionally provides a new and also recommended metric called pmvd. If you are a non-US user, you can download this extended version from Ulrike Groempings web site.

Depends R(>= 2.2.1), MASS, boot, survey, mitools, graphics

Imports methods, corpcor

Suggests care

Maintainer Ulrike Groemping <ulrike.groemping@bht-berlin.de>

License GPL-2

LazyLoad yes

URL https://prof.bht-berlin.de/groemping/relaimpo/,
https://prof.bht-berlin.de/groemping/

NeedsCompilation no

Author Ulrike Groemping [aut, cre],
Matthias Lehrkamp [ctb]

Repository CRAN

Date/Publication 2023-10-04 15:10:02 UTC

R topics documented:

Index   **35**

---

relaimpo-package   *Package to calculate relative importance metrics for linear models*

Description

relaimpo calculates several relative importance metrics for the linear model. The recommended metrics are lmg (R^2 partitioned by averaging over orders, like in Lindemann, Merenda and Gold (1980, p.119ff)) and pmvd (a newly proposed metric by Feldman (2005), non-US version only). For completeness, several other metrics are also on offer. Other packages with related topics: hier.part, relimp.

Details

relaimpo calculates the metrics and also offers the possibility of bootstrapping them and of displaying results in print and graphically.

It is possible to designate a subset of variables as adjustment variables that always stay in the model so that relative importance is only assessed among the remaining variables.

Models can have up to 2-way interactions that are treated hierarchically - i.e. an interaction is only allowed in a model that also contains all its main effects. In case of interactions, only metric lmg can be used.

Observations with missing values are by default excluded from the analysis for most functions. The function mianalyze.relimp allows to draw conclusions from a set of multiply imputed data sets. This function is currently more restrictive than the rest of the package in terms of data types and models that can be used (when summarizing the multiply imputed samples without calculating confidence intervals, all possibilities available elsewhere are also applicable in mianalyze.relimp).

relaimpo does accomodate complex survey designs by making use of the facilities in package **survey**. Currently, interactions and calculated variables cannot be combined with using a complex survey design in bootstrapping functions.

Acknowlegment

This package uses as an internal function the function nchoosek from **vsn**, authored by Wolfgang Huber, available under LGPL. Furthermore, it uses a modified version of the function carscore from **care** by Verena Zuber and Korbinian Strimmer.

Warning

lmg and pmvd are computer-intensive. Although they are calculated based on the covariance matrix, which saves substantial computing time in comparison to carrying out actual regressions, these methods still take quite long for problems with many regressors. Obviously, this is particularly relevant in combination with bootstrapping.

Note

There are two versions of this package. The version on CRAN is globally licensed under GPL version 2 (or later). There is an extended version with the interesting additional metric pmvd that is licensed according to GPL version 2 under the geographical restriction "outside of the US" because of potential issues with US patent 6,640,204. This version can be obtained from Ulrike Groempings website (cf. references section). Whenever you load the package, a display tells you, which version you are loading.

Author(s)

Ulrike Groemping, BHT Berlin

References

Chevan, A. and Sutherland, M. (1991) Hierarchical Partitioning. *The American Statistician* **45**, 90–96.

Darlington, R.B. (1968) Multiple regression in psychological research and practice. *Psychological Bulletin* **69**, 161–182.

Feldman, B. (2005) Relative Importance and Value. Manuscript (Version 1.1, March 19 2005), downloadable at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2255827

Genizi, A. (1993) Decomposition of R2 in multiple regression with correlated regressors. *Statistica Sinica* **3**, 407–420. Downloadable at https://www3.stat.sinica.edu.tw/statistica/password.asp?vol=3&num=2&art=10

Groemping, U. (2006) Relative Importance for Linear Regression in R: The Package relaimpo *Journal of Statistical Software* **17**, Issue 1. Downloadable at https://www.jstatsoft.org/v17/i01

Lindeman, R.H., Merenda, P.F. and Gold, R.Z. (1980) *Introduction to Bivariate and Multivariate Analysis*, Glenview IL: Scott, Foresman.

Zuber, V. and Strimmer, K. (2010) *Variable importance and model selection by decorrelation*. Preprint, downloadable at https://arxiv.org/abs/1007.5516

Go to https://prof.bht-berlin.de/groemping/relaimpo/ for further information and references.

See Also

calc.relimp, booteval.relimp, mianalyze.relimp, classesmethods.relaimpo, package **hier.part**, package **survey**

---

booteval.relimp   *Functions to Bootstrap Relative Importance Metrics*

Description

These functions provide bootstrap confidence intervals for relative importances. `boot.relimp` uses the R package boot to do the actual bootstrapping of requested metrics (which may take quite a while), while `booteval.relimp` evaluates the results and provides confidence intervals. Output from `booteval.relimp` is printed with a tailored print method, and a plot method produces bar plots with confidence indication of the relative importance metrics.

Usage

```
## generic function
boot.relimp(object, ...)

## default S3 method
## Default S3 method:
boot.relimp(object, x = NULL, ..., b = 1000, type = "lmg",
    rank = TRUE, diff = TRUE, rela = FALSE, always = NULL,
    groups = NULL, groupnames = NULL, fixed=FALSE,
    weights = NULL, design = NULL)

## S3 method for formula objects
## S3 method for class 'formula'
boot.relimp(formula, data, weights, na.action, ..., subset = NULL)

## S3 method for objects of class lm
## S3 method for class 'lm'
boot.relimp(object, type = "lmg", groups = NULL, groupnames=NULL, always = NULL,
    ..., b=1000)

## function for evaluating bootstrap results
booteval.relimp(bootrun, bty = "perc", level = 0.95,
    sort = FALSE, norank = FALSE, nodiff = FALSE,
    typesel = c("lmg", "pmvd", "last", "first", "betasq", "pratt", "genizi", "car"))
```

Arguments

`object`	cf. `calc.relimp`.
`formula`	cf. `calc.relimp`. But note the additional restriction that - in connection with the `design`=-option - it is currently not possible to use factors, interactions or calculated quantities in a formula.
`x`	cf. `calc.relimp`.
`b`	is the number of bootstrap runs requested on boot.relimp (default: b=1000). Make sure to set this to a lower number, if you are simply testing code.
`type`	cf. `calc.relimp`.
`rank`	is a logical requesting bootstrapping of ranks (rank=TRUE, default) for each metric from type
`diff`	is a logical requesting bootstrapping of pairwise differences in relative importance (diff=TRUE, default) for each metric in type

rela	cf. calc.relimp.
always	cf. calc.relimp.
groups	cf. calc.relimp.
groupnames	cf. calc.relimp.
weights	cf. calc.relimp for specification of weights. See also the Details section of this help page for usage of different types of weights.
design	cf. calc.relimp. But note that there are currently some restrictions regarding usability of other possibilities, when using a design in boot.relimp: formulae can only be simpler than usual, and factors, interactions or calculated variables in formulae are not permitted.
	For a description of the bootstrap method's treatment of designs, see the details section. In the current version, using a design in bootstrapping must be considered EXPERIMENTAL.
fixed	is a logical requesting bootstrapping for a fixed design matrix (if TRUE). The default is bootstrapping for randomly drawn samples (fixed = FALSE).
data	cf. calc.relimp.
subset	cf. calc.relimp.
na.action	cf. calc.relimp.
...	usable for further arguments, particularly all arguments of the default method can be given to all other methods
bootrun	is an object of class relimplmboot created by function boot.relimp. It hands over all relevant information on the bootstrap runs to function booteval.relimp.
bty	is the type of bootstrap interval requested (a character string), as handed over to the function boot.ci from package boot. Possible choices are bca, perc, basic and norm. student is not supported.
level	is a single confidence level or a numeric vector of confidence levels.
sort	is a logical requesting output sorted by size of relative contribution (sort=TRUE) or by variable position in list (sort=FALSE, default).
norank	is a logical that suppresses of rank letters (norank=TRUE) even if ranks have been bootstrapped.
nodiff	is a logical that suppresses output of confidence intervals for differences (nodiff=TRUE), even if differences have been bootstrapped.
typesel	provides the metrics that are to be reported. Default: all available ones (intersection of those available in object bootrun and those requested in typesel). typesel accepts the same values as type.

Details

Calculations of metrics are based on the function calc.relimp. Bootstrapping is done with the R package boot, resampling the full observation vectors by default (combinations of response, weights and regressors, cf. Fox (2002)). If fixed=TRUE is specified, bootstrapping is based on residuals rather than full observations, keeping the X-variables fixed.

If the `weights` option is used, weights are resampled together with the full observations, and weighted contributions are calculated for each resample (no re-normalization is done within the resamples.)

Please note that usage of weights in linear models can have very different reasons. One motivation is a different variability of different observations, where weights are the inverse variances. This is the way `weights` are treated in function `lm` and also in `calc.relimp` and in `boot.relimp`, if a vector of weights is given with the `weights` option. Specifically, weights do not affect the resampling probability in bootstrapping, i.e. each observation has the same probability to be included in resamples.

If the weights in a data frame represent the multiplicity of each observation (i.e. there are several units with identical combination of values in the data, and the weights represent the number of units with exactly this value pattern for each row of the data frame), they can also be directly used as weights in `calc.relimp` for calculating the metrics. However, such frequency weights cannot be appropriately accomodated in `boot.relimp`; instead, the data frame with frequencies has to be expanded to include one row for each unit before using the resampling routine (e.g. using function `untable` from package **reshape** or function `expand.table` from package **epitools**.

In survey situations, weights are used to generate a more representative picture of the population: an observation's weight is typically the number of units of the population that this single observed unit represents. In this situation, there is no reason to consider observations with higher weights as less variable than observations with lower weights; thus, while estimates can again be obtained treating the weights in the same way as mentioned before, their usage in estimation of standard errors and in bootstrapping is different. In order to appropriately accomodate survey weights for these purposes, it is not sufficient to only provide the weights vector; instead, it is necessary to provide a design generated with package **survey** or an object of class svyglm (produced by function svyglm) that includes the appropriate design information.

Clusters are a way to take care of dependency structures like in longitudinal data. Thus, while relaimpo does not (currently) cover linear mixed models (e.g. produced by function lme), it is possible to accomodate clustered data by applying function svyglm with linear link function and gaussian distribution to a design that contains clusters. The bootstrapping approach subsequently takes care of the dependence by considering clusters as sampling units. Users who want to use this approach can mimic the second example below.

If the `design` option is used (experimental), resampling is done within package **survey**, and the resampled contributions are also calculated within package survey. The results from these calculations are plugged into an object from package **boot**, and confidence interval calculation is subsequently handled in **boot**. The approach is considered experimental: so far no simulation studies have been conducted for complex survey designs, and because of limited experience (in spite of thorough testing) it is not unlikely that bugs will be found by users who are routinely using complex survey designs.

The output provides results for all selected relative importance metrics. The output object can be printed and plotted (description of syntax: classesmethods.relaimpo).

Printed output: In addition to the standard output of calc.relimp (one row for each regressor, one column for each bootstrapped metric), there is a table of confidence intervals for each selected metric (one row per combination of regressor and metric). This table is enhanced by information on rank confidence intervals, if ranks have been bootstrapped (rank=TRUE) and norank=FALSE. In addition, if differences have been bootstrapped (diff=TRUE) and nodiff=FALSE, there is a table of estimated pairwise differences with confidence intervals.

Graphical output: Application of the plot method to the object created by `booteval.relimp` yields barplot representations for all bootstrapped metrics (all in one graphics window). Confidence level (`lev=`) and number of characters in variable names to be used (`names.abbrev=`) can be modified. Confidence bounds are indicated on the graphs by added vertical lines. `par()` options can be used for modifying output (exceptions: `mfrow`, `oma` and `mar` are overridden by the plot method).

Value

The value of `boot.relimp` is of class `relimplmboot`. It is designed to be useful as input for `booteval.relimp` and is not further described here. `booteval.relimp` returns an object of class `relimplmbooteval`, the items of which can be accessed by the `$` or the `@` extractors.

In addition to the items described for function `calc.relimp`, which are also available here, the following items may be of interest for further calculations:

metric.lower	matrix of lower confidence bounds for "metric": one row for each confidence level, one column for each element of "metric". "metric" can be any of `lmg`, `lmg.rank`, `lmg.diff`, ... (replace `lmg` with other available relative importance metrics, cf. `calc.relimp`)
metric.upper	matrix of upper confidence bounds for "metric": one row for each confidence level, one column for each element of "metric"
metric.boot	matrix of bootstrap results for "metric": one row for each bootstrap run, one column for each element of "metric". Here, "metric" can be chosen as any of the above-mentioned and also as R^2
nboot	number of bootstrap runs underlying the evaluations
level	confidence levels

Warning

The bootstrap confidence intervals should be used for exploratory purposes only. They can be somewhat liberal: Limited simulations for percentile intervals have shown that non-coverage probabilities can be up to twice the nominal probabilities. More investigations are needed.

Be aware that the method itself needs some computing time in case of many regressors. Hence, bootstrapping should be used with awareness of computing time issues.

`relaimpo` is a package for univariate linear models. Using `relaimpo` on objects that inherit from class `lm` but are not univariate linear model objects may produce nonsensical results without warning. Objects of class `mlm` or `glm` with link functions other than identity or family other than gaussian lead to an error message.

Note

There are two versions of this package. The version on CRAN is globally licensed under GPL version 2 (or later). There is an extended version with the interesting additional metric `pmvd` that is licensed according to GPL version 2 under the geographical restriction "outside of the US" because of potential issues with US patent 6,640,204. This version can be obtained from Ulrike Groempings website (cf. references section). Whenever you load the package, a display tells you, which version you are loading.

Author(s)

Ulrike Groemping, BHT Berlin

References

Chevan, A. and Sutherland, M. (1991) Hierarchical Partitioning. *The American Statistician* **45**, 90–96.

Darlington, R.B. (1968) Multiple regression in psychological research and practice. *Psychological Bulletin* **69**, 161–182.

Feldman, B. (2005) Relative Importance and Value. Manuscript (Version 1.1, March 19 2005), downloadable at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2255827

Fox, J. (2002) Bootstrapping regression models. *An R and S-PLUS Companion to Applied Regression: A web appendix to the book.* https://socialsciences.mcmaster.ca/jfox/Books/Companion-2E/appendix/Appendix-Bootstrapping.pdf.

Genizi, A. (1993) Decomposition of R2 in multiple regression with correlated regressors. *Statistica Sinica* **3**, 407–420. Downloadable at https://www3.stat.sinica.edu.tw/statistica/password.asp?vol=3&num=2&art=10

Groemping, U. (2006) Relative Importance for Linear Regression in R: The Package relaimpo *Journal of Statistical Software* **17**, Issue 1. Downloadable at https://www.jstatsoft.org/v17/i01

Lindeman, R.H., Merenda, P.F. and Gold, R.Z. (1980) *Introduction to Bivariate and Multivariate Analysis*, Glenview IL: Scott, Foresman.

Zuber, V. and Strimmer, K. (2010) *Variable importance and model selection by decorrelation.* Preprint, downloadable at https://arxiv.org/abs/1007.5516

Go to https://prof.bht-berlin.de/groemping/relaimpo/ for further information and references.

See Also

relaimpo, calc.relimp, mianalyze.relimp, classesmethods.relimp

Examples

```
#######################################################################
### Example: relative importance of various socioeconomic indicators
###            for Fertility in Switzerland
### Fertility is first column of data set swiss
#######################################################################
data(swiss)
   # bootstrapping
      bootswiss <- boot.relimp(swiss, b = 100,
               type = c("lmg", "last", "first", "pratt"),
               rank = TRUE, diff = TRUE, rela = TRUE)
      # for demonstration purposes only 100 bootstrap replications

      #alternatively, use formula interface
      bootsub <- boot.relimp(Fertility~Education+Catholic+Infant.Mortality, swiss,
            subset=Catholic>40, b = 100, type = c("lmg", "last", "first", "pratt"),
```

```
              rank = TRUE, diff = TRUE)
          # for demonstration purposes only 100 bootstrap replications

    #default output (percentily intervals, as of Version 2 of the package)
     booteval.relimp(bootswiss)
     plot(booteval.relimp(bootswiss))

     #sorted printout, chosen confidence levels, chosen interval method
     #store as object
         result <- booteval.relimp(bootsub, bty="bca",
               sort = TRUE, level=c(0.8,0.9))
          #because of only 100 bootstrap replications,
          #default bca intervals produce warnings
     #output driven by print method
         result
     #result plotting with default settings
     #(largest confidence level, names abbreviated to length 4)
         plot(result)
     #result plotting with modified settings (chosen confidence level,
     #names abbreviated to chosen length)
         plot(result, level=0.8,names.abbrev=5)
     #result plotting with longer names shown vertically
         par(las=2)
         plot(result, level=0.9,names.abbrev=6)
     #plot does react to options set with par()
     #exceptions: mfrow, mar and oma are set within the plot routine itself

########################################################################
### Example: bootstrapping clustered data
###          data taken from example.lmm of package lmm
### y is change in pulse (heart beats per minute)
###     15 min (occ 1 to 3) and 90 min (occ 4 to 6) after
###     treatment with Placebo (occ 1 or 4) low (occ 2 or 5)
###     or high (occ 3 or 6) dose of marihuana
### each of 9 subjects is observed under most or all
### of the 6 possible conditions
########################################################################
## create example data from package lmm
y <- c(16,20,16,20,-6,-4,
    12,24,12,-6,4,-8,
    8,8,26,-4,4,8,
    20,8,20,-4,
    8,4,-8,22,-8,
    10,20,28,-20,-4,-4,
    4,28,24,12,8,18,
    -8,20,24,-3,8,-24,
    20,24,8,12)
occ <- c(1,2,3,4,5,6,
      1,2,3,4,5,6,
      1,2,3,4,5,6,
      1,2,5,6,
      1,2,3,5,6,
      1,2,3,4,5,6,
```

```
        1,2,3,4,5,6,
        1,2,3,4,5,6,
        2,3,4,5)
subj <- c(1,1,1,1,1,1,
          2,2,2,2,2,2,
          3,3,3,3,3,3,
          4,4,4,4,
          5,5,5,5,5,
          6,6,6,6,6,6,
          7,7,7,7,7,7,
          8,8,8,8,8,8,
          9,9,9,9)
### manual creation of dummies
### reference category placebo after 90min (occ=4)
dumm1 <- as.numeric(occ==1)
dumm2 <- as.numeric(occ==2)
dumm3 <- as.numeric(occ==3)
dumm5 <- as.numeric(occ==5)
dumm6 <- as.numeric(occ==6)

## create data frame
dat <- data.frame(y,dumm1,dumm2,dumm3,dumm5,dumm6,subj)

### create design with clusters
des <- svydesign(id=~subj,data=dat)

### apply bootstrapping
### using the design with subjects as clusters implies that
###      clusters are generally kept in or excluded as a whole
### of course, b=100 is too small, only chosen for speed of package creation
bt <- boot.relimp(y~dumm1+dumm2+dumm3+dumm5+dumm6,data=dat,
    design=des,b=100,type=c("lmg","first","last","betasq","pratt"))

### calculate and display results
booteval.relimp(bt,lev=0.9,nodiff=TRUE)
```

---

calc.relimp   *Function to calculate relative importance metrics for linear models*

Description

calc.relimp calculates several relative importance metrics for the linear model. The recommended metrics are lmg (R^2 partitioned by averaging over orders, like in Lindemann, Merenda and Gold (1980, p.119ff)) and pmvd (a newly proposed metric by Feldman (2005) that is provided in the non-US version of the package only). For completeness and comparison purposes, several other metrics are also on offer (cf. e.g. Darlington (1968)).

Usage

```
## generic function
calc.relimp(object, ...)

## default S3 method
## Default S3 method:
calc.relimp(object, x = NULL, ...,
       type = "lmg", diff = FALSE, rank = TRUE, rela = FALSE, always = NULL,
       groups = NULL, groupnames = NULL, weights=NULL, design=NULL)

## S3 method for formula object
## S3 method for class 'formula'
calc.relimp(formula, data, weights, na.action, ..., subset=NULL)

## S3 method for objects of class lm
## S3 method for class 'lm'
calc.relimp(object, type = "lmg", groups = NULL, groupnames=NULL, always = NULL, ...)
```

Arguments

object
: The class of this object determines which of the methods is used: There are special methods for output objects from function `lm` (or linear model objects inheriting from class lm generated by other functions like `glm` and svyglm) and for formula objects. For all other types of object, the default method is used.

  Thus, object can be

  a formula (e.g. y~x1+x2+x3+x2:x3) (cf. below for details)

  OR

  the output of a linear model call (inheriting from class lm, but not mlm); output objects from lm, glm, svyglm or aov work (if linear with identity link in case of glm's); there may be further functions that output objects inheriting from lm which may or may not work reasonably with calc.relimp; for calc.relimp to be appropriate, the underlying model must at least be linear!

  The restrictions on usage of interactions listed under item formula below also apply to linear model objects.

  OR

  the covariance matrix of a response y and regressors x, (e.g. obtained by cov(cbind(y,x)), if y is a column vector of response values and x a corresponding matrix of regressors)

  OR

  a (raw) data matrix or data frame with the response variable in the first column

  OR

  a response vector or one-column matrix, if x contains the corresponding matrix or data frame of regressors.

formula
: The first object, if a formula is to be given; one response only.

Interaction terms are currently limited to second-order.

Note: If several interaction terms are given, calculations may be very resource intensive, if these are all connected (e.g. with A:B, B:C, C:D, all A,B,C,D are connected, while with A:B, C:D, D:E there are separate groups A,B and C,D,E).

Interaction terms occurring in always do not increase resource usage (but are only permitted if the respective main effects also occur in always).

Interactions and groups currently cannot be used simultaneously.

x   a (raw) data matrix or data frame containing the regressors, if object is a response vector or one-column matrix

OR

NULL, if object is anything else

type   can be a character string, character vector or list of character strings. It is the collection of metrics that are to be calculated. Available metrics: lmg, pmvd (non-US version only), last, first, betasq, pratt, genizi and car. For brief sketches of their meaning cf. details section.

diff   logical; if TRUE, pairwise differences between the relative contributions are calculated; default FALSE

rank   logical; if TRUE, ranks of regressors in terms of relative contributions are calculated; default TRUE

rela   is a logical requesting relative importances summing to 100% (rela=TRUE). If rela is FALSE (default), some of the metrics sum to R^2 (lmg, pmvd, pratt), others do not have a meaningful sum (last, first, betasq).

always   is a vector of column numbers or names of variables to be always in the model (adjusted for). Valid numbers are 2 to (number of regressors + 1) (1 is reserved for the response), valid character strings are all column names of object or x respectively that refer to regressor variables. Numbers and names cannot be mixed.

Relative importance is only assessed for the variables not selected in always.

This option currently does not work for metrics genizi and car.

groups   is a list of vectors of column numbers or names of variables to be combined into groups. If only one group is needed, a vector can be given. The numbers and character strings needed are of the same form as for always.

Relative importance is only allocated between groups of regressors, no subdivision within groups is calculated. Regressors that do not occur in any group are included as singletons. A regressor must not occur in always and in groups. Also, groups cannot be used with a linear model or a formula in case of higher order effects (interactions). Finally, groups only works with the four metrics lmg, pmvd, last and first.

groupnames   is a vector of names for the variable groups to be used for annotation of output.

weights   is a vector of case weights for the observations in the data frame (or matrix). You can EITHER specify weights OR a design. Note that weights must not be specified for linear model objects (since these should contain their weights as part of the model).

design is a design object of class survey.design (cf. package **survey**). You can EI-THER specify a design OR weights. For calc.relimp, the design is used for calculating weights only. Note that it is discouraged (though possible) to spec-ify a design for a conventional linear model object (since a survey-specific linear model should be used for survey data, cf. function svyglm).

Also note that care is needed when using subset together with design: the subset-Option only treats the data handed directly to calc.relimp, the design has to be equivalently treated beforehand.

data if first object is of class formula: an optional matrix or data frame that the vari-ables in formula and subset come from; if it is omitted, all names must be mean-ingful in the environment from which calc.relimp is called

subset if first object is of class formula: an optional expression indicating the subset of the observations of data that should be used in the fit. This can be a logical vector, or a numeric vector indicating which observation numbers are to be in-cluded, or a character vector of the row names to be included. All (non-missing) observations are included by default.

na.action if first object is of class formula: an optional function that indicates what should happen when the data contain 'NA's. The default is first, any na.action attribute of data, second the setting given in the call to calc.relimp, third the na.action setting of options. Possible choices are "na.fail", (print an error message and terminate if there are any incomplete observations), "na.omit" or "na.exclude" (equivalent for package relaimpo, both analyse complete cases only and print a warning, this is also what is done the default method).

... usable for further arguments, particularly most arguments of default method can be given to all other methods (exception: weights and design cannot be given to lm-method)

Details

lmg is the R^2 contribution averaged over orderings among regressors, cf. e.g. Lindeman, Merenda and Gold 1980, p.119ff or Chevan and Sutherland (1991).

pmvd is the proportional marginal variance decomposition as proposed by Feldman (2005) (non-US version only). It can be interpreted as a weighted average over orderings among regressors, with data-dependent weights.

last is each variables contribution when included last, also sometimes called usefulness.

first is each variables contribution when included first, which is just the squared covariance be-tween y and the variable.

betasq is the squared standardized coefficient.

pratt is the product of the standardized coefficient and the correlation.

genizi is the R^2 decomposition according to Genizi 1993

car is the R^2 decomposition according to Zuber and Strimmer 2010, also available from package **care** (squares of scores produced by function carscore

Each metric is calculated using the internal function "metric"calc, e.g. lmgcalc.

Five of the metrics in calc.relimp (lmg, pmvd, pratt, genizi and car), decompose the model R^2. calc.relimp (lmg, pmvd, pratt, genizi and car) sum to the R^2 that is to be decomposed, if rela = FALSE and to 100pct if rela = TRUE.

The other metrics also (artificially) sum to 100pct if rela = TRUE. If rela = FALSE, they are given relative to var(y) (or the conditional variance of y after adjusting out the variables requested in always) but do not sum to R^2.

If always requests some variables to be always in the model, these are conditioned upon (i.e. included into the model first). Only the remaining R^2 that is not explained by these variables is decomposed among the other regressors. This currently does not work for metrics genizi and car.

Four of the metrics, lmg, pmvd, first and last, are related to the order in which the variables are included into the model. For these it is possible to consider the variables in groups that are always entered into the model together.

Note that **relaimpo** can only provide metric lmg for models with interactions (2-way interactions only). It averages only over those orders, for which the interactions enter the model after both their main effects.

Note that there are different types of weights, weights indicating the variability of the response (observations with a more variable responses receive a lower weight than those with a less variable response, like in the Aitken estimator), frequency weights indicating the number of observations with exactly the observed data pattern of the current observation, or weights indicating the number of population units represented by the current observation (inverse sampling probability, weights typically used in survey situations). All three types of weight alike can be handed to function calc.relimp using the weights= option. Note, however, that they have to be treated differently for bootstrapping (cf. [boot.relimp](#)).

Data from complex surveys can be treated by providing a survey design with design=-option. For calc.relimp, it is also sufficient to provide the weights derived from the design using the weights=-option.

calc.relimp cannot handle data with missing values directly. It applies complete-case analysis, i.e. drops all units with any missing values by default. While this can be appropriate, if there are only few missing values, data with more severe missingness issues need special treatment. Package **relaimpo** offers the function mianalyze.relimp that handles multiply-imputed datasets (that can be created by several other R-packages). Currently, possibilities in this function are limited due to the fact that it uses complex survey designs and bootstrapping which do not (yet) go together well with factors, interactions and calculated quantities in formulae.

Value

var.y	the variance of the response
R2	the coefficient of determination, R^2
R2.decomp	the part of the coefficient of determination that is decomposed among the variables under investigation
lmg	vector of relative contributions obtained from the lmg method, if lmg has been requested in type
lmg.diff	vector of pairwise differences between relative contributions obtained from the lmg method, if lmg has been requested in type and diff=TRUE
lmg.rank	rank of the regressors relative contributions obtained from the lmg method, if lmg has been requested in type and rank=TRUE
metric, metric.diff, metric.rank	
	analogous to lmg for other metrics

`ave.coeffs`	average coefficients for variables not not requested by always only for models of different sizes;
	note that coefficients refer to modeling residuals after adjusting out variables listed in always (both from response and other explanatory variables)
`namen`	names of variables, starting with response
`type`	character vector of metrics available
`rela`	Have metrics been normalized to sum 100% ?
`always`	column numbers of variables always in the model; in case of factors, the column numbers given here are not identical to those in the call to `calc.relimp`, but refer to the columns of the model matrix
`alwaysnam`	names of variables always in the model
`call`	contains the call that generated the object

Warning

`lmg` and `pmvd` are computer-intensive. Although they are calculated based on the covariance matrix, which saves substantial computing time in comparison to carrying out actual regressions, these methods still take quite long for problems with many regressors.

`relaimpo` is a package for univariate linear models. Using `relaimpo` on objects that inherit from class `lm` but are not univariate linear model objects may produce nonsensical results without warning. Objects of class `mlm` or `glm` with link functions other than identity or family other than gaussian lead to an error message.

Note

There are two versions of this package. The version on CRAN is globally licensed under GPL version 2 (or later). There is an extended version with the interesting additional metric `pmvd` that is licensed according to GPL version 2 under the geographical restriction "outside of the US" because of potential issues with US patent 6,640,204. This version can be obtained from Ulrike Groempings website (cf. references section). Whenever you load the package, a display tells you, which version you are loading.

Author(s)

Ulrike Groemping, BHT Berlin

References

Chevan, A. and Sutherland, M. (1991) Hierarchical Partitioning. *The American Statistician* **45**, 90–96.

Darlington, R.B. (1968) Multiple regression in psychological research and practice. *Psychological Bulletin* **69**, 161–182.

Feldman, B. (2005) Relative Importance and Value. Manuscript (Version 1.1, March 19 2005), downloadable at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2255827

Genizi, A. (1993) Decomposition of R2 in multiple regression with correlated regressors. *Statistica Sinica* **3**, 407–420. Downloadable at https://www3.stat.sinica.edu.tw/statistica/password.asp?vol=3&num=2&art=10

Groemping, U. (2006) Relative Importance for Linear Regression in R: The Package relaimpo *Journal of Statistical Software* **17**, Issue 1. Downloadable at https://www.jstatsoft.org/v17/i01

Lindeman, R.H., Merenda, P.F. and Gold, R.Z. (1980) *Introduction to Bivariate and Multivariate Analysis*, Glenview IL: Scott, Foresman.

Zuber, V. and Strimmer, K. (2010) *Variable importance and model selection by decorrelation.* Preprint, downloadable at https://arxiv.org/abs/1007.5516

Go to https://prof.bht-berlin.de/groemping/relaimpo/ for further information and references.

See Also

relaimpo, booteval.relimp, mianalyze.relimp, classesmethods.relaimpo

Examples

```
#######################################################################
### Example: relative importance of various socioeconomic indicators
###           for Fertility in Switzerland
### Fertility is first column of data set swiss
#######################################################################
data(swiss)
    calc.relimp(swiss,
       type = c("lmg", "last", "first", "betasq", "pratt", "genizi", "car") )
    # calculation of all available relative importance metrics
        # non-US version offers the additional metric "pmvd",
        # i.e. call would be
        # calc.relimp(cov(swiss),
        # type = c("lmg", "pmvd", "last", "first", "betasq, "pratt"),
        # rela = TRUE )
    ## same analysis with formula or lm method and a few modified options
   crf <- calc.relimp(Fertility~Agriculture+Examination+Education+Catholic+Infant.Mortality,swiss,
        subset = Catholic>40,
        type = c("lmg", "last", "first", "betasq", "pratt"), rela = TRUE )
    crf
   linmod <- lm(Fertility~Agriculture+Examination+Education+Catholic+Infant.Mortality,swiss)
    crlm <- calc.relimp(linmod,
       type = c("lmg", "last", "first", "betasq", "pratt", "genizi", "car"), rela = TRUE )
    plot(crlm)
    # bar plot of the relative importance metrics

    #of statistical interest in this context: correlation matrix
        cor(swiss)

    #demonstration of conditioning on one regressor using always
    calc.relimp(swiss,
       type = c("lmg", "last", "first", "betasq", "pratt"), rela = FALSE,
       always = "Education" )

    # using calc.relimp with grouping of two regressors
    # and weights (not reasonable here, purely for demo purposes)
    calc.relimp(swiss,
```

```
        type = c("lmg", "last", "first"), rela = FALSE,
        groups = c("Education","Examination"), weights = abs(-23:23) )

    # using calc.relimp with grouping of two regressors
    # and a design object (not reasonable here, purely for demo purposes)
    des <- svydesign(~1, data=swiss, weights=~abs(-23:23))
    calc.relimp(swiss,
        type = c("lmg", "last", "first"), rela = FALSE,
        groups = c("Education","Examination"), groupnames ="EduExam", design = des)

    # calc.relimp with factors (betasq and pratt not possible)
      # (calc.relimp would not be necessary here,
      # because the experiment is balanced)
    calc.relimp(1/time~poison+treat,data=poisons, rela = FALSE,
        type = c("lmg", "last", "first"))
    # including also the interaction (lmg possible only)
    calc.relimp(1/time~poison*treat,data=poisons, rela = FALSE)
```

```
classesmethods.relaimpo
```
Classes and Methods in Package relaimpo

Description

Output objects from package **relaimpo** have classes relimplm (output from calc.relimp), relimplmboot (output from boot.relimp), relimplmbooteval (output from booteval.relimp) or relimplmbootMI. For classes relimplm, relimplmbooteval and relimplmbootMI, there are methods for plotting and printing, usage of which is described below. For class relimplmbootMI, there is in addition a summary-method, which produces a less detailed output than the show / print - method. For classes relimplm, relimplmbooteval and relimplmbootMI, there is in addition a method for extracting slots of the class with $.

Usage

```
## S3 method for class 'relimplm'
print(x,...,show.coeffs = ifelse(any(c("lmg", "pmvd") %in% x@type)
    & is.null(x@always), TRUE, FALSE))

## S3 method for class 'relimplm'
plot(x,...,names.abbrev=4, ylim=NULL, main=NULL, cex.title=1.5)

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

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

## S3 method for class 'relimplmbooteval'
```

```
plot(x, ..., lev=max(x@level), names.abbrev=4, ylim=NULL,
    main=NULL, cex.title=1.5)

## S3 method for class 'relimplmbootMI'
plot(x, ..., lev=max(x@level), names.abbrev=4, ylim=NULL,
    main=NULL, cex.title=1.5)

## S3 method for class 'relimplmbootMI'
summary(object, ..., lev = max(object@level))
```

Arguments

x	x is an output object from package **relaimpo** of the required class
show.coeffs	show.coeffs, if set to FALSE, suppresses printing of averaged coefficients, which are otherwise printed, if lmg and/or pmvd are among the metrics in x and always is NULL. See details for a discussion of the averaged coefficients, if always is not NULL.
object	object is an object of class relimplmbootMI (output from function mianalyze.relimp)
...	further arguments to functions
names.abbrev	names.abbrev is an integer that provides the number of characters to which the bar labels are shortened (default: 4).
ylim	The plot routines try to use appropriate scaling. If adjustments are needed, ylim can be used like usually on plot.
main	The plot routine uses a default title based on the reponses name. If adjustments are desired, main can be used for specifying a different title. Note that only the first title is affected (in case of plot.relimpbooteval, there is also a sub title that cannot be changed).
cex.title	cex.title specifies the text size for the overall title. Thus, the par option cex.main can be used for specifying the size of individual plot titles.
lev	lev is a numeric that provides the confidence level to be plotted or displayed respectivly (default: maximum available confidence level; 0.5<lev<1).

Details

This documentation part describes S3 methods. In addition there are S4 methods for show which coincide with the S3 methods for print and an S4 method for coercing objects of relimplm to lists (of their numeric elements).

Print (and show) methods produce annotated output for calc.relimp, booteval.relimp, and mianalyze.relimp (or the objects produced by these functions). Since version 2.1, calc.relimp provides averaged coefficients for different sub model sizes (slot ave.coeffs of class relimplm), if metrics based on averaging over orderings (lmg and/or pmvd) are calculated. These are per default printed if the slot always of x is NULL. If some variables were forced into all models (non-NULL always), the averaged coefficients refer to the adjusted model after taking residuls from regressions on the always-columns of the X-matrix for both response and the other columns of the X-matrix. The reason is that these could be easily and cheaply implemented into the existing code and do

correspond to sub models relevant for `lmg` and `pmvd`. Users who are interested in these coefficients, can set option `show.coeffs=TRUE` in spite of non-NULL `always`.

The plot methods produce barplots of relative contributions, either of the metrics alone for output objects of class `relimplm` from function `calc.relimp`, or of the metrics with lines indicating confidence intervals for output objects of class `relimplmbooteval` from function `booteval.relimp` or `relimplmbootMI` from function `booteval.relimp`.

Most `par()` options can be set and should work on plot. Exceptions: mfrow, oma and mar are set by the plot function, depending on the number of metrics to plot and the amount of annotating text required.

The summary-method for class `relimplmbootMI` allows to quickly display brief output and to change the confidence level versus the level used in the original run (with interval bounds stored in the "metric".lower and "metric".upper slots and displayed by print and show methods).

Because of a defined S3-extraction method, slots of classes `relimplm`, `relimplmbooteval` and `relimplmbootMI` can be extracted not only with the @ extractor but also with $. Hence, output elements from functions `calc.relimp`, `booteval.relimp`, and `mianalyze.relimp` can be extracted as though the output objects were lists.

Note that there also is an an internally-used class relimplmtest that permits the internal function calc.relimp_default.intern to output further detail needed for usage from within other funtions.

Author(s)

Ulrike Groemping, BHT Berlin

References

Go to https://prof.bht-berlin.de/groemping/ for further information and references.

See Also

relaimpo, calc.relimp, booteval.relimp, mianalyze.relimp, relimplm-class, relimplmboot-class, relimplmbooteval-class, relimplmbootMI-class

---

mianalyze.relimp	*Function to do relative importance calculations based on multiply imputed datasets*

Description

The function mianalyze.relimp takes a list of imputed data frames (or matrices), calculates relative importance metrics for each of these and combines these metrics into overall estimates with estimated variances according to the method by Rubin (1987). The output object can be summarized, printed and plotted.

Usage

```
mianalyze.relimp(implist, level = 0.95, sort = FALSE, ..., b = 50, type = "lmg",
    diff = TRUE, no.CI = FALSE, rela = FALSE, always = NULL, groups = NULL,
    groupnames = NULL, deslist = NULL, bootlist.out = FALSE, formula = NULL,
    weights = NULL, strata=NULL, ids=NULL)
```

Arguments

implist	list of data frames or matrices containing multiply-imputed datasets, or object of class imputationList
	If no formula is given, the first column of each data frame/matrix is assumed to be the response variable, the other columns are regressors.
	If a list of designs is also given, the variables component of each design must consist of the necessary columns from the respective entry in implist; if no formula is given, the variables component of each design must coincide (except for the order of columns) with the respective entry in implist.
level	is a single confidence level (between 0.5 and 1)
sort	is a logical requesting output sorted by size of relative contribution (sort=TRUE) or by variable position in list (sort=FALSE, default).
...	Further arguments, currently none available
b	is the number of bootstrap runs requested on boot.relimp (default: b=50). Make sure to set this to a higher number, if you want to subsequently use the bootlist slot for calculating further confidence intervals with function booteval.relimp.
type	cf. calc.relimp.
diff	is a logical requesting bootstrapping of pairwise differences in relative importance (diff=TRUE, default) for each metric in type
no.CI	if set to TRUE, suppresses calculation of confidence intervals and only averages estimated metrics from all imputed data sets in implist. Currently, no.CI = TRUE is the only setting for which mianalyze.relimp works when using models with factors, groups or interactions.
rela	cf. calc.relimp.
always	cf. calc.relimp.
groups	cf. calc.relimp.
groupnames	cf. calc.relimp.
deslist	is a list of design object of class survey.design (cf. package survey). You can EITHER specify a deslist OR weights and/or strata and/or ids. Note that the design list must contain the same data objects (in the "variables" element) that are listed in implist, so that a lot of storage space is needed in case of large datasets.
	If deslist is not given, the function creates a list of designs using weights, strata, and ids information. Whenever the required designs are simple enough to be covered by specifying weights, strata, and ids, this is by far preferrable in terms of storage.

bootlist.out   If TRUE, the individual bootstrap results for each multiply imputed data set are stored in the bootlist slot of the output object (may be storage-intensive).

formula   cf. [boot.relimp](); NOTE: If no.CI = FALSE, i.e. confidence intervals are not suppressed, formula has to follow the same restrictions as mentioned under item design for [boot.relimp]() (no calculated variables, no interaction terms, no factors), since confidence interval calculations in mianalyze.relimp are design-based, even if no deslist-option is given.

weights   is a vector of case weights for the observations in the data frame (or matrix). You can EITHER specify weights OR a deslist. If weights is NULL, equal weights are assumed, unless otherwise specified in deslist. For the different types of weights and their appropriate treatment for obtaining confidence intervals, cf. the "Details" section of [boot.relimp]().

strata   is a strata request that will be handed to function svydesign for defining the strata in a survey design (to be given to mianalyze without the ~). You can EITHER specify strata OR a deslist. If strata is NULL, one stratum is assumed, unless otherwise specified in deslist.

ids   is an id-request that will be handed to function svydesign for defining the clusters in a survey design (to be given to mianalyze without the ~). You can EITHER specify ids OR a deslist. If ids is NULL, it is assumed that there are no clusters, unless otherwise specified in deslist.

Details

Multiple imputation is a contemporary method for handling data with a substantial missing value problem. It produces a number of completed data sets (e.g. 10) the inference from which is subsequently combined. The most frequently used way of combination is the one by Rubin: estimates from the different completed data sets are averaged, and the variance is estimated by combining the average over the estimated variances (within imputation variance) with an appropriately-scaled variance between estimates, and confidence intervals are obtained by using a t-distribution with appropriately chosen degrees of freedom.

The variance-covariance matrix of the vector of estimates for each individual completed data set is obtained from function withReplicates in package **survey** based on **survey**'s bootstrap replication weights. On request (bootlist.out=TRUE), the underlying bootstrap resamples are also stored in the bootlist-slot of the output object. In this case, list elements of the bootlist-slot are objects of class relimplmboot and can be processed by function booteval.relimp. This can help in getting an impression whether the overall aggregated confidence intervals are heavily distorted towards symmetry. If such sanity-checking is intended, the default value for b should be substantially increased.

Function mianalyze.relimp needs a list of multiply-imputed data sets or an object of class imputationList for input. Multiply imputed data sets can - within R - be obtained from various packages. Hints for creating lists of the form needed for mianalyze.relimp are given below for users of functions aregImpute, mice, and amelia. Users of packages **norm**, **cat**, **mix**, or **pan** (who have managed to operate these extremely uncomfortable packages) can of course also produce lists of imputed data sets (only less comfortably).

For an object imp of class mids obtained from function mice in package **mice**, the code

```
lapply(as.list(1:imp$m),function(obj) complete(imp,action=obj))
```

produces a list of multiply-imputed data sets as needed for function mianalyze.relimp. For an object f of class aregImpute produced by function aregImpute in package **Hmisc**,

lapply(as.list(1:f$m),function(obj) impute(imp,imputation=obj))

produces the required list of multiply-imputed data sets. For an object output produced by function amelia in package **Amelia**, the code

output[1:output$amelia.args$m]

produces the list of multiply-imputed data sets as needed for function mianalyze.relimp.

For multiple imputation, practice is in many cases ahead of theory; this is no different with function mianalyze.relimp. Users should note that the validity of confidence intervals has only been proven for likelihood-based analyses. Since the metrics calculated in **relaimpo** are not strictly likelihood-based, the confidence intervals from function mianalyze.relimp must be considered approximate and experimental.

Value

The value returned by function mianalyze.relimp is an object of class relimplmbootMI (if no.CI = FALSE, default) or an object of class relimplm (if no.CI=TRUE). It can be printed, plotted and summarized using special methods. For extracting its items, the @ or $ extractors can be used.

In addition to the items described for function [calc.relimp](), which are also available here, the following items from class relimplmbootMI may be of interest for further calculations:

metric.lower	matrix of lower confidence bounds for "metric": one row for each confidence level, one column for each element of "metric". "metric" can be any of lmg, lmg.rank, lmg.diff, ... (replace lmg with other available relative importance metrics, cf. [calc.relimp]())
metric.upper	matrix of upper confidence bounds for "metric": one row for each confidence level, one column for each element of "metric"
nboot	number of bootstrap runs underlying the evaluations
level	confidence level
MIresult	object of class MIresult that can be processed by the function summary.MIresult from package survey
bootlist	only available if bootlist.out=TRUE has been chosen; list of objects of class boot.relimp; each list element can be input to function [booteval.relimp]()

Warning

The confidence intervals produced here should be used for exploratory purposes only. They can be somewhat liberal and are likely to be too symmetric particularly for small data sets. The confidence intervals produced by function mianalyze.relimp need further research into their behaviour and are currently considered experimental.

Be aware that the methods themselves (lmg and even more pmvd) need some computing time in case of many regressors. Hence, bootstrapping of multiple data sets should be used with awareness of computing time issues.

Note

There are two versions of this package. The version on CRAN is globally licensed under GPL version 2 (or later). There is an extended version with the interesting additional metric pmvd that is licensed according to GPL version 2 under the geographical restriction "outside of the US" because of potential issues with US patent 6,640,204. This version can be obtained from Ulrike Groempings website (cf. references section). Whenever you load the package, a display tells you, which version you are loading.

Author(s)

Ulrike Groemping, BHT Berlin

References

Chevan, A. and Sutherland, M. (1991) Hierarchical Partitioning. *The American Statistician* **45**, 90–96.

Darlington, R.B. (1968) Multiple regression in psychological research and practice. *Psychological Bulletin* **69**, 161–182.

Feldman, B. (2005) Relative Importance and Value. Manuscript (Version 1.1, March 19 2005), downloadable at https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2255827

Genizi, A. (1993) Decomposition of R2 in multiple regression with correlated regressors. *Statistica Sinica* **3**, 407–420. Downloadable at https://www3.stat.sinica.edu.tw/statistica/password.asp?vol=3&num=2&art=10

Groemping, U. (2006) Relative Importance for Linear Regression in R: The Package relaimpo *Journal of Statistical Software* **17**, Issue 1. Downloadable at https://www.jstatsoft.org/v17/i01

Lindeman, R.H., Merenda, P.F. and Gold, R.Z. (1980) *Introduction to Bivariate and Multivariate Analysis*, Glenview IL: Scott, Foresman.

Little, R.J.A. and Rubin, D.B. (2002) *Statistical Analysis with Missing Data*, Wiley, New York.

Zuber, V. and Strimmer, K. (2010) *Variable importance and model selection by decorrelation*. Preprint, downloadable at https://arxiv.org/abs/1007.5516

Go to https://prof.bht-berlin.de/groemping/ for further information and references.

See Also

[relaimpo](), [calc.relimp](), [booteval.relimp](), [classesmethods.relaimpo]()

Examples

```
## smi contains a list of 5 imputed datasets (class imputationList) from package mitools
## (first element of smi is list of data frames)
## it is not a well-suited example for relative importance but easily available for demonstrating
##          multiple imputation-related functionality

data(smi)

## obtain averaged estimates only, without confidence intervals
## works with factors and interactions
```

```
 mianalyze.relimp(smi[[1]], formula = cistot ~ drkfre+sex+wave, no.CI = TRUE)
 ## for obtaining all individual estimates, use lapply:
smi.cr.list <- lapply(smi[[1]], function(obj) calc.relimp(cistot ~ drkfre+sex+wave, data=obj))
 ## display result for first individual imputed data set
 smi.cr.list[[1]]

 ## obtain confidence intervals,
## currently only usable for models without calculated variables, factors, groups, interactions

 ## call without using weights, strata, clusters or a design list
 mianalyze.relimp(smi[[1]], formula = cistot ~ mdrkfre+sex+wave)
 ## call using the id column (identical in all smi data sets) for cluster structure
 ident <- smi[[1]][[1]]$id
 mitest <- mianalyze.relimp(smi[[1]], formula = cistot ~ mdrkfre+sex+wave, ids=ident)
 mitest
     ## postprocess: look at intervals with different confidence level
     summary(mitest,lev=0.8)
 ## call with design list
deslist <- lapply(smi[[1]], function(obj) svydesign(~id,strata=~sex,weights=~cistot,data=obj))
 mitest <- mianalyze.relimp(smi[[1]], formula = cistot ~ mdrkfre+sex+wave, deslist=deslist,
       level=c(0.8))
 mitest
```

---

relimplm-class	*Class relimplm*

Description

This is the class of output objects from the function `calc.relimp` in package **relaimpo**. Its elements are described in the documentation of `calc.relimp`.

Objects from the Class

Objects should only be created by calls to the function `calc.relimp`.

Slots

`var.y`: Object of class "numeric"

`R2`: Object of class "numeric"

`R2.decomp`: Object of class "numeric"

`lmg`: Object of class "numeric"

`pmvd`: Object of class "numeric"

`first`: Object of class "numeric"

`last`: Object of class "numeric"

`betasq`: Object of class "numeric"

`pratt`: Object of class "numeric"

genizi: Object of class "numeric"

car: Object of class "numeric"

lmg.rank: Object of class "numeric"

pmvd.rank: Object of class "numeric"

first.rank: Object of class "numeric"

last.rank: Object of class "numeric"

betasq.rank: Object of class "numeric"

pratt.rank: Object of class "numeric"

genizi.rank: Object of class "numeric"

car.rank: Object of class "numeric"

lmg.diff: Object of class "numeric"

pmvd.diff: Object of class "numeric"

first.diff: Object of class "numeric"

last.diff: Object of class "numeric"

betasq.diff: Object of class "numeric"

pratt.diff: Object of class "numeric"

genizi.diff: Object of class "numeric"

car.diff: Object of class "numeric"

namen: Object of class "character": variable names, starting with name for response

nobs: Object of class "numeric": number of valid observations used in analysis (available only if calculations are based on raw data)

ave.coeffs: Object of class "matrix": average coefficients for models of different sizes

type: Object of class "character": metrics that have been calculated

rela: Object of class "logical": TRUE means that metrics have been normalized to sum 100pct

always: Object of class "numintnull": variables always in the model (adjusted for), in terms of columns in the model matrix (i.e. in case of factors in the model, there are more entries than in the call to function calc.relimp; first possible column is 2 (1 reserved for response))

alwaysnam: Object of class "charnull": variable names of variables always in model

groupdocu: Object of class "list": information on group definitions

call: Object of class "langnull": the call that created the object

Methods

This documentation section documents S4 methods only. There are also S3-methods for printing, plotting and extracting slots with $. For their usage, see classesmethods.relaimpo.

coerce S4-method: If an object of this class is coerced to list by as(object, "list"), the slots are output to the list in the following order: var.y, R2, lmg, lmg.rank, lmg.diff, next three in non-US version only: pmvd, pmvd.rank, pmvd.diff, last, last.rank, last.diff, ..., pratt, pratt.rank, pratt.diff, namen, type, rela, always, alwaysnam

show S4-method: identical to print

Author(s)

Ulrike Groemping, BHT Berlin

See Also

[calc.relimp](), [print.relimplm](), [plot.relimplm](), **[relaimpo]()**

---

relimplmboot-class   *Class relimplmboot*

Description

This is the class for output objects from function `boot.relimp`. It is needed as input to function `booteval.relimp`.

Objects from the Class

Objects should only be created by calls to the function `boot.relimp`.

Slots

`boot`: Object of class `"boot"`, output from bootstrapping

`type`: Object of class `"character"`, metrics that have been bootstrapped

`nboot`: Object of class `"numeric"`, number of bootstrap runs

`rank`: Object of class `"logical"`, have ranks been bootstrapped ?

`diff`: Object of class `"logical"`, have differences been bootstrapped ?

`rela`: Object of class `"logical"`, have the metrics been forced to add up to 100pct (TRUE) or not?

`fixed`: Object of class `"logical"`, has bootstrapping been done for fixed (TRUE) or random (FALSE) regressors?

`namen`: Object of class `"character"`: variable names of all variables (y and the regressors)

`nobs`: Object of class `"numeric"`: number of valid observations used in calculations

`always`: Object of class `"numintnull"`: variables always in the model

`alwaysnam`: Object of class `"charnull"`: variable names of variables always in model

`groupdocu`: Object of class `"list"`: information on group definitions

`wt`: Object of class `"numintnull"`: observation weights

`vcov`: Object of class `"numintmatnull"`: variance-covariance matrix of the estimate

`call`: Object of class `"langnull"`: the call that created the object

Methods

This documentation section documents S4 methods only. There is an analogous S3-method for printing.

show S4 method: identical to `print`

Author(s)

Ulrike Groemping, BHT Berlin

See Also

[boot.relimp](#), [booteval.relimp](#), [relimplmbooteval-class](#),[relimplmbootMI-class](#)

---

relimplmbooteval-class

Class relimplmbooteval

Description

Output object from function [booteval.relimp](#), described there.

Objects from the Class

Objects should only be created by calls to function booteval.relimp.

Slots

lmg.lower: Object of class "matrix" ~~

lmg.upper: Object of class "matrix" ~~

lmg.rank.lower: Object of class "matrix" ~~

lmg.rank.upper: Object of class "matrix" ~~

lmg.diff.lower: Object of class "matrix" ~~

lmg.diff.upper: Object of class "matrix" ~~

pmvd.lower: Object of class "matrix" ~~

pmvd.upper: Object of class "matrix" ~~

pmvd.rank.lower: Object of class "matrix" ~~

pmvd.rank.upper: Object of class "matrix" ~~

pmvd.diff.lower: Object of class "matrix" ~~

pmvd.diff.upper: Object of class "matrix" ~~

last.lower: Object of class "matrix" ~~

last.upper: Object of class "matrix" ~~

last.rank.lower: Object of class "matrix" ~~

last.rank.upper: Object of class "matrix" ~~

last.diff.lower: Object of class "matrix" ~~

last.diff.upper: Object of class "matrix" ~~

first.lower: Object of class "matrix" ~~

first.upper: Object of class "matrix" ~~

first.rank.lower: Object of class "matrix" ~~

first.rank.upper: Object of class "matrix" ~~

first.diff.lower: Object of class "matrix" ~~

first.diff.upper: Object of class "matrix" ~~

betasq.lower: Object of class "matrix" ~~

betasq.upper: Object of class "matrix" ~~

betasq.rank.lower: Object of class "matrix" ~~

betasq.rank.upper: Object of class "matrix" ~~

betasq.diff.lower: Object of class "matrix" ~~

betasq.diff.upper: Object of class "matrix" ~~

pratt.lower: Object of class "matrix" ~~

pratt.upper: Object of class "matrix" ~~

pratt.rank.lower: Object of class "matrix" ~~

pratt.rank.upper: Object of class "matrix" ~~

pratt.diff.lower: Object of class "matrix" ~~

pratt.diff.upper: Object of class "matrix" ~~

genizi.lower: Object of class "matrix" ~~

genizi.upper: Object of class "matrix" ~~

genizi.rank.lower: Object of class "matrix" ~~

genizi.rank.upper: Object of class "matrix" ~~

genizi.diff.lower: Object of class "matrix" ~~

genizi.diff.upper: Object of class "matrix" ~~

car.lower: Object of class "matrix" ~~

car.upper: Object of class "matrix" ~~

car.rank.lower: Object of class "matrix" ~~

car.rank.upper: Object of class "matrix" ~~

car.diff.lower: Object of class "matrix" ~~

car.diff.upper: Object of class "matrix" ~~

var.y.boot: Object of class "numeric" ~~

R2.boot: Object of class "numeric" ~~

R2.decomp.boot: Object of class "numeric" ~~

lmg.boot: Object of class "matrix" ~~

pmvd.boot: Object of class "matrix" ~~

last.boot: Object of class "matrix" ~~

first.boot: Object of class "matrix" ~~

betasq.boot: Object of class "matrix" ~~

pratt.boot: Object of class "matrix" ~~

```
genizi.boot: Object of class "matrix" ~~
car.boot: Object of class "matrix" ~~
lmg.rank.boot: Object of class "matrix" ~~
pmvd.rank.boot: Object of class "matrix" ~~
last.rank.boot: Object of class "matrix" ~~
first.rank.boot: Object of class "matrix" ~~
betasq.rank.boot: Object of class "matrix" ~~
pratt.rank.boot: Object of class "matrix" ~~
genizi.rank.boot: Object of class "matrix" ~~
car.rank.boot: Object of class "matrix" ~~
lmg.diff.boot: Object of class "matrix" ~~
pmvd.diff.boot: Object of class "matrix" ~~
last.diff.boot: Object of class "matrix" ~~
first.diff.boot: Object of class "matrix" ~~
betasq.diff.boot: Object of class "matrix" ~~
pratt.diff.boot: Object of class "matrix" ~~
genizi.diff.boot: Object of class "matrix" ~~
car.diff.boot: Object of class "matrix" ~~
est: Object of class "numintnull" ~~
vcov: Object of class "numintmatnull" ~~
level: Object of class "numeric" ~~
nboot: Object of class "numeric" ~~
diffnam: Object of class "character" ~~
rank: Object of class "logical" ~~
diff: Object of class "logical" ~~
rela: Object of class "logical" ~~
fixed: Object of class "logical" ~~
type: Object of class "character" ~~
sort: Object of class "logical" ~~
bty: Object of class "character" ~~
mark: Object of class "matrix" ~~
markdiff: Object of class "matrix" ~~
var.y: Object of class "numeric" ~~
R2: Object of class "numeric" ~~
R2.decomp: Object of class "numeric" ~~
lmg: Object of class "numeric" ~~
pmvd: Object of class "numeric" ~~
```

`first:` Object of class "`numeric`" ~~

`last:` Object of class "`numeric`" ~~

`betasq:` Object of class "`numeric`" ~~

`pratt:` Object of class "`numeric`" ~~

`genizi:` Object of class "`numeric`" ~~

`car:` Object of class "`numeric`" ~~

`lmg.rank:` Object of class "`numeric`" ~~

`pmvd.rank:` Object of class "`numeric`" ~~

`first.rank:` Object of class "`numeric`" ~~

`last.rank:` Object of class "`numeric`" ~~

`betasq.rank:` Object of class "`numeric`" ~~

`pratt.rank:` Object of class "`numeric`" ~~

`genizi.rank:` Object of class "`numeric`" ~~

`car.rank:` Object of class "`numeric`" ~~

`lmg.diff:` Object of class "`numeric`" ~~

`pmvd.diff:` Object of class "`numeric`" ~~

`first.diff:` Object of class "`numeric`" ~~

`last.diff:` Object of class "`numeric`" ~~

`betasq.diff:` Object of class "`numeric`" ~~

`pratt.diff:` Object of class "`numeric`" ~~

`genizi.diff:` Object of class "`numeric`" ~~

`car.diff:` Object of class "`numeric`" ~~

`namen:` Object of class "`character`" ~~

`nobs:` Object of class "`numeric`" ~~

`ave.coeffs:` Object of class "`numintmatnull`" ~~

`always:` Object of class "`numintnull`" ~~

`alwaysnam:` Object of class "`charnull`" ~~

`groupdocu:` Object of class "`list`" ~~

`call:` Object of class "`langnull`" ~~

Extends

Class "relimplm", directly.

Methods

This documentation section documents S4 methods only. There are also S3-methods for printing, plotting and extracting slots with $. For their usage, see classesmethods.relaimpo.

show S4 method: identical to `print`

Author(s)

Ulrike Groemping, BHT Berlin

See Also

[relimplm-class](), [relimplmboot-class](), [relimplmbootMI-class](), [booteval.relimp](), [print.relimplmbooteval](), [plot.relimplmbooteval]()

---

relimplmbootMI-class   *Class relimplmbootMI*

Description

Output object from function [mianalyze.relimp](), described there.

Objects from the Class

Objects should only be created by calls to function mianalyze.relimp.

Slots

lmg.lower: Object of class "matrix"

lmg.upper: Object of class "matrix"

lmg.rank.lower: Object of class "matrix"

lmg.rank.upper: Object of class "matrix"

lmg.diff.lower: Object of class "matrix"

lmg.diff.upper: Object of class "matrix"

pmvd.lower: Object of class "matrix"

pmvd.upper: Object of class "matrix"

pmvd.rank.lower: Object of class "matrix"

pmvd.rank.upper: Object of class "matrix"

pmvd.diff.lower: Object of class "matrix"

pmvd.diff.upper: Object of class "matrix"

last.lower: Object of class "matrix"

last.upper: Object of class "matrix"

last.rank.lower: Object of class "matrix"

last.rank.upper: Object of class "matrix"

last.diff.lower: Object of class "matrix"

last.diff.upper: Object of class "matrix"

first.lower: Object of class "matrix"

first.upper: Object of class "matrix"

`first.rank.lower`: Object of class "matrix"

`first.rank.upper`: Object of class "matrix"

`first.diff.lower`: Object of class "matrix"

`first.diff.upper`: Object of class "matrix"

`betasq.lower`: Object of class "matrix"

`betasq.upper`: Object of class "matrix"

`betasq.rank.lower`: Object of class "matrix"

`betasq.rank.upper`: Object of class "matrix"

`betasq.diff.lower`: Object of class "matrix"

`betasq.diff.upper`: Object of class "matrix"

`pratt.lower`: Object of class "matrix"

`pratt.upper`: Object of class "matrix"

`pratt.rank.lower`: Object of class "matrix"

`pratt.rank.upper`: Object of class "matrix"

`pratt.diff.lower`: Object of class "matrix"

`pratt.diff.upper`: Object of class "matrix"

`genizi.lower`: Object of class "matrix"

`genizi.upper`: Object of class "matrix"

`genizi.rank.lower`: Object of class "matrix"

`genizi.rank.upper`: Object of class "matrix"

`genizi.diff.lower`: Object of class "matrix"

`genizi.diff.upper`: Object of class "matrix"

`car.lower`: Object of class "matrix"

`car.upper`: Object of class "matrix"

`car.rank.lower`: Object of class "matrix"

`car.rank.upper`: Object of class "matrix"

`car.diff.lower`: Object of class "matrix"

`car.diff.upper`: Object of class "matrix"

`MIresult`: Object of class "MIresult"

`est`: Object of class "numintnull"

`vcov`: Object of class "nummatnull"

`bootlist`: Object of class "listnull", list of objects of class relimplmboot or null

`level`: Object of class "numeric"

`nboot`: Object of class "numeric"

`diffnam`: Object of class "character"

`rank`: Object of class "logical"

`diff`: Object of class "logical"

rela: Object of class "logical"

fixed: Object of class "logical"

type: Object of class "character"

sort: Object of class "logical"

bty: Object of class "character"

mark: Object of class "matrix"

markdiff: Object of class "matrix"

var.y: Object of class "numeric"

R2: Object of class "numeric"

R2.decomp: Object of class "numeric"

lmg: Object of class "numeric"

pmvd: Object of class "numeric"

first: Object of class "numeric"

last: Object of class "numeric"

betasq: Object of class "numeric"

pratt: Object of class "numeric"

genizi: Object of class "numeric"

car: Object of class "numeric"

lmg.rank: Object of class "numeric"

pmvd.rank: Object of class "numeric"

first.rank: Object of class "numeric"

last.rank: Object of class "numeric"

betasq.rank: Object of class "numeric"

pratt.rank: Object of class "numeric"

genizi.rank: Object of class "numeric"

car.rank: Object of class "numeric"

lmg.diff: Object of class "numeric"

pmvd.diff: Object of class "numeric"

first.diff: Object of class "numeric"

last.diff: Object of class "numeric"

betasq.diff: Object of class "numeric"

pratt.diff: Object of class "numeric"

genizi.diff: Object of class "numeric"

car.diff: Object of class "numeric"

namen: Object of class "character"

nobs: Object of class "numeric": number of valid observations used in analysis

always: Object of class "numintnull": variables always in the model

alwaysnam: Object of class "charnull": variable names of variables always in model

Extends

Class "relimplm", directly.

Methods

This documentation section documents S4 methods only. There are also S3-methods for printing, plotting, extracting slots with $ and producing a brief summary. For their usage, see `classesmethods.relaimpo`.

show S4-method: identical to `print`
     to the slot `MIresult` for a briefer output than the standard printout

Author(s)

Ulrike Groemping, BHT Berlin

See Also

`relimplm-class`, `relimplmbooteval-class`, `booteval.relimp`, `print.relimplmbootMI`, `plot.relimplmbootMI`, `summary.relimplmbootMI`

Index