Package ‘ic.infer’

January 26, 2018

Title  Inequality Constrained Inference in Linear Normal Situations
Version 1.1-6
Depends R(>= 2.5.0)
Imports quadprog, mvtnorm, boot, kappalab
Suggests relaimpo
Date 2018-01-26
Author Ulrike Groemping
Maintainer Ulrike Groemping <groemping@beuth-hochschule.de>
Description Implements inequality constrained inference. This includes parameter estimation in
 normal (linear) models under linear equality and inequality constraints, as well as normal likeli-
 hood ratio tests involving inequality-constrained hypotheses. For inequality-constrained lin-
 ear models, averaging over R-squared for different orderings of regressors is also included.
License GPL (>= 2)
LazyLoad yes
LazyData yes
Encoding latin1
URL http://prof.beuth-hochschule.de/groemping/
NeedsCompilation no
Repository CRAN
Date/Publication 2018-01-26 22:01:12 UTC

R topics documented:

bodyfat ................................................................. 2
contr.diff ............................................................... 3
grades ................................................................. 4
ic.est ................................................................. 5
ic.infer ............................................................... 7
ic.test ............................................................... 9
ic.weights .......................................................... 13
Description

Data set with three explanatory variables and response variable body fat for 20 healthy females aged 35-44

Usage

bodyfat

Format

A data frame with four columns:

- **Triceps**  triceps skinfold thickness
- **Thigh**  thigh circumference
- **Midarm**  midarm circumference
- **BodyFat**  body fat

Details

The data set contains three explanatory variables and the response variable body fat for 20 healthy females aged 35-44. As the variable body fat is very expensive to obtain, predicting it with the cheaper dimensional measurements is desirable. There is substantial multicollinearity among the explanatory variables.

Author(s)

Ulrike Groemping, BHT Berlin

Source


The data are published on the accompanying CD-Roms of those books (Table 1 in Chapter 7) and are also available online on the books homepages or from the UCLA website linked below. (Note that earlier editions of the book had Neter as first author and included Wasserman as author, but the earlier editions do not have these data.)
**contr.diff**

**References**


---

**Description**

Function `contr.diff` is a contrast function for factors with ordered values. Coefficients for factors formatted with `contr.diff` are the increments from the current level to the neighbouring lower level.

**Usage**

```r
ccontr.diff(n, contrasts = TRUE)
```

**Arguments**

- `n`: vector of levels or integer number of levels
- `contrasts`: logical indicating whether contrasts should be computed

**Details**

The design matrix for an ordered factor formatted with `contr.diff` consists of ones for the current level itself and all lower levels. Thus, the estimated coefficients for each level are the estimated differences to the next lower level.

With this coding, the matrix `ui` in functions of package `ic.infer` can be chosen as the identity matrix for monotonicity constraints on the factor.

**Value**

A matrix with a row for each level and a column for each dummy variable (when applied to a factor in a linear model).

**Author(s)**

Ulrike Groemping, BHT Berlin

**See Also**

See also `ic.test`, `ic.est`, `orm`, `contrasts` for other contrast functions.
Examples

```r
## mu, Sigma and covariance matrix
means <- c(3,5,2,7)
## contrast matrix
contr.diff(4)
## design matrix
X <- cbind(rep(1,4),contr.diff(4))
## estimated coefficients
solve(t(X)%*%X,t(X)%*%means)
```

---

**grades**  
*Data set grades: Grade point averages by HSR and ACTC*

**Description**

The data set contains first-year grade point averages (GPAs) from 2397 Iowa university first-years who entered the university of Iowa as freshmen in the fall of 1978. The GPAs are separated out by two ordinal variables with 9 categories each, High-School-Ranking percentiles and ACT Classification.

**Usage**

grades

**Format**

A data frame with four columns:

**HSR**  high-school-ranking percentiles

**ACTC**  ACT classification (ACT is an organization that offers, among other things, college entrance exams in the US; up to 1996, ACT stood for “American College Testing”.)

**meanGPA**  grade point average for the HSR/ACTC combination

**n**  sample size for the HSR/ACTC combination

**Author(s)**

Ulrike Groemping, BHT Berlin

**Source**


Thanks go to Wiley for granting a complimentary license for embedding the data into the package.
ic.est

Functions for order-restricted estimates and printing thereof

Description
Function ic.est estimates a mean vector under linear inequality constraints, functions print.orest and summary.orest provide printed results in different degrees of detail.

Usage
ic.est(x, Sigma, ui, ci = NULL, index = 1:nrow(Sigma), meq = 0, tol = sqrt(.Machine$double.eps))

## S3 method for class 'orest'
print(x, digits = max(3,getOption("digits") - 3), scientific = FALSE, ...)

## S3 method for class 'orest'
summary(object, display.unrestr = FALSE, brief = FALSE, digits = max(3,getOption("digits") - 3), scientific = FALSE, ...)

Arguments

x
for ic.est: unrestricted vector (e.g. mean of a sample of random vectors), from which the expected value under linear inequality (and perhaps equality) restrictions is to be estimated

for print.orest: object of class orest (normally produced by ic.est or orlm)

object
for summary.orest: object of class orest (normally produced by ic.est or orlm)

Sigma
covariance or correlation matrix (or any multiple thereof) of x

ui
matrix (or vector in case of one single restriction only) defining the left-hand side of the restriction

ui%*%mu >= ci.

where mu is the expectation vector of x; the first few of these restrictions can be declared equality- instead of inequality restrictions (cf. argument meq); if only part of the elements of mu are subject to restrictions, the columns of ui can be restricted to these elements, if their index numbers are provided in index

Rows of ui must be linearly independent; in case of linearly dependent rows the function gives an error message with a hint which subset of rows is independent. Note that the restrictions must define a (possibly translated) cone, i.e. e.g. interval restrictions on a parameter are not permitted.

See contr.diff for examples of how to comfortably define various types of restriction.

ci
vector on the right-hand side of the restriction (cf. ui), defaults to a vector of zeroes

index
index numbers of the components of mu, which are subject to the specified constraints as ui%*%mu[index] >= ci
meq integer number (default 0) giving the number of rows of ui that are used for equality restrictions instead of inequality restrictions.

tol numerical tolerance value; estimates closer to 0 than tol are set to exactly 0

digits number of digits to be used in printing

scientific if FALSE, suppresses scientific representation of numbers (default: FALSE)

... further arguments to print

display.unrestr if TRUE, unrestricted estimate (i.e. object) is also displayed

brief if TRUE, suppress printing of restrictions; default: FALSE

Details

Function ic.est heavily relies on package quadprog for determining the optimizer. It is a convenience wrapper for solve.QP from that package. The function is guaranteed to work appropriately if the specified restrictions determine a (translated) cone. In that case, the estimate is the projection along matrix \(\Sigma\) onto one of the faces of that cone (including the interior as the face of the highest dimension); this means that it minimizes the quadratic form \(t(x-b)^*%solve(\Sigma,x-b)\) among all \(b\) that satisfy the restrictions \(ui*b>=ci\) (or, if specified by meq, with the first meq restrictions equality instead of inequality restrictions).

Value

Function ic.est outputs a list with the following elements:

- \(b.unrestr\) x
- \(b.restr\) restricted estimate
- \(\Sigma\) as input
- \(ui\) as input
- \(ci\) as input
- \(restr.index\) index of components of mu, which are subject to the specified constraints as in input index
- \(meq\) as input
- \(iact\) active restrictions, i.e. restrictions that are satisfied with equality in the solution, as output by solve.QP

Author(s)

Ulrike Groemping, BHT Berlin

See Also

See also ic.test, ic.weights, orlm, solve.QP
Examples

```r
## different correlation structures
corr.plus <- matrix(c(1, 0.9, 0.9, 1, 2, 2)
corr.null <- matrix(c(1, 0.9, 0.9, 1, 2, 2)
corr.minus <- matrix(c(1, -0.9, -0.9, 1, 2, 2)
## unrestricted vectors
x1 <- c(1, -1)
x2 <- c(-1, -1)
x3 <- c(10, -1)
## estimation under restriction non-negative orthant
## or first element equal to 0, second non-negative
ice <- ic.est(x1, corr.plus, ui=diag(c(1,1)), ci=c(0,0))
summary(ice)
ic2 <- ic.est(x1, corr.plus, ui=diag(c(1,1)), ci=c(0,0), meq=1)
summary(ice2)
ic.est(x2, corr.plus, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x3, corr.plus, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x1, corr.null, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x1, corr.null, ui=diag(c(1,1)), ci=c(0,0), meq=1)
ic.est(x2, corr.null, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x2, corr.null, ui=diag(c(1,1)), ci=c(0,0), meq=1)
ic.est(x3, corr.null, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x3, corr.null, ui=diag(c(1,1)), ci=c(0,0), meq=1)
ic.est(x1, corr.minus, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x1, corr.minus, ui=diag(c(1,1)), ci=c(0,0), meq=1)
ic.est(x2, corr.minus, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x2, corr.minus, ui=diag(c(1,1)), ci=c(0,0), meq=1)
ic.est(x3, corr.minus, ui=diag(c(1,1)), ci=c(0,0))
ic.est(x3, corr.minus, ui=diag(c(1,1)), ci=c(0,0), meq=1)
## estimation under one element restricted to being non-negative
ic.est(x3, corr.plus, ui=1, ci=0, index=1)
ic.est(x3, corr.plus, ui=1, ci=0, index=2)
```

ic.infer  
Package for inequality-constrained estimation and testing

Description

Package ic.infer implements estimation and testing for multivariate normal expectations with linear equality- and inequality constraints. This also includes inference on linear models with linear equality- and inequality constraints on the parameters. Decomposition of R-squared is also included for these models.

Details

Function ic.est estimates the constrained expectation of a multivariate normal random vector, function ic.test conducts related tests.
Function `orlm` estimates constrained parameters in normal linear models based on a linear model object or a covariance matrix. The function offers the possibility of bootstrapping the estimates. Tests and confidence intervals are provided by a summary function.

Function `or.relimp` decomposes the $R^2$-values analogously to metric `lmg` in package `relaimpo` for unconstrained linear models. However, `or.relimp` is far less comfortable to use and subject to severe limitations, since automatic selection of restrictions for sub models is not in all cases trivial.

The package makes use of various other R packages: `quadprog` is used for constrained estimation, `mvtnorm` in calculation of weights for null distributions of test statistics, `kappalab` for averaging over orderings in function `or.relimp`, and `boot` for bootstrapping.

The theory behind inequality-constrained estimation and testing as well as functionality of the package are explained in a vignette (Link from within dynamic help: `NNOdocOicNinferNpdf`) that is based on Groemping (2010). The vignette can also be opened from the command line by `vignetteHBicNinferBI`.

**Value**

The output of function `ic.est` belongs to S3 class `orest`.
The output of function `ic.test` belongs to S3 class `ict`.
The output of function `orlm` belongs to S3 classes `orlm` and `orest`.
All these classes offer print and summary methods.
The output of function `or.relimp` is a named vector.

**Acknowledgements**

This package uses as an internal function the function `nchoosek` from `vsn`, authored by Wolfgang Huber, available under LGPL.

It also uses modifications of numerical routines that were provided by John Fox in R-help.

Thanks go to Wiley for permission of incorporating the grades data from Table 1.3.1 of Robertson, Wright and Dykstra (1988) into the package.

**Author(s)**

Ulrike Groemping, BHT Berlin

**References**

ic.test

See Also

See also \texttt{ic.est, ic.test, orlm, or.relimp}, packages \pkg{boot}, \pkg{kappalab}, \pkg{mvtnorm}, \pkg{quadprog}, and \pkg{relaimpo}

Examples

\begin{verbatim}
## unrestricted linear model for grade point averages
limo <- lm(meanGPA~.-n, weights=n, data=grades)
summary(limo)
## restricted linear model with restrictions that better HSR ranking
## cannot deteriorate meanGPA
orlimo <- orlm(lm(meanGPA~.-n, weights=n, data=grades), index=2:9,
ui=make.mon.ui(grades$HSR))
summary(orlimo, brief=TRUE)
\end{verbatim}

\texttt{ic.test} \hspace{1cm} \textit{Function for testing inequality-related hypotheses for multivariate normal random variables}

Description

\texttt{ic.test} tests linear inequality hypotheses for multivariate normal means by likelihood ratio tests. print and summary functions display results in different degrees of detail.

Usage

\begin{verbatim}
ic.test(obj, TP = 1, s2 = 1, df.error = Inf,
ui0.11 = diag(rep(1, length(obj$b.restr))),
ci0.11 = NULL, meq.alt = 0,
df = NULL, wt = NULL, tol=sqrt(.Machine$double.eps), ...)
## S3 method for class 'ict'
print(x, digits = max(3, getOption("digits") - 3), scientific = FALSE, ...)
## S3 method for class 'ict'
summary(object, brief = TRUE, digits = max(3, getOption("digits") - 3),
scientific = FALSE, tol=sqrt(.Machine$double.eps), ...)
\end{verbatim}

Arguments

\begin{itemize}
\item \textbf{obj} \hspace{0.5cm} Object of class \code{orest} that contains unrestricted and restricted estimate, covariance structure, and restriction; for objects of class \code{orlm} (that inherit from class \code{orest}) information on \texttt{s2} and \texttt{df.error} is taken from \texttt{obj} (i.e. specifications of \texttt{s2} and \texttt{df.error} in the call to \texttt{ic.test} are ignored)
\item \textbf{TP} \hspace{0.5cm} type of test problem, cf. details
\item \textbf{s2} \hspace{0.5cm} multiplier that modifies the matrix \texttt{obj$Sigma} into the (estimated) covariance matrix of the unrestricted estimate; \texttt{obj$Sigma} may be a covariance matrix (\texttt{s2}=1, default), a correlation matrix or an otherwise rescaled covariance matrix (e.g. \texttt{cov.unscaled} from a linear model)
\end{itemize}
error degrees of freedom connected with estimation of \( s^2 \) (e.g. residual df from linear model); if \( df\_error < \infty \), the test is based on a mixture of beta-distributions with parameters \( df/2 \) and \( df\_error/2 \), otherwise the test is based on a mixture of chi-square distributions with degrees of freedom in \( df \).

matrix (or vector in case of one restriction only) for defining (additional) equality restrictions for TP11 (in addition to restrictions in \( \text{obj} \)); note that there must be as many columns as there are elements of vector \( \text{b.restr} \) (no extra index vector taken);

if there is overlap between restrictions in \( \text{ui0.11} \) and restrictions already present in \( \text{obj} \), restrictions already present in \( \text{obj} \) are projected out for \( \text{ui0.11} \); for example, the default choice for \( \text{ui0.11} \) means that all elements of the expectation are 0; some of these restrictions may already be present in \( \text{obj} \) and are projected out of \( \text{ui0.11} \) by \( \text{ic.test} \)

right-hand-side vector for equality restrictions defined by \( \text{ui0.11} \); so far, these should be 0!

number of equality restrictions (from beginning) that are maintained under the alternative hypothesis (for TP21)

optional vector of degrees of freedom for mixed chibar- or beta- distributions; if omitted, degrees of freedom and weights are calculated; if given, must be accompanied by corresponding \( \text{wt} \)

optional vector of weights for mixed chibar- or beta- distributions; if omitted, weights are calculated using function \( \text{ic.weights} \); if given, must be accompanied by corresponding \( \text{df} \) (can be obtained from call to \( \text{ic.weights} \) or from previous runs of \( \text{ic.test} \))

output object from \( \text{ic.test} \) (of class \( \text{ict} \))

numerical tolerance value; estimates closer to 0 than \( \text{tol} \) are set to exactly 0

Further options, e.g. algorithm for \( \text{ic.weights} \)

number of digits to display

if FALSE, suppresses scientific format; default: FALSE

output object from \( \text{ic.test} \) (of class \( \text{ict} \))

if TRUE, requests brief output without restrictions (default), otherwise restrictions are shown with indication, which are active

The following test problems are implemented:

TP=1: H0: restrictions valid with equality vs. H1: at least one inequality

TP=2: H0: all restrictions true vs. H1: at least one restriction false

TP=3: H0: restrictions false vs. H1: restrictions true (with inequality)

TP=11: H0: restriction valid with equality and further linear equalities vs. H1: at least one equality from H0 violated, restriction valid

TP=21: H0: restrictions valid (including some equality restrictions) vs. H1: at least one restriction from H0 violated, some equality restrictions are maintained
Note that TPs 1 and 11 can reject H0 even if H1 is violated by the data. Rejection of H0 does not provide evidence for H1 (but only against H0) in these TPs because H1 is not the opposite of H0. The tests concentrate their power in H1, but are only guaranteed to observe their level for the stated H0.

Also note that TP 3 does not make sense if obj involves equality restrictions (obj$meq>0).

Under TPs 1, 2, 11, and 21, the distributions of test statistics are mixtures of chi-square distributions (df.error=Inf) or beta-distributions (df.error finite) with different degrees of freedom (chi-square) or parameter combinations (beta). Shapiro (1988) gives detailed information on the mixing weights for the different scenarios. Basically, there are two different situations:

If meq=0, the weights are probabilities that a random variable with covariance matrix $\mathbf{u}_i\%\%\text{cov}\%\%\%t(\mathbf{u}_i)$ is realized in the positive orthant or its lower-dimensional faces, respectively (if $\mathbf{u}_i$ has too few columns, blow up by columns of 0s in appropriate positions) (Shapiro, formulae (5.5) or (5.10), respectively).

If meq > 0 (but not all restrictions are equality restrictions), the weights are probabilities that a random variable with covariance matrix the inverse of the lower right corner of $\text{solve}(\mathbf{u}_i\%\%\text{cov}\%\%\%t(\mathbf{u}_i))$ is realized in the positive orthant or its lower-dimensional faces, respectively (Shapiro, formula (5.9)).

These weights must then be combined with the appropriate degrees of freedom - these can be worked out by realizing that either the null hypothesis or the alternative hypothesis has fixed dimension and the respective mixing degrees of freedom are obtained by taking the difference to the dimension of the respective other hypothesis, which is correct because - given a certain dimension of the inequality-restricted estimate, the inequality-restricted estimate is a projection onto a linear space of that dimension.

The test for TP 3 (cf. e.g. Sasabuchi 1980) is based on the intersection-union principle and simply obtains its p-value as the maximum p-value from testing the individual restrictions.

Value

object of class ic.t, which is a list containing elements

TP test problem identifier (cf. argument TP)

b.unrestr unrestricted estimate

b.restr restricted estimate

ui restriction matrix, LHS

ci restriction vector, RHS

restr.index elements of mean referred to by ui and ci

meq number of equality restrictions (first meq rows of ui), meq must not exceed nrow(ui)-1

iact row numbers of active restrictions (all equality restrictions plus inequality restrictions that are met with equality by the solution b.restr)

ui.extra additional restrictions for TP=11, calculated from input parameter ui0.11 by projecting out restrictions present in ui and - if necessary - omitting linearly dependent rows

b.eqrestr equality-restricted estimate for TP=1
**b.extra.restr** estimate for null hypothesis of TP=11

**T** test statistic

**p.value** p-value

**s2** input parameter

**cov** matrix with s2*cov equal to covariance matrix of unrestricted estimate

**df.error** input parameter

**df.bar** vector of degrees of freedom for test statistic distribution, cf. also input parameter df

**wt.bar** vector of weights for test statistic distribution, cf. also input parameter wt

**Note**

Package versions up to 1.1-4 had a bug that caused p-values for TP=11 to be too large.

**Author(s)**

Ulrike Groemping, BHT Berlin

**References**


**See Also**

See also *ic.est, ic.weights*

**Examples**

```r
corr.plus <- matrix(c(1, 0.5, 0.5, 1), 2, 2)
corr.null <- matrix(c(1, 0, 0, 1), 2, 2)
corr.minus <- matrix(c(1, -0.5, -0.5, 1), 2, 2)
## unrestricted vectors
x1 <- c(1, 1)
x2 <- c(-1, 1)
ict1 <- ic.test(ic.est(x1, corr.plus, ui=diag(c(1,1)), ci=c(0,0)))
ict1
summary(ict1)
ic.test(ic.est(x1, corr.plus, ui=diag(c(1,1)), ci=c(0,0)), s2=1, df.error=10)
ic.test(ic.est(x1, corr.minus, ui=diag(c(1,1)), ci=c(0,0)))
ic.test(ic.est(x1, corr.minus, ui=diag(c(1,1)), ci=c(0,0)), s2=1, df.error=10)
ic.test(ic.est(x2, corr.plus, ui=diag(c(1,1)), ci=c(0,0)))
ic.test(ic.est(x2, corr.plus, ui=diag(c(1,1)), ci=c(0,0)), s2=1, df.error=10)
ic.test(ic.est(x2, corr.minus, ui=diag(c(1,1)), ci=c(0,0)))
ic.test(ic.est(x2, corr.minus, ui=diag(c(1,1)), ci=c(0,0)), s2=1, df.error=10)
ict2 <- ic.test(ic.est(x2, corr.plus, ui=diag(c(1,1)), ci=c(0,0)), TP=2)
```
ic.weights

functions for calculating the distributions of normal distribution order-related likelihood ratio tests

**Description**

Test statistics of normal distribution-based order-related likelihood ratio tests are often distributed as mixtures of chi-square or beta-distributions with different parameters. These functions determine the mixing weights and the cumulative distribution functions based on these. They can be directly used and are called by function ic.test.

**Usage**

```r
ic.weights(corr, ...)  
pchibar(x, df, wt)  
pbetabar(x, df1, df2, wt)
```

**Arguments**

- `corr`  
  *corr* is the correlation or covariance matrix (or any multiple thereof) of the data or coefficients for which weights are to be calculated

- `...`  
  *...* contains further arguments to be given to function `pmvnorm` of package `mvtnorm` for calculating multivariate normal rectangle probabilities; it is possible to select an algorithm (default in current version of `mvtnorm`: `algorithm = GenzBretz()`) and to tune weight accuracy by modifying including additional parameters into the algorithm specification, cf. help for `GenzBretz`

**Example**

```r
summary(ic.t2)  
ic.t3 <- ic.test(ic.est(x1, corr.plus, ui=diag(c(1,1)), ci=c(0,0)), TP=3)  
summary(ic.t3)
```

```r
ic.t11 <- ic.test(ic.est(x1, corr.plus, ui=c(1,1), ci=0), TP=11, ui0.11 =c(1,0))  
summary(ic.t11)
```

```r
## larger example  
corr.plus <- diag(1,8)  
for (i in 1:7)  
  for (j in (i+1):8)  
    corr.plus[i,j] <- corr.plus[j,i] <- 0.5
u <- rbind(rep(1,6), c(-1,-1,1,1,1), c(-1,0,1,0,0,0,0,0,0,0,0,0,-1,0,1))  
ic <- ic.est(c(rep(1,4),rep(4,4)), corr.plus, ui=0, ci=rep(0,4), index=2:7, meq = 1)
ict1 <- ic.test(ict1, TP=1)  
summary(ict1)
ict2 <- ic.test(ice, TP=2)  
summary(ict2)
ict11 <- ic.test(ice, TP=11)  
summary(ict11, digits=3)
ic <- ic.est(c(rep(1,4),rep(4,4)), corr.plus, ui=0, ci=rep(0,4), index=2:7)
ic.t3 <- ic.test(ice, TP=3)  
summary(ic.t3)
```
x is the quantile for which the distribution function is to be calculated

df is the vector of the degrees of freedom for the chi-square distributions that are mixed into the chi-square-distribution with the proportions given in wt

wt each element of wt is the mixing weight of the chi-square distribution with df as in the corresponding element of df; such weights can be calculated with function ic.weights

df1 vector of first parameters of the beta-distributions to be mixed into the betabar-distribution

df2 second parameter of the beta-distributions to be mixed into the betabar-distribution; error degrees of freedom in the tests implemented for linear models in summary.orlm;

NOTE: see details for the (perhaps unexpected) constancy of df2

Details

Function `ic.weights` uses results by Kudo (1963) regarding the calculation of the weights. The weights are the probabilities that the projection along its covariance onto the non-negative orthant of a multivariate normal random vector with expectation 0 and correlation \( \text{corr} \) lies in faces of dimensions \( \text{nrow(\text{corr})} : 1 \) (in this order). It is known that these probabilities coincide with various other useful probabilities related to order-related hypothesis testing, cf. e.g. Shapiro (1988). Calculation of the weights involves various calls to function `pmvnorm` from package `mvtnorm`.

Functions `pchibar` (taken from package `ibdreg`) and `pbetabar` calculate cumulative probabilities from mixtures of chi-square and beta-distributions, respectively.

IMPORTANT: Contrary to likelihood ratio theory in linear models, the beta distributions mixed always use the error sum of squares from the unrestricted model, i.e. the smallest possible error sum of squares with a fixed no. of df. Therefore, the second df entry is not increased when decreasing the first! This is appropriate for the test statistics calculated by functions `ic.test` or `summary.orlm`, but not necessarily for test statistics obtained elsewhere.

Value

`ic.weights` returns the vector of weights, `pchibar` and `pbetabar` return the cumulative probability of the respective distribution. Function `ic.weights` relies on package `mvtnorm` for determining multivariate normal rectangle probabilities. Note that these calculations involve Monte Carlo steps so that these weights are not completely repeatable.

Author(s)

Ulrike Groemping, BHT Berlin

References


See Also

ic.test.orm, pmvnorm, GenzBretz

Examples

```r
z <- 0.5
corr <- matrix(c(1,0.9,0.9,1),2,2)
print(wt.plus <- ic.weights(corr))
T <- c(z,z)%*%solve(corr,c(z,z))
1-pchibar(T,2:0,wt.plus)
1-pbetabar(T/(T+10),2:0,10,wt.plus)
corr <- matrix(c(1,0,0,1),2,2)
print(wt.0 <- ic.weights(corr))
T <- c(z,z)%*%solve(corr,c(z,z))
1-pchibar(T,2:0,wt.0)
1-pbetabar(T/(T+10),2:0,10,wt.0)
corr <- matrix(c(1,-0.9,-0.9,1),2,2)
print(wt.minus <- ic.weights(corr))
T <- c(z,z)%*%solve(corr,c(z,z))
1-pchibar(T,2:0,wt.minus)
1-pbetabar(T/(T+10),2:0,10,wt.minus)
```

Description

nchoosek is originally taken from package vsn by Wolfgang Huber, GaussianElimination and RREF have been provided by John Fox in R-help and have been modified by the author to provide more output

Usage

```r
nchoosek(n, k)  ## not exported, calculates all combinations
GaussianElimination(A, B, tol=sqrt(.Machine$double.eps),
                     verbose=FALSE)  ## not exported
RREF(X, ...)  ## not exported, calculates reduced Echelon form
```

Arguments

- `n`: number of elements to choose from
- `k`: number of elements to choose
- `A`: argument to GaussianElimination
- `B`: argument to GaussianElimination
- `tol`: argument to GaussianElimination
- `verbose`: argument to GaussianElimination
- `X`: matrix to be reduced to reduced Echelon form
- `...`: further arguments to GaussianElimination
Value

nchoosek returns all subsets of size \( k \), for Gaussian Elimination and RREF cf. comments in code. The latter are used for reducing a matrix with less than full row rank to a set of linearly independent rows.

Author(s)

Ulrike Groemping, BHT Berlin, based on code by John Fox and Wolfgang Huber

See Also

ic.test.orlm

Examples

```r
z <- 0.5
corr <- matrix(c(1, 0.9, 0.9, 1), 2, 2)
print(wt.plus <- ic.weights(corr))
T <- c(z, z)%%solve(corr, c(z, z))
1-pchibar(T, 2:0, wt.plus)
1-pbetabar(T/(T+10), 2:0, 10, wt.plus)
corr <- matrix(c(1, 0, 0, 1), 2, 2)
print(wt.0 <- ic.weights(corr))
T <- c(z, z)%%solve(corr, c(z, z))
1-pchibar(T, 2:0, wt.0)
1-pbetabar(T/(T+10), 2:0, 10, wt.0)
corr <- matrix(c(1, -0.9, -0.9, 1), 2, 2)
print(wt.minus <- ic.weights(corr))
T <- c(z, z)%%solve(corr, c(z, z))
1-pchibar(T, 2:0, wt.minus)
1-pbetabar(T/(T+10), 2:0, 10, wt.minus)
```

make.mon.ui  

Function for creating the matrix ui for monotonicity (in)equality restrictions

Description

Function make.mon.ui creates the matrix ui for a factor, depending on its coding.

Usage

```r
make.mon.ui(x, type = "coeff", contr = NULL)
```
### make.mon.ui

#### Arguments

- **x**: An R factor (in case of `type = "coeff"`) or the dimension of the multivariate normal distribution (in case of `type = "mean"`).
- **type**: The situation for which `ui` is needed: can be `coeff` for coefficients in a linear model or `mean` for the expectation vector of a multivariate normal distribution.
- **contr**: Relevant in case of `type = "coeff"` only, ignored otherwise; the contrast with which `x` is coded; if the `contrasts` attribute of `x` is a character string, `contr = NULL` uses this character string, otherwise `contr = NULL` is identical to `contr = "contr.treatment"`. Explicit choices for `contr` can be any of `contr.treatment`, `contr.SAS`, `contr.diff` and `contr.sum` (must be given in quotes). The other generally-available codings (`contr.helmert` and `contr.poly`) do not easily permit conclusions about monotonicity. If the value for `contr` is not compatible with the factors coding, an error is thrown.

#### Details

The function determines the matrix `ui` as needed for the functions in package `ic.infer`, when a monotone increase from first to last level of the `x` is under investigation (type = "coeff") or when a monotone increase among the components of the expectation vector is investigated (type = "mean"). The respective monotone decrease can be accommodated by `make.mon.ui()`.

If the coding of the factor `x` is explicitly given, the function throws an error if the actual coding does not correspond to the specified value of `contr`.

Care is needed when using `make.mon.ui` with a linear model: It is the users responsibility to make sure that the coding used in the model corresponds to the coding used in `make.mon.ui`.

#### Value

A square matrix with as many rows and columns as there are dummy variables for the factor.

#### Author(s)

Ulrike Groemping, BHT Berlin

#### See Also

See also `contrasts` for how to apply contrasts, `contrast` for the available contrasts in package `stats`, `contr.diff` for the specific monotonicity contrast function from this package.

#### Examples

```r
gifte <- boot::poisons  ## gifte is German for poisons
## default: contr.treatment (with default base 1)
linmod <- lm(l/time-poison+treat, gifte)
summary(orlm(linmod, ui=make.mon.ui(gifte$poison), index=2:3))

## next: contr.diff
```


Function to calculate relative importance for order-restricted linear models

Description
The function calculates relative importance by averaging over the variables R-squared contributions from all orderings of variables for linear models with inequality restrictions on the parameters. NOTE: only useful if each restriction refers to exactly one variable, or if it is adequate to reduce multi-variable restrictions by omitting the affected variables but leaving the restriction otherwise intact.

Usage

```r
or.relimp(model, ui, ci = NULL, ...)  
# S3 method for class 'lm'
or.relimp(model, ui, ci = NULL, index = 2:length(coef(model)), meq = 0,  
tol = sqrt(.Machine$double.eps), ...)

# Default S3 method:
or.relimp(model, ui, ci = NULL, index = 2:ncol(model), meq = 0,  
tol = sqrt(.Machine$double.eps), ...)

all.R2(covmat, ui, ci = NULL, index = 2:ncol(covmat), meq = 0,  
tol = sqrt(.Machine$double.eps), ...)
# user does not need to call this function
```

Arguments

- `model`: a linear model object of class `lm` with data included; for function `or.relimp`, all explanatory variables must be numeric (i.e. no factors), and higher-order terms (e.g. interactions) are not permitted.

- `ui`, `ci`: additional arguments to `summary.lm` and `summary.glm`, respectively.

- `index`: a vector specifying the order of variables to be used in the calculation.

- `meq`: an integer specifying the number of equality restrictions.

- `tol`: a numeric value specifying the tolerance level for numerical comparisons.

- `all.R2`: a function to calculate the relative importance of all variables in a given model.

Example:

```r
contrasts(gife$poison) <- "contr.diff"
linmod <- lm(1/time-poisontreat, gife)
summary(orlm(linmod, ui=make.mon.ui(gife$poison), index=2:3))

# next: contr.SAS
contrasts(gife$poison) <- "contr.SAS"
linmod <- lm(1/time-poisontreat, gife)
summary(orlm(linmod, ui=make.mon.ui(gife$poison), index=2:3))

# next: contr.sum
contrasts(gife$poison) <- "contr.sum"
linmod <- lm(1/time-poisontreat, gife)
summary(orlm(linmod, ui=make.mon.ui(gife$poison), index=2:3))
```
OR
the covariance matrix of the response (first position) and all regressors
covmat
the covariance matrix of the response (first position) and all regressors
ui
cf. explanation in link{orlm}; cf. also details below
ci
cf. explanation in link{orlm}
index
cf. explanation in link{orlm}
meq
cf. explanation in link{orlm}
tol
cf. explanation in link{orlm}
... Further options

Details
Function or.relimp uses function all.R2 for calculating the R-squared values of all subsets that are subsequently handed to function Shapley.value (from package kappalab), which takes care of the averaging over ordering.

WARNING: In models with subsets of the regressors, the columns of the matrix ui referring to regressors outside the current subset are simply deleted for the sub model. This is only reasonable if either the individual constraints refer to individual parameters only (e.g. all parameters restricted to be non-negative) or if the constraints are still reasonable in the sub model with some variables deleted, e.g. perhaps (depending on the application) sum of all parameters less or equal to 1.

WARNING: If the number of regressors (p) is large, the functions quickly becomes unmanageable (a vector of size 2^p is returned or handled in the process.

Value
all.R2 returns a vector (2^p elements) with all R-squared values (p is the number of regressors, vector is ordered from empty to full model in natural order (cf. ic.infer::nchoosek for the order within one model size).
or.relimp returns a vector (p elements) with average R-squared contributions from all models with respective subset of restrictions ui %>% beta >= ci enforced.

Author(s)
Ulrike Groemping, BHT Berlin

See Also
See also orlm for order-restricted linear models and calc.relimp from R-package relaimpo for a much more comfortable and much faster routine for unrestricted linear models

Examples

covswiss <- cov(swiss)
## all R2-values for restricted linear model with restrictions that
## Catholic and Infant.Mortality have non-negative coefficients
R2s <- all.R2(covswiss, ui=rbind(c(0,0,0,1,0),c(0,0,0,0,1)))
R2s
require(kappalab) ## directly using package kappalab
Shapley.value(set.func(R2s))

### with convenience wrapper from this package
or.relimp(covswiss, ui=rbind(c(0,0,0,1,0),c(0,0,0,0,1)))

### also works on linear models
limo <- lm(swiss)
#or.relimp(limo, ui=rbind(c(0,0,0,1,0),c(0,0,0,0,1)))

## same model using index vector
or.relimp(limo, ui=rbind(c(1,0),c(0,1)), index=5:6)

---

**Description**

Function `orlm` calculates order-restricted linear models (linear equality and inequality constraints). It uses the internal function `boot.orlm` for bootstrapping, which in turn uses the internal functions `orlm.forboot...`. The remaining functions extract coefficients, provide a residual plot, give a short printout or a more extensive summary.

**Usage**

```r
orlm(model, ui, ci, ...)
## S3 method for class 'lm'
orlm(model, ui, ci, index = 2:length(coef(model)), meq = 0,
     orig.out = FALSE, boot = FALSE, B = 1000, fixed = FALSE,
     tol = sqrt(.Machine$double.eps), ...)
## Default S3 method:
orlm(model, ui, ci, index = NULL, meq = 0,
     tol = sqrt(.Machine$double.eps), df.error = NULL, ...)
boot.orlm(model, B = 1000, fixed = FALSE, ui, ci, index, meq)
orlm.forboot.fixed(data, indices, ...)
orlm.forboot(data, indices, index = index, ...)
## S3 method for class 'orlm'
coef(object, ...)
## S3 method for class 'orlm'
plot(x, caption = "Residuals vs Fitted",
     panel = if (add.smooth) panel.smooth else points, sub.caption = NULL,
     main = "", ..., id.n = 3, labels.id = names(x$residuals), cex.id = 0.75,
     add.smooth = getOption("add.smooth"), label.pos = c(4, 2),
     cex.caption = 1)
## S3 method for class 'orlm'
print(x, digits = max(3, getOption("digits") - 3), ...)
```
## S3 method for class 'orlm'

```r
summary(object, display.unrestr = FALSE, brief = FALSE,
        digits = max(3, getOption("digits") - 3),
        scientific = FALSE, overall.tests = TRUE,
        bootCIs = TRUE, bty = "perc", level = 0.95, ...)
```

### Arguments

- **model**
  - a linear model object (class `lm`) with data included
  - OR
  - a covariance matrix of Y and all regressors (in this order)
- **ui**
  - matrix (or vector in case of one single restriction only) defining the left-hand side of the restriction
  - `ui %*% beta >= ci`,
  - where `beta` is the parameter vector; the first few of these restrictions can be declared equality- instead of inequality restrictions (cf. argument `meq`); if only part of the elements of `beta` are subject to restrictions, the columns of `ui` can be restricted to these elements, if their index numbers are provided in `index`; by default, `index` excludes the intercept, i.e. the columns of `ui` refer to the non-intercept elements of `coef(model)`
- **ci**
  - vector on the right-hand side of the restriction (cf. `ui`)
- **index**
  - index numbers of the components of `beta`, which are subject to the specified constraints as `ui %*% beta[index] >= ci`, default is `index = 2:length(coef(model))`, i.e. `ui` is supposed to have columns for all coefficients except the intercept;
  - CAUTIONs:
    - `index` refers to the position of the coefficient in the model. The first coefficient is usually the intercept (which is therefore per default excluded from restrictions).
    - If the intercept is included into restrictions (model with intercept, `index` containing the element 1, intercept-related column of `ui` not consisting of zeroes only), R-squared values may become unreasonable, if the restriction on the intercept is active.
- **meq**
  - integer number (default 0) giving the number of rows of `ui` that are used for equality restrictions instead of inequality restrictions.
- **orig.out**
  - should the original model be included in the output list? (default: FALSE)
- **boot**
  - should bootstrapping be conducted? (default: FALSE)
- **B**
  - number of bootstrap samples (default: 1000)
- **fixed**
  - should bootstrapping consider the sample as fixed and bootstrap residuals? (default: FALSE)
data handed to bootstrap sampling routine
indices for sampling
tol numerical tolerance value; estimates closer to 0 than tol are set to exactly 0
df.error error degrees of freedom (number of observations minus number of columns of covariance matrix) for orlm.default; required in order to calculate adequate covariance matrix and tests; valid coefficient estimates can also be obtained for arbitrary values of df.error

Further options

object object of class orlm (created by function orlm)
x object of class orlm (created by function orlm)
caption like in function plot.lm
panel like in function plot.lm
sub.caption like in function plot.lm
main like in function plot.lm
id.n like in function plot.lm
labels.id like in function plot.lm
cex.id like in function plot.lm
add.smooth like in function plot.lm
label.pos like in function plot.lm
cex.caption like in function plot.lm
digits number of digits to display
display.unrestr if TRUE, also display unrestricted model; default: FALSE
brief if TRUE, suppress printing of restrictions; default: FALSE
scientific if FALSE, suppresses scientific format; default: FALSE
overall.tests if FALSE, suppresses output of overall model tests; default: TRUE; for models with large sets of restrictions, tests can take up substantial time because of weight calculation
bootCIs if FALSE, suppresses bootstrap confidence intervals, even though the obj contains a bootout element; default: TRUE
bty type of bootstrap confidence interval; any of "perc", "bca", "norm" or "basic", cf. function boot.ci from package boot, default: "perc"
level confidence level for bootstrap confidence intervals, default: 0.95

Details

Function orlm performs order restricted linear model analysis. Functions coef.orlm, plot.orlm, print.orlm, and summary.orlm provide methods for reporting the results on an object of S3 class orlm. The functions directly referring to bootstrapping are internal and should not be called by the user but are called from within function orlm if option boot is set to TRUE.
Of course, bootstrapping is not possible, if function orlm is applied to a covariance matrix, since the raw data are not available in this case. Also note that the intercept is not estimated in this case but can easily be estimated from the resulting estimate if the variable means are known (cf. example).

The output from summary.orlm provides information about the restrictions, a comparison of $R^2$-values for unrestricted and restricted model, restricted estimates, and

- if requested (option boot set to TRUE in function orlm and option bootCIs set to TRUE in the summary function) with bootstrap confidence intervals,

- if requested (option overall.tests set to TRUE) several restriction-related tests (implemented by calls to ic.test): The analogue to the overall F-Test in the ordinary linear model is the test of all coefficients but intercept equal to 0 within the restricted parameter space. In addition, three tests related to the restriction are reported:

Test 1: H0: Restriction valid with equality vs. H1: at least one inequality
Test 2: H0: Restriction valid vs. H1: restriction violated
Test 3: H0: Restriction violated or valid with equality vs. H1: all restrictions valid with inequality

Test 3 is conducted in case of no equality-restrictions only.

Value

The output of function orlm belongs to S3 classes orlm and orrest. It is a list with the following items:

- `b.restr`: restricted estimate
- `b.unrestr`: unrestricted estimate
- `R2`: R-squared
- `residuals`: residuals of restricted model
- `fitted.values`: fitted values of restricted model
- `weights`: observation weights
- `orig.R2`: R-squared of unrestricted model
- `df.error`: error degrees of freedom of unrestricted model
- `s2`: MSE of unrestricted model
- `Sigma`: variance covariance matrix of beta-hat in unrestricted model
- `origmodel`: unrestricted model itself (NULL, if orig.out=FALSE)
- `ui`: as input
- `ci`: as input
- `restr.index`: the input vector index
- `meq`: as input
- `iact`: active restrictions, i.e. restrictions that are satisfied with equality in the solution, as output by solve.QP
- `bootout`: object of class boot obtained by bootstrapping, will be used by summary.orlm for calculating bootstrap confidence intervals; NULL if boot=FALSE
Note

Package versions up to 1.1-4 had a bug in function ic.test that caused the p-value of the overall model test to be too large.

Author(s)

Ulrike Groemping, BHT Berlin

References


See Also

See also `ic.est, ic.test, or.relimp, solve.QP`

Examples

```r
limo <- lm(swiss)
## restricted linear model with restrictions that
## - Education and Examination have same coefficient
## - Catholic and Infant.Mortality have non-negative coefficients
orlimo <- orlm(limo, ui=rbind(c(0,1,-1,0,0),c(0,0,0,1,0),c(0,0,0,0,1)), meq=1)

orlimo
plot(orlimo)
summary(orlimo)
## same model using index vector
orlimo <- orlm(limo, ui=rbind(c(1,-1,0,0),c(0,0,1,0),c(0,0,0,1)), index=3:6, meq=1)

## reduced number of bootstrap samples below reasonable size for example run time
orlimo <- orlm(limo, ui=rbind(c(1,-1,0,0),c(0,0,1,0),c(0,0,0,1)),
  index=3:6, meq=1, boot=TRUE, B=100)
summary(orlimo)

## bootstrap considering data as fixed
orlimof <- orlm(limo, ui=rbind(c(1,-1,0,0),c(0,0,1,0),c(0,0,0,1)),
  index=3:6, meq=1, boot=TRUE, B=100, fixed=TRUE)
summary(orlimof, brief=TRUE)
```
Index

*Topic datasets
  bodyfat, 2
  grades, 4
*Topic htest
  ic.infer, 7
  ic.test, 9
  orlm, 20
*Topic models
  ic.infer, 7
  or.relimp, 18
  orlm, 20
*Topic multivariate
  contr.diff, 3
  ic.est, 5
  ic.infer, 7
  ic.test, 9
  ic.weights, 13
  internal.functions, 15
  make.mon.ui, 16
  or.relimp, 18
  orlm, 20
*Topic optimize
  contr.diff, 3
  ic.est, 5
  make.mon.ui, 16
*Topic regression
  ic.infer, 7
  or.relimp, 18
  orlm, 20

all.R2 (or.relimp), 18

gaussianelimination
  (internal.functions), 15
  grades, 4
  ic.est, 3, 5, 9, 12, 24
  ic.infer, 7
  ic.infer-package (ic.infer), 7
  ic.test, 3, 6, 9, 9, 15, 16, 24
  ic.weights, 6, 10, 12, 13
  internal.functions, 15
  make.mon.ui, 16

nchoosek (internal.functions), 15

or.relimp, 9, 18, 24
  orlm, 3, 6, 9, 15, 16, 19, 20

pbetabar (ic.weights), 13
pchibar (ic.weights), 13
plot.lm, 22
plot.orlm (orlm), 20
print.ict (ic.test), 9
print.orest (ic.est), 5
print.orlm (orlm), 20

RREF (internal.functions), 15

summary.ict (ic.test), 9
summary.orest (ic.est), 5
summary.orlm (orlm), 20

bodyfat, 2
boot.orlm (orlm), 20

test.orlm (orlm), 20
contr.diff, 3, 5, 17, 21
contrast, 17
contrasts, 3, 17