Package ‘CLME’

January 20, 2025

Title Constrained Inference for Linear Mixed Effects Models

Version 2.0-12

Depends R (>= 4.0.0), shiny, lme4, MASS

Imports nlme, methods, isotone, stringr, prettyR, stats, openxlsx, graphics

Suggests testthat, nnet

Date 2020-06-04

Description Estimation and inference for linear models where some or all of the fixed-effects coefficients are subject to order restrictions. This package uses the robust residual bootstrap methodology for inference, and can handle some structure in the residual variance matrix.

Maintainer Casey M. Jelsema <jelsema.casey@gmail.com>

BugReports https://github.com/jelsema/CLME/issues

LazyLoad no

License GPL-3

Encoding UTF-8

RoxygenNote 7.1.0

NeedsCompilation no

Author Casey M. Jelsema [aut, cre], Shyamal D. Peddada [aut]

Repository CRAN

Date/Publication 2020-06-07 18:30:02 UTC

Contents

`CLME-package` *Constrained inference for linear mixed models.*

Description

Constrained inference on linear fixed and mixed models using residual bootstrap. Covariates and random effects are permitted but not required.

Appropriate credit should be given when publishing results obtained using **CLME**, or when developing other programs/packages based off of this one. Use `citation(package="CLME")` for Bibtex information.

The work was produced in part with funding from the Intramural Research Program of the NIH, National Institute of Environmental Health Sciences (Z01 ES101744).

Details

This package was introduced in Jelsema and Peddada (2016). The primary function is `clme`. The other functions in this package may be run separately, but in general are designed for use by `clme`.

The method which is implemented is the constrained linear mixed effects model described in Farnan, Ivanova, and Peddada (2014). See that paper for more details regarding the method. Here we give a brief overview of the assumed model:

$$Y = X_1\theta_1 + X_2\theta_2 + U\xi + \epsilon$$

where

- X_1 is a $N \times p_1$ design matrix.
- θ_1 are the coefficients (often treatment effects).
- X_2 is a $N \times p_2$ matrix of fixed covariates.
- θ_1 are the coefficients for the covariates.
- U is a $N \times c$ matrix of random effects.
- ξ is a zero-mean random vector with covariance T (see below).
- ϵ is a zero-mean random vector with covariance Σ (see below).

Neither covariates (X_2) nor random effects (U) are required by the model or **CLME**. The covariance matrix of ξ is given by:

$$T = diag\left(\tau_1^2 I_{c_1}, \tau_2^2 I_{c_2}, \ldots, \tau_q^2 I_{c_q}\right)$$

The first c_1 random effects will share a common variance, τ_1^2, the next c_2 random effects will share a common variance, and so on. Note that $c = \sum_{i=1}^{q} c_i$. Homogeneity of variances in the random effects can be induced by letting $q = 1$ (hence $c_1 = c = ncol(U)$).

Similarly, the covariance matrix of ϵ is given by:

$$\Sigma = diag\left(\sigma_1^2 I_{n_1}, \sigma_2^2 I_{n_2}, \ldots, \sigma_q^2 I_{n_k}\right)$$

Again, the first n_1 observations will share a common variance, σ_1^2, the next n_2 will share a common variance, and so on. Note that $N = \sum_{i=1}^{k} n_i$. Homogeneity of variances in the residuals can be induced by letting $k = 1$.

The order constraints are defined by the matrix A. This is an $r \times p$ matrix where r is the number of constraints, and $p = p_1 + p_2$ is the dimension of $\theta = (\theta_1', \theta_2')'$. Formally the hypothesis being tested is:

$$H_a : A\theta > 0$$

For several default orders (simple, umbrella, simple tree) the A matrix can be automatically generated. Alternatively, the user may define a custom A matrix to test other patterns among the elements of θ. See `create.constraints` and `clme` for more details.

For computational reasons, the implementation is not identical to the model expressed. Particularly, the fixed-effects matrix (or matrices) and the random effects matrix are assumed to be columns in a data frame, not passed as matrices. The A matrix is not $r\ timesp$, but $r\ times2$, where each row gives the indices of the constrained coefficients. See `create.constraints` for further explanation.

The creation of this package **CLME**, this manual, and the vignette were all supported by the Intramural Research Program of the United States' National Institutes of Health (Z01 ES101744).

Author(s)

Maintainer: Casey M. Jelsema <jelsema.casey@gmail.com>

Authors:

- Shyamal D. Peddada

References

Jelsema, C. M. and Peddada, S. D. (2016). CLME: An R Package for Linear Mixed Effects Models under Inequality Constraints. *Journal of Statistical Software*, 75(1), 1-32. doi:10.18637/jss.v075.i01

Farnan, L., Ivanova, A., and Peddada, S. D. (2014). Linear Mixed Efects Models under Inequality Constraints with Applications. *PLOS ONE*, 9(1). e84778. doi: 10.1371/journal.pone.0084778

See Also

Useful links:

- Report bugs at https://github.com/jelsema/CLME/issues

`AIC.clme` *Akaike information criterion*

Description

Calculates the Akaike and Bayesian information criterion for objects of class `clme`.

Calculates the Akaike and Bayesian information criterion for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
AIC(object, ..., k = 2)

## S3 method for class 'summary.clme'
AIC(object, ..., k = 2)
```

Arguments

`object`	object of class `clme`.
`...`	space for additional arguments.
`k`	value multiplied by number of coefficients

Details

The log-likelihood is assumed to be the Normal distribution. The model uses residual bootstrap methodology, and Normality is neither required nor assumed. Therefore the log-likelihood and these information criterion may not be useful measures for comparing models. For `k=2`, the function computes the AIC. To obtain BIC, set $k = log(n/(2 * pi))$; which the method `BIC.clme` does.

Value

Returns the information criterion (numeric).

See Also

`CLME-package` `clme`

`CLME-package` `clme`

Examples

```
data( rat.blood )

cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

AIC( clme.out )
AIC( clme.out, k=log( nobs(clme.out)/(2*pi) ) )
```

`BIC.clme` *Bayesian information criterion*

Description

Calculates the Bayesian information criterion for objects of class `clme`.

Calculates the Akaike and Bayesian information criterion for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
BIC(object, ..., k = log(nobs(object)/(2 * pi)))

## S3 method for class 'summary.clme'
BIC(object, ..., k = log(nobs(object)/(2 * pi)))
```

Arguments

`object`	object of class `clme`.
`...`	space for additional arguments.
`k`	value multiplied by number of coefficients

Details

The log-likelihood is assumed to be the Normal distribution. The model uses residual bootstrap methodology, and Normality is neither required nor assumed. Therefore the log-likelihood and these information criterion may not be useful measures for comparing models. For `k=2`, the function computes the AIC. To obtain BIC, set $k = log(n/(2 * pi))$; which the method `BIC.clme` does.

Value

Returns the Bayesian information criterion (numeric).

See Also

`CLME-package` `clme`

`CLME-package` `clme`

Examples

```
data( rat.blood )

cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

BIC( clme.out )
BIC( clme.out, k=log( nobs(clme.out)/(2*pi) ) )
```

`clme` *Constrained Inference for Linear Mixed Effects Models*

Description

Constrained inference for linear fixed or mixed effects models using distribution-free bootstrap methodology

Usage

```
clme(
  formula,
  data = NULL,
  gfix = NULL,
  constraints = list(),
  tsf = lrt.stat,
  tsf.ind = w.stat.ind,
  mySolver = "LS",
  all_pair = FALSE,
  verbose = c(FALSE, FALSE, FALSE),
  ...
)
```

Arguments

formula	a formula expression. The constrained effect must come before any unconstrained covariates on the right-hand side of the expression. The constrained effect should be an ordered factor.
data	data frame containing the variables in the model.
gfix	optional vector of group levels for residual variances. Data should be sorted by this value.
constraints	optional list containing the constraints. See Details for further information.
tsf	function to calculate the test statistic.
tsf.ind	function to calculate the test statistic for individual constrats. See Details for further information.
mySolver	solver to use in isotonization (passed to activeSet).
all_pair	logical, whether all pairwise comparisons should be considered (constraints will be ignored).
verbose	optional. Vector of 3 logicals. The first causes printing of iteration step, the second two are passed as the verbose argument to the functions minque and clme_em, respectively.
...	space for additional arguments.

Details

If any random effects are included, the function computes MINQUE estimates of variance components. After, clme_em is run to obtain the observed values. If nsim>0, a bootstrap test is performed using resid_boot. For the argument levels the first list element should be the column index (in data) of the constrained effect. The second element should be the true order of the levels.

Value

The output of clme is an object of the class clme, which is list with elements:

- theta estimates of θ coefficients

- theta estimates of θ_0 coefficients under the null hypothesis
- ssq estimate of residual variance(s), σ_i^2.
- tsq estimate of random effects variance component(s), τ_i^2.
- cov.theta the unconstrained covariance matrix of θ
- ts.glb test statistic for the global hypothesis.
- ts.ind test statistics for each of the constraints.
- mySolver the solver used for isotonization.
- constraints list containing the constraints (A) and the contrast for the global test (B).
- dframe data frame containing the variables in the model.
- residuals matrix containing residuals. For mixed models three types of residuals are given.
- random.effects estimates of random effects.
- gfix group sample sizes for residual variances.
- gran group sizes for random effect variance components.
- gfix_group group names for residual variances.
- formula the formula used in the model.
- call the function call.
- order list describing the specified or estimated constraints.
- P1 the number of constrained parameters.
- nsim the number of bootstrap simulations used for inference.

Note

The argument constraints is a list containing the order restrictions. The elements are order, node, decreasing, A, and B, though not all are necessary. The function can calculate the last two for default orders (simple, umbrella, or simple tree). For default orders, constraints should be a list containing any subset of order, node, and descending. See Figure 1 from Jelsema \& Peddada (2016); the pictured node of the simple tree orders (middle column) is 1, and the node for the umbrella orders (right column) is 3. These may be vectors (e.g. order=('simple','umbrella')). If any of these three are missing, the function will test for all possible values of the missing element(s), excluding simple tree.

For non-default orders, the elements A and B should be provided. A is an $r \times 2$ matrix (where r is the number of linear constraints, $0 < r$. Each row should contain two indices, the first element is the index of the lesser coefficient, the second element is the index of the greater coefficient. So a row of $(1, 2)$ corresponds to the constraint $\theta_1 \leq \theta_2$, and a row $(4, 3)$ corresponds to the constraint $\theta_4 \leq \theta_3$, etc. Element B should hold similar contrasts, specifically those needed for calculating the Williams' type test statistic (B is only needed if tsf=w.stat) The argument tsf is a function to calculate the desired test statistic. The default function calculates likelihood ratio type test statistic. A Williams type test statistic, which is the maximum of the test statistic over the constraints in constraints\$B, is also available, and custom functions may be defined. See w.stat for details. By default, homogeneity of variances is assumed for residuals (e.g., gfix does not define groups) and for each random effect. Some values can be passed to clme that are not used in this function. For instance, seed and nsim can each be passed as an argument here, and summary.clme will use these values.

References

Jelsema, C. M. and Peddada, S. D. (2016). CLME: An R Package for Linear Mixed Effects Models under Inequality Constraints. *Journal of Statistical Software*, 75(1), 1-32. doi:10.18637/jss.v075.i01

Examples

```
data( rat.blood )
cons <- list(order="simple", decreasing=FALSE, node=1 )

clme.out <- clme(mcv ~ time + temp + sex + (1|id), data=rat.blood ,
                constraints=cons, seed=42, nsim=10 )
```

---

clme_em_fixed — *Constrained EM algorithm for linear fixed or mixed effects models.*

Description

clme_em_fixed performs a constrained EM algorithm for linear fixed effects models.

clme_em_mixed performs a constrained EM algorithm for linear mixed effects models.

clme_em is the general function, it will call the others. These Expectation-maximization (EM) algorithms estimate model parameters and compute a test statistic.

Usage

```
clme_em_fixed(
  Y,
  X1,
  X2 = NULL,
  U = NULL,
  Nks = dim(X1)[1],
  Qs = dim(U)[2],
  constraints,
  mq.phi = NULL,
  tsf = lrt.stat,
  tsf.ind = w.stat.ind,
  mySolver = "LS",
  em.iter = 500,
  em.eps = 1e-04,
  all_pair = FALSE,
  dvar = NULL,
  verbose = FALSE,
  ...
)

clme_em_mixed(
```

```
  Y,
  X1,
  X2 = NULL,
  U = NULL,
  Nks = dim(X1)[1],
  Qs = dim(U)[2],
  constraints,
  mq.phi = NULL,
  tsf = lrt.stat,
  tsf.ind = w.stat.ind,
  mySolver = "LS",
  em.iter = 500,
  em.eps = 1e-04,
  all_pair = FALSE,
  dvar = NULL,
  verbose = FALSE,
  ...
)

clme_em(
  Y,
  X1,
  X2 = NULL,
  U = NULL,
  Nks = nrow(X1),
  Qs = ncol(U),
  constraints,
  mq.phi = NULL,
  tsf = lrt.stat,
  tsf.ind = w.stat.ind,
  mySolver = "LS",
  em.iter = 500,
  em.eps = 1e-04,
  all_pair = FALSE,
  dvar = NULL,
  verbose = FALSE,
  ...
)
```

Arguments

Y	$N \times 1$ vector of response data.
X1	$N \times p_1$ design matrix.
X2	optional $N \times p_2$ matrix of covariates.
U	optional $N \times c$ matrix of random effects.
Nks	optional $K \times 1$ vector of group sizes.
Qs	optional $Q \times 1$ vector of group sizes for random effects.

constraints	list containing the constraints. See Details.
mq.phi	optional MINQUE estimates of variance parameters.
tsf	function to calculate the test statistic.
tsf.ind	function to calculate the test statistic for individual constrats. See Details for further information.
mySolver	solver to use in isotonization (passed to activeSet).
em.iter	maximum number of iterations permitted for the EM algorithm.
em.eps	criterion for convergence for the EM algorithm.
all_pair	logical, whether all pairwise comparisons should be considered (constraints will be ignored).
dvar	fixed values to replace bootstrap variance of 0.
verbose	if TRUE, function prints messages on progress of the EM algorithm.
...	space for additional arguments.

Details

Argument constraints is a list including at least the elements A, B, and Anull. This argument can be generated by function create.constraints.

Value

The function returns a list with the elements:

- theta coefficient estimates.
- theta.null vector of coefficient estimates under the null hypothesis.
- ssq estimate of residual variance term(s).
- tsq estimate of variance components for any random effects.
- cov.theta covariance matrix of the unconstrained coefficients.
- ts.glb test statistic for the global hypothesis.
- ts.ind test statistics for each of the constraints.
- mySolver the solver used for isotonization.

Note

There are few error catches in these functions. If only the EM estimates are desired, users are recommended to run clme setting nsim=0.

By default, homogeneous variances are assumed for the residuals and (if included) random effects. Heterogeneity can be induced using the arguments Nks and Qs, which refer to the vectors $(n_1, n_2, \ldots, n_k)$ and $(c_1, c_2, \ldots, c_q)$, respectively. See CLME-package for further explanation the model and these values.

See w.stat and lrt.stat for more details on using custom test statistics.

See Also

CLME-package clme create.constraints lrt.stat w.stat

Examples

```
data( rat.blood )

model_mats <- model_terms_clme( mcv ~ time + temp + sex + (1|id), data = rat.blood )

Y  <- model_mats$Y
X1 <- model_mats$X1
X2 <- model_mats$X2
U  <- model_mats$U

cons <- list(order = "simple", decreasing = FALSE, node = 1 )

clme.out <- clme_em(Y = Y, X1 = X1, X2 = X2, U = U, constraints = cons)
```

`clme_resids` *Computes various types of residuals*

Description

Computes several types of residuals for objects of class `clme`.

Usage

```
clme_resids(formula, data, gfix = NULL)
```

Arguments

`formula`	a formula expression. The constrained effect(s) must come before any unconstrained covariates on the right-hand side of the expression. The first `ncon` terms will be assumed to be constrained.
`data`	data frame containing the variables in the model.
`gfix`	optional vector of group levels for residual variances. Data should be sorted by this value.

Details

For fixed-effects models $Y = X\beta + \epsilon$, residuals are given as $\hat{e} = Y - X\hat{\beta}$. For mixed-effects models $Y = X\beta + +U\xi + \epsilon$, three types of residuals are available. $PA = Y - X\hat{\beta}\backslash SS = U\hat{\xi}\backslash FM = Y - X\hat{\beta} - U\hat{\xi}$

Value

List containing the elements `PA`, `SS`, `FM`, `cov.theta`, `xi`, `ssq`, `tsq`. `PA`, `SS`, `FM` are defined above (for fixed-effects models, the residuals are only `PA`). Then `cov.theta` is the unconstrained covariance matrix of the fixed-effects coefficients, `xi` is the vector of random effect estimates, and `ssq` and `tsq` are unconstrained estimates of the variance components.

Note

There are few error catches in these functions. If only the EM estimates are desired, users are recommended to run clme setting nsim=0.

By default, homogeneous variances are assumed for the residuals and (if included) random effects. Heterogeneity can be induced using the arguments Nks and Qs, which refer to the vectors $(n_1, n_2, \ldots, n_k)$ and $(c_1, c_2, \ldots, c_q)$, respectively. See CLME-package for further explanation the model and these values.

See w.stat and lrt.stat for more details on using custom test statistics.

See Also

CLME-package clme

Examples

```
## Not run:
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )

clme.out <- clme_resids(mcv ~ time + temp + sex + (1|id), data = rat.blood )

## End(Not run)
```

---

confint.clme *Individual confidence intervals*

Description

Calculates confidence intervals for fixed effects parameter estimates in objects of class clme.

Calculates confidence intervals for fixed effects parameter estimates in objects of class clme.

Usage

```
## S3 method for class 'clme'
confint(object, parm, level = 0.95, ...)

## S3 method for class 'summary.clme'
confint(object, parm, level = 0.95, ...)
```

Arguments

object	object of class clme.
parm	parameter for which confidence intervals are computed (not used).
level	nominal confidence level.
...	space for additional arguments.

Details

Confidence intervals are computed using Standard Normal critical values. Standard errors are taken from the covariance matrix of the unconstrained parameter estimates.

Value

Returns a matrix with two columns named lcl and ucl (lower and upper confidence limit).

See Also

CLME-package clme

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

confint( clme.out )
```

---

create.constraints *Generate common order constraints*

Description

Automatically generates the constraints in the format used by clme. Allowed orders are simple, simple tree, and umbrella orders.

Usage

```
create.constraints(P1, constraints)
```

Arguments

P1	the length of θ_1, the vector constrained coefficients.
constraints	List with the elements order, node, and decreasing. See Details for further information.

Details

The elements of `constraints` are:

- `order`: string. Currently "simple", "simple.tree" and "umbrella" are supported.
- `node`: numeric, the node of the coefficients (unnecessary for simple orders).
- `decreasing`: logical. For simple orders, is the trend decreasing? For umbrella and simple tree, does the nodal parameter have the greatest value (e.g., the peak, instead of the valley)?

See `clme` for more information and a depiction of these three elements.

Value

The function returns a list containing the elements of input argument `constraints` as well as

- `A` matrix of dimension $r \times 2$ containing the order constraints, where r is the number of linear constraints.
- `B` matrix containing the contrasts necessary for computation of the Williams' type test statistic (may be identical to `A`).
- `Anull` matrix similar to `A` which defines all possible constraints. Used to obtain parameter estimates under the null hypothesis.
- `order` the input argument for `constraints\$order`.
- `node` the input argument for `constraints\$node`.
- `decreasing` the input argument for `constraints\$decreasing`

See `w.stat` for more information on `B`

Note

The function `clme` also utilizes the argument `constraints`. For `clme`, this argument may either be identical to the argument of this function, or may be the output of `create.constraints` (that is, a list containing appropriate matrices `A`, `Anull`, and if necessary, `B`).

An example the the `A` matrix might be:

[1,]	[,1]	[,2]
[2,]	1	2
[3,]	2	3
[4,]	4	3
[5,]	5	4
[6,]	6	5

This matrix defines what **CLME** describes as a decreasing umbrella order. The first row defines the constraint that $\theta_1 \leq \theta_2$, the second row defined the constraint $\theta_2 \leq \theta_3$, the third row defines $\theta_4 \leq \theta_3$, and so on. The values are indexes, and the left column is the index of the parameter constrained to be smaller.

See Also

clme, w.stat

Examples

```
## Not run:
  # For simple order, the node does not matter
  create.constraints( P1 = 5, constraints = list( order='simple' ,
                                                  decreasing=FALSE ))

  # Compare constraints against decreasing=TRUE
  create.constraints( P1 = 5, constraints=list( order='simple' ,
                                                decreasing=TRUE ))

  # Umbrella order
  create.constraints( P1 = 5, constraints=list( order='umbrella' , node=3
                                                , decreasing=FALSE ))

## End(Not run)
```

---

fibroid — *Fibroid Growth Study*

Description

This data set contains a subset of the data from the Fibroid Growth Study.

[,1]	ID	ID for subject.
[,2]	fid	ID for fibroid (each women could have multiple fibroids).
[,3]	lfgr	log fibroid growth rate. See details.
[,4]	age	age category Younger, Middle, Older.
[,5]	loc	location of fibroid, corpus, fundus, or lower segment.
[,6]	bmi	body mass index of subject.
[,7]	preg	parity, whether the subject had delivered a child.
[,8]	race	race of subject (Black or White only).
[,9]	vol	initial volume of fibroid.

Usage

```
data(fibroid)
```

Format

A data frame containing 240 observations on 9 variables.

Details

The response variable `lfgr` was calculated as the change in log fibroid volume, divided by the length of time between measurements. The growth rates were averaged to produce a single value for each fibroid, which was scaled to represent a 6-month percent change in volume.

References

Peddada, Laughlin, Miner, Guyon, Haneke, Vahdat, Semelka, Kowalik, Armao, Davis, and Baird(2008). Growth of Uterine Leiomyomata Among Premenopausal Black and White Women. Proceedings of the National Academy of Sciences of the United States of America, 105(50), 19887-19892. URL http://www.pnas.org/content/105/50/19887.full.pdf.

---

fixef.clme *Extract fixed effects*

Description

Extracts the fixed effects estimates from objects of class `clme`.

Usage

```
## S3 method for class 'clme'
fixef(object, ...)

## S3 method for class 'summary.clme'
fixef(object, ...)

## S3 method for class 'clme'
fixef(object, ...)

fixed.effects(object, ...)

## S3 method for class 'summary.clme'
fixed.effects(object, ...)

## S3 method for class 'clme'
fixed.effects(object, ...)

## S3 method for class 'clme'
coefficients(object, ...)

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

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

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

Arguments

object	object of class clme.
...	space for additional arguments

Value

Returns a numeric vector.

See Also

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

fixef( clme.out )
```

---

formula.clme *Extract formula*

Description

Extracts the formula from objects of class clme.

Usage

```
## S3 method for class 'clme'
formula(x, ...)
```

Arguments

x	object of class clme.
...	space for additional arguments

Details

The package **CLME** parametrizes the model with no intercept term. If an intercept was included, it will be removed automatically.

Value

Returns a formula object

See Also

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

formula( clme.out )
```

---

is.clme *Constructor method for objects S3 class clme*

Description

Test if an object is of class `clme` or coerce an object to be such.

Usage

```
is.clme(x)

as.clme(x, ...)
```

Arguments

x	list with the elements corresponding to the output of clme.
...	space for additional arguments.

Value

Returns an object of the class `clme`.

See Also

CLME-package clme

Examples

```
data( rat.blood )

cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

is.clme( clme.out )
as.clme( clme.out )
```

`logLik.clme` *Log-likelihood*

Description

Computes the log-likelihood of the fitted model for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
logLik(object, ...)

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

Arguments

`object`	object of class `clme`.
`...`	space for additional arguments

Details

The log-likelihood is computed using the Normal distribution. The model uses residual bootstrap methodology, and Normality is neither required nor assumed. Therefore the log-likelihood may not be a useful measure in the context of **CLME**.

Value

Numeric.

See Also

`CLME-package` `clme`

`logLik.clme`

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

logLik( clme.out )
```

`lrt.stat` *Likelihood ratio type statistic (global)*

Description

Calculates the likeihood ratio type test statistic (under Normality assumption) for a constrained linear mixed effects model. This is the default test statistic for **CLME**.

Usage

```
lrt.stat(theta, theta.null, cov.theta, ...)
```

Arguments

`theta`	estimated coefficients.
`theta.null`	coefficients estimated under the null hypothesis.
`cov.theta`	covariance matrix of the (unconstrained) coefficients.
`...`	additional arguments, to enable custom test statistic functions.

Value

Output is a numeric value.

Note

This is an internal function, unlikely to be useful outside of CLME-package. To define custom functions, the arguments available are:

`theta`, `theta.null`, `cov.theta`, `B`, `A`, `Y`, `X1`, `X2`, `U`, `tsq`, `ssq`, `Nks`, and `Qs`.

Of the additional arguments, `B` and `A` are identical to those produced by `create.constraints`. The rest, `Y`, `X1`, `X2`, `U`, `tsq`, , `ssq`, `Nks`, and `Qs`, are equivalent to arguments to `clme_em`.

Custom functions must produce numeric output. Output may have length greater than 1, which corresponds to testing multiple global hypotheses.

See Also

`clme_em`, `w.stat`

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )

clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

# Individually compute lrt statistic
lrt.stat(clme.out$theta, clme.out$theta.null, clme.out$cov.theta )
```

---

minque — *MINQUE Algorithm*

Description

Algorithm to obtain MINQUE estimates of variance components of a linear mixed effects model.

Usage

```
minque(
  Y,
  X1,
  X2 = NULL,
  U = NULL,
  Nks = dim(X1)[1],
  Qs = dim(U)[2],
  mq.eps = 1e-04,
  mq.iter = 500,
  verbose = FALSE,
  ...
)
```

Arguments

Y	$N \times 1$ vector of response data.
X1	$N \times p_1$ design matrix.
X2	optional $N \times p_2$ matrix of covariates.
U	optional $N \times c$ matrix of random effects.
Nks	optional $K \times 1$ vector of group sizes.
Qs	optional $Q \times 1$ vector of group sizes for random effects.
mq.eps	criterion for convergence for the MINQUE algorithm.
mq.iter	maximum number of iterations permitted for the MINQUE algorithm.
verbose	if TRUE, function prints messages on progress of the MINQUE algorithm.
...	space for additional arguments.

Details

By default, the model assumes homogeneity of variances for both the residuals and the random effects (if included). See the Details in `clme_em` for more information on how to use the arguments `Nks` and `Qs` to permit heterogeneous variances.

Value

The function returns a vector of the form $(\tau_1^2, \tau_2^2, \ldots, \tau_q^2, \sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2)'$. If there are no random effects, then the output is just $(\sigma_1^2, \sigma_2^2, \ldots, \sigma_k^2)'$.

Note

This function is called by several other function in **CLME** to obtain estimates of the random effect variances. If there are no random effects, they will not call `minque`.

Examples

```
data( rat.blood )

model_mats <- model_terms_clme( mcv ~ time + temp + sex + (1|id) ,
                                data = rat.blood )
Y  <- model_mats$Y
X1 <- model_mats$X1
X2 <- model_mats$X2
U  <- model_mats$U

# No covariates or random effects
minque(Y = Y, X1 = X1 )

# Include covariates and random effects
minque(Y = Y, X1 = X1, X2 = X2, U = U )
```

`model.matrix.clme` *Extract the model design matrix.*

Description

Extracts the fixed-effects design matrix from objects of class `clme`.

Usage

```
## S3 method for class 'clme'
model.matrix(object, type = "fixef", ...)

## S3 method for class 'summary.clme'
model.matrix(object, ...)
```

Arguments

object	an object of class clme.
type	specify whether to return the fixed-effects or random-effects matrix.
...	space for additional arguments

Value

Returns a matrix.

See Also

CLME-package clme

model.matrix.clme

Examples

```
## Not run:
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

model.matrix( clme.out )

## End(Not run)
```

---

model_terms_clme *Create model matrices for* clme

Description

Parses formulas to creates model matrices for clme.

Usage

```
model_terms_clme(formula, data)
```

Arguments

formula	a formula defining a linear fixed or mixed effects model. The constrained effect(s) must come before any unconstrained covariates on the right-hand side of the expression. The first ncon terms will be assumed to be constrained.
data	data frame containing the variables in the model.

Value

A list with the elements:

Y	response variable
X1	design matrix for constrained effect
X2	design matrix for covariates
P1	number of constrained coefficients
U	matrix of random effects
formula	the final formula call (automatically removes intercept)
dframe	the dataframe containing the variables in the model
REidx	an element to define random effect variance components
REnames	an element to define random effect variance components

Note

The first term on the right-hand side of the formula should be the fixed effect with constrained coefficients. Random effects are represented with a vertical bar, so for example the random effect `U` would be included by `Y ~ X1 + (1|U)`.

The intercept is removed automatically. This is done to ensure that parameter estimates are of the means of interest, rather than being expressed as a mean with offsets.

See Also

`CLME-package` `clme`

Examples

```
data( rat.blood )
model_terms_clme( mcv ~ time + temp + sex + (1|id) , data = rat.blood )
```

`nobs.clme` *Number of observations*

Description

Obtains the number of observations used to fit an model for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
nobs(object, ...)

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

Arguments

`object`	an object of class `clme`.
`...`	space for additional arguments

Value

Numeric.

See Also

CLME-package clme

nobs.clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

nobs( clme.out )
```

`plot.clme` *S3 method to plot objects of class* `clme`

Description

Generates a basic plot of estimated coefficients which are subject to constraints (θ_1). Lines indicate individual constraints (not global tests) and significance.

Usage

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

Arguments

`x`	object of class 'clme' to be plotted.
`...`	additional plotting arguments.

Note

While it is possible to plot the output of a clme fit, this will only plot the fitted means. To indicate significance, plotting must be performed on the summary of a clme fit. This method will change the class so that plot.summary.clme will be called properly.

See Also

CLME-package clme plot.summary.clme

Examples

```
## Not run:
  set.seed( 42 )
  data( rat.blood )
  cons <- list(order = "simple", decreasing = FALSE, node = 1 )
  clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                   constraints = cons, seed = 42, nsim = 10)
  plot( clme.out )

## End(Not run)
```

---

plot.summary.clme *S3 method to plot objects of class* clme

Description

Generates a basic plot of estimated coefficients which are subject to constraints (θ_1). Lines indicate individual constraints (not global tests) and significance.

Usage

```
## S3 method for class 'summary.clme'
plot(
  x,
  alpha = 0.05,
  legendx = "below",
  inset = 0.01,
  ci = FALSE,
  ylim = NULL,
  cex = 1.75,
  pch = 21,
  bg = "white",
  xlab = expression(paste("Component of ", theta[1])),
  ylab = expression(paste("Estimated Value of ", theta[1])),
  tree = NULL,
  ...
)
```

Arguments

x	object of class 'clme' to be plotted.
alpha	significance level of the test.

legendx	character indicating placement of legend. See Details.
inset	inset distance(s) from the margins as a fraction of the plot region when legend is placed by keyword.
ci	plot individual confidence intervals.
ylim	limits of the y axis.
cex	size of plotting symbols.
pch	plotting symbols.
bg	background (fill) color of the plotting symbols.
xlab	label of the x axis.
ylab	label of the y axis.
tree	logical to produce alternate graph for tree ordering.
...	additional plotting arguments.

Details

All of the individual contrasts in the `constraints\$A` matrix are tested and plotted. The global test is not represented (unless it happens to coincide with an individual contrast). Only the elements of θ which appear in any constraints (e.g. the elements of θ_1) are plotted. Coefficients for the covariates are not plotted. Solid lines denote no significant difference, while dashed lines denote statistical significance. Significance is determined by the individual p-value being less than or equal to the supplied α threshold. By default a legend denoting the meaning of solid and dashed lines will be placed below the graph. Argument `legendx` may be set to a legend keyword (e.g. `legend=''bottomright''`) to place it inside the graph at the specified location. Setting `legendx` to `FALSE` or to a non-supported keyword suppresses the legend. Confidence intervals for the coefficients may be plotted. They are individual confidence intervals, and are computed using the covariance matrix of the unconstrained estimates of θ_1. These confidence intervals have higher coverage probability than the nominal value, and as such may appear to be in conflict with the significance tests. Alternate forms of confidence intervals may be provided in future updates.#'

See Also

CLME-package clme

Examples

```
## Not run:
  set.seed( 42 )
  data( rat.blood )
  cons <- list(order = "simple", decreasing = FALSE, node = 1 )
  clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                   constraints = cons, seed = 42, nsim = 10)
  clme.out2 <- summary( clme.out )
  plot( clme.out2 )

## End(Not run)
```

print.clme *Printout of fitted object.*

Description

Prints basic information on a fitted object of class `clme`.

Usage

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

Arguments

`x`	an object of class `clme`.
`...`	space for additional arguments

Value

Text printed to console.

See Also

CLME-package clme

Examples

```
## Not run:
data( rat.blood )
set.seed( 42 )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 10)

print( clme.out )

## End(Not run)
```

print.summary.clme — *S3 method to print a summary for objects of class* `clme`

Description

Summarizes the output of objects of class `clme`, such as those produced by `clme`. Prints a tabulated display of global and individual tests, as well as parameter estimates.

Usage

```
## S3 method for class 'summary.clme'
print(x, alpha = 0.05, digits = 4, ...)
```

Arguments

`x`	an object of class `clme`.
`alpha`	level of significance.
`digits`	number of decimal digits to print.
`...`	additional arguments passed to other functions.

Value

`NULL`, just prints results to the console.

Note

The individual tests are performed on the specified order. If no specific order was specified, then the individual tests are performed on the estimated order.

See Also

`CLME-package` `clme`

Examples

```
## Not run:
  set.seed( 42 )
  data( rat.blood )
  cons <- list(order = "simple", decreasing = FALSE, node = 1 )
  clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                   constraints = cons, seed = 42, nsim = 10)

  summary( clme.out )

## End(Not run)
```

print.varcorr_clme *Printout for variance components*

Description

Prints variance components of an objects of clme.

Usage

```
## S3 method for class 'varcorr_clme'
print(object, rdig = 5, ...)
```

Arguments

object	object of class clme.
rdig	number of digits to round to.
...	space for additional arguments.

Value

Text printed to console.

See Also

CLME-package clme

Examples

```
## Not run:
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

print.varcorr_clme( clme.out )

## End(Not run)
```

random.effects *Extract random effects*

Description

Extract random effects

Extract random effects

Usage

```
random.effects(object, ...)

## S3 method for class 'summary.clme'
random.effects(object, ...)
```

Arguments

object	object of class clme.
...	space for additional arguments

ranef.clme *Extract random effects*

Description

Extracts the random effects estimates from objects of class `clme`.

Usage

```
## S3 method for class 'clme'
ranef(object, ...)

## S3 method for class 'summary.clme'
ranef(object, ...)

## S3 method for class 'clme'
ranef(object, ...)

## S3 method for class 'clme'
random.effects(object, ...)
```

Arguments

object	object of class clme.
...	space for additional arguments

Value

Returns a numeric vector.

See Also

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

ranef( clme.out )
```

---

rat.blood *Experiment on mice*

Description

This data set contains the data from an experiment on 24 Sprague-Dawley rats from Cora et al (2012).

[,1]	id	ID for rat (factor).
[,2]	time	time period (in order, 0 , 6, 24, 48, 72, 96 hours).
[,3]	temp	storage temperature reference ("Ref") vs. room temperature ("RT").
[,4]	sex	sex, male ("Male") vs. female ("Female"). Coded as "Female"=1.
[,5]	wbc	white blood cell count ($10^3/\mu L$).
[,6]	rbc	red blood cell count)$10^6/\mu L$).
[,7]	hgb	hemoglobin concentration (g/dl).
[,8]	hct	hematocrit (%).
[,9]	spun	(HCT %).
[,10]	mcv	MCV, a measurement of erythrocyte volume (fl).
[,11]	mch	mean corpuscular hemoglobin (pg).
[,12]	mchc	mean corpuscular hemoglobin concentration (g/dl).
[,13]	plts	platelet count ($10^3/\mu L$).

Usage

```
data(rat.blood)
```

Format

A data frame containing 241 observations on 13 variables.

Details

The response variable lfgr was calculated as the change in log fibroid volume, divided by the length of time between measurements. The growth rates were averaged to produce a single value for each fibroid, which was scaled to represent a 6-month percent change in volume.

References

Cora M, King D, Betz L, Wilson R, and Travlos G (2012). Artifactual changes in Sprauge-Dawley rat hematologic parameters after storage of samples at 3 C and 21 C. Journal of the American Association for Laboratory Animal Science, 51(5), 616-621. URL http://www.ncbi.nlm.nih.gov/pmc/articles/PMC3447451/.

---

residuals.clme *Various types of residuals*

Description

Computes several types of residuals for objects of class clme.

Usage

```
## S3 method for class 'clme'
residuals(object, type = "FM", ...)

## S3 method for class 'summary.clme'
residuals(object, type = "FM", ...)
```

Arguments

object	object of class clme.
type	type of residual (for mixed-effects models only).
...	space for additional arguments

Details

For fixed-effects models $Y = X\beta + \epsilon$, residuals are given as

$$\hat{e} = Y - X\hat{\beta}$$

. For mixed-effects models $Y = X\beta + +U\xi + \epsilon$, three types of residuals are available. $PA = Y - X\hat{\beta} \backslash SS = U\hat{\xi} \backslash FM = Y - X\hat{\beta} - U\hat{\xi}$

Value

Returns a numeric matrix.

See Also

CLME-package clme

Examples

```
## Not run:
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

residuals( clme.out, type='PA' )

## End(Not run)
```

---

resid_boot *Obtain Residual Bootstrap*

Description

Generates bootstrap samples of the data vector.

Usage

```
resid_boot(
  formula,
  data,
  gfix = NULL,
  eps = NULL,
  xi = NULL,
  null.resids = TRUE,
  theta = NULL,
  ssq = NULL,
  tsq = NULL,
  cov.theta = NULL,
  seed = NULL,
  nsim = 1000,
  mySolver = "LS",
  ...
)
```

Arguments

formula	a formula expression. The constrained effect(s) must come before any unconstrained covariates on the right-hand side of the expression. The first ncon terms will be assumed to be constrained.
data	data frame containing the variables in the model.

gfix	optional vector of group levels for residual variances. Data should be sorted by this value.
eps	estimates of residuals.
xi	estimates of random effects.
null.resids	logical indicating if residuals should be computed under the null hypothesis.
theta	estimates of fixed effects coefficients. Estimated if not submitted.
ssq	estimates of residual variance components. Estimated if not submitted.
tsq	estimates of random effects variance components. Estimated if not submitted.
cov.theta	covariance matrix of fixed effects coefficients. Estimated if not submitted.
seed	set the seed for the RNG.
nsim	number of bootstrap samples to use for significance testing.
mySolver	solver to use, passed to activeSet.
...	space for additional arguments.

Details

If any of the parameters theta, ssq, tsq, eps, or xi are provided, the function will use those values in generating the bootstrap samples. They will be estimated if not submitted. Ifnull.resids=TRUE, then theta will be projected onto the space of the null hypothesis ($H_0 : \theta_1 = \theta_2 = ... = \theta_{p_1}$) regardless of whether it is provided or estimated. To generate bootstraps with a specific theta, set null.residuals=FALSE.

Value

Output is $N\ timesnsim$ matrix, where each column is a bootstrap sample of the response data Y.

Note

This function is primarily designed to be called by clme.

By default, homogeneous variances are assumed for the residuals and (if included) random effects. Heterogeneity can be induced using the arguments Nks and Qs, which refer to the vectors $(n_1, n_2, \ldots, n_k)$ and $(c_1, c_2, \ldots, c_q)$, respectively. See clme_em for further explanation of these values.

See Also

clme

Examples

```
data( rat.blood )
boot_sample <- resid_boot(mcv ~ time + temp + sex + (1|id), nsim = 10,
                          data = rat.blood, null.resids = TRUE  )
```

shiny_clme *Shiny GUI for CLME*

Description

Opens a graphical user interface to run **CLME**, built from the **shiny** package.

The UI for the shiny app in CLME

The server for the shiny app in CLME

Usage

```
shiny_clme()

shinyUI_clme

shinyServer_clme(input, output)
```

Arguments

`input`	input from GUI.
`output`	output to GUI.

Format

An object of class `shiny.tag.list` (inherits from `list`) of length 3.

Details

Currently the GUI does not allow specification of custom orders for the alternative hypothesis. Future versions may enable this capability. The data should be a CSV or table-delimited file with the first row being a header. Variables are identified using their column letter or number (e.g., 1 or A). Separate multiple variables with a comma (e.g., 1,2,4 or A,B,D), or select a range of variables with a dash (e.g., 1-4 or A-D). Set to 'None' (default) to indicate no covariates or random effects. If group levels for the constrained effect are character, they may not be read in the proper order. An extra column may contain the ordered group levels (it may therefore have different length than the rest of the dataset).

Note

This function is primarily designed to call `clme`.

Examples

```
## Not run:  shiny_clme()
```

sigma.clme *Residual variance components*

Description

Extract residual variance components for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
sigma(object, ...)
```

Arguments

object	object of class `clme`.
...	space for additional arguments

Value

Numeric.

See Also

`CLME-package` `clme`

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

sigma( clme.out )
```

sigma.summary.clme *Residual variance components*

Description

Extract residual variance components for objects of class `clme`.

Usage

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

Arguments

`object`	object of class `clme`.
`...`	space for additional arguments

Value

Numeric.

See Also

`CLME-package` `clme`

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

sigma( clme.out )
```

`summary.clme` *Produce summary values for objects of class* `clme`

Description

Summarizes the output of objects of class `clme`, such as those produced by `clme`.

Usage

```
## S3 method for class 'clme'
summary(object, nsim = 1000, seed = NULL, verbose = c(FALSE, FALSE), ...)
```

Arguments

`object`	an object of class `clme`.
`nsim`	the number of bootstrap samples to use for inference.
`seed`	the value for the seed of the random number generator.
`verbose`	vector of logicals. First element will print progress for bootstrap test, second element is passed to the EM algorithm for every bootstrap sample.
`...`	additional arguments passed to other functions.

Value

The output of summary.clme is an object of the class summary.clme. This is a list containing the input object (of class clme), along with elements:

p.value	p-value for the global hypothesis
p.value.ind	p-values for each of the constraints

See Also

CLME-package clme

Examples

```
## Not run:
  set.seed( 42 )
  data( rat.blood )
  cons <- list(order = "simple", decreasing = FALSE, node = 1 )
  clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                   constraints = cons, seed = 42, nsim = 10)

  summary( clme.out )

## End(Not run)
```

VarCorr *Variance components*

Description

Extracts variance components for objects of class clme.

Usage

```
VarCorr(x, sigma, rdig)

## S3 method for class 'summary.clme'
VarCorr(x, sigma, rdig)

## S3 method for class 'clme'
VarCorr(x, sigma, rdig)
```

Arguments

x	object of class summary.clme.
sigma	(unused at present).
rdig	number of digits to round to (unused at present).

Value

Numeric.

See Also

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

VarCorr( clme.out )
```

---

vcov.clme *Variance-covariance matrix*

Description

Extracts variance-covariance matrix for objects of class `clme`.

Usage

```
## S3 method for class 'clme'
vcov(object, ...)

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

Arguments

`object`	object of class clme.
`...`	space for additional arguments

Value

Numeric matrix.

See Also

CLME-package clme

Examples

```
data( rat.blood )
cons <- list(order = "simple", decreasing = FALSE, node = 1 )
clme.out <- clme(mcv ~ time + temp + sex + (1|id), data = rat.blood ,
                 constraints = cons, seed = 42, nsim = 0)

vcov( clme.out )
```

---

w.stat *Williams' Type Test Statistic.*

Description

Calculates a Williams' type test statistic for a constrained linear mixed effects model.

Usage

```
w.stat(theta, cov.theta, B, A, ...)

w.stat.ind(theta, cov.theta, B, A, ...)
```

Arguments

`theta`	estimated coefficients.
`cov.theta`	covariance matrix of the (unconstrained) coefficients.
`B`	matrix to obtain the global contrast.
`A`	matrix of linear constraints.
`...`	additional arguments, to enable custom test statistic functions.

Details

See `create.constraints` for an example of `A`. Argument `B` is similar, but defines the global contrast for a Williams' type test statistic. This is the largest hypothesized difference in the constrained coefficients. So for an increasing simple order, the test statistic is the difference between the two extreme coefficients, θ_1 and θ_{p_1}, divided by the standard error (unconstrained). For an umbrella order order, two contrasts are considered, θ_1 to θ_s, and θ_{p_1} to θ_s, each divided by the appropriate unconstrained standard error. A general way to express this statistic is:

$$W = max\theta_{B[i,2]} - \theta_{B[i,1]}/sqrt(VAR(\theta_{B[i,2]} - \theta_{B[i,1]}))$$

where the numerator is the difference in the constrained estimates, and the standard error in the denominator is based on the covariance matrix of the unconstrained estimates.

The function `w.stat.ind` does the same, but uses the `A` matrix which defines all of the individual constraints, and returns a test statistic for each constraints instead of taking the maximum.

Value

Output is a numeric value.

Note

See lrt.stat for information on creating custom test statistics.

Examples

```
theta  <- exp(1:4/4)
th.cov <- diag(4)
X1     <- matrix( 0 , nrow=1 , ncol=4 )
const  <- create.constraints( P1=4 , constraints=list(order='simple' ,
                                                    decreasing=FALSE) )

w.stat( theta , th.cov , const$B , const$A )

w.stat.ind( theta , th.cov , const$B , const$A )
```

Index