Package ‘pbkrtest’

October 14, 2022

Version 0.5.1

Title Parametric Bootstrap, Kenward-Roger and Satterthwaite Based Methods for Test in Mixed Models

Maintainer Søren Højsgaard <sorenh@math.aau.dk>

Description Test in mixed effects models. Attention is on mixed effects models as implemented in the 'lme4' package. For linear mixed models, this package implements (1) a parametric bootstrap test, (2) a Kenward-Roger-typ modification of F-tests for linear mixed effects models and (3) a Satterthwaite-type modification of F-tests for linear mixed effects models. The package also implements a parametric bootstrap test for generalized linear mixed models. The facilities of the package are documented in the paper by Halekoh and Højsgaard, (2012, <doi:10.18637/jss.v059.i09>). Please see 'citation(\'pbkrtest\')' for citation details.

URL https://people.math.aau.dk/~sorenh/software/pbkrtest/

Depends R (>= 3.5.0), lme4 (>= 1.1.10)
Imports broom, dplyr, magrittr, MASS, Matrix (>= 1.2.3), methods, numDeriv, parallel, knitr

Encoding UTF-8

ZipData no

License GPL (>= 2)

ByteCompile Yes

RoxygenNote 7.1.1

LazyData true

VignetteBuilder knitr

NeedsCompilation no

Author Ulrich Halekoh [aut, cph],
            Søren Højsgaard [aut, cre, cph]

Repository CRAN

Date/Publication 2021-03-09 09:50:08 UTC
Description

Yield and sugar percentage in sugar beets from a split plot experiment. The experimental layout was as follows: There were three blocks. In each block, the harvest time defines the "whole plot" and the sowing time defines the "split plot". Each plot was 25m² and the yield is recorded in kg. See 'details' for the experimental layout. The data originates from a study carried out at The Danish Institute for Agricultural Sciences (the institute does not exist any longer; it became integrated in a Danish university).

Usage

beets

Format

A dataframe with 5 columns and 30 rows.

Details

Experimental plan
Sowing times
1 4. april
2 12. april
3 21. april
4 29. april
5 18. may
Harvest times
1 2. october
2 21. october
Plot allocation:
data-budworm

Block 1 | Block 2 | Block 3
+----------|---------|----------+
Plot 1 1 1 1 | 2 2 2 2 | 1 1 1 1 | Harvest time
1-15 3 4 5 2 | 3 2 4 5 | 1 5 2 3 4 | Sowing time
|----------|---------|----------|
Plot 2 2 2 2 | 1 1 1 1 | 2 2 2 2 | Harvest time
16-30 2 1 5 4 | 4 1 3 2 | 1 4 3 2 5 | Sowing time
|----------|---------|----------|

References

Examples

data(beets)

beets$bh <- with(beets, interaction(block, harvest))
summary(aov(yield ~ block + sow + harvest + Error(bh), beets))
summary(aov(sugpct ~ block + sow + harvest + Error(bh), beets))

Description
Effect of Insecticide on survival of tobacco budworms number of killed budworms exposed to an insecticidepp mortality of the moth tobacco budworm 'Heliothis virescens' for 6 doses of the pyrethroid trans-cypermethrin differentiated with respect to sex

Usage
budworm

Format
This data frame contains 12 rows and 4 columns:

sex: sex of the budworm
dose: dose of the insecticide trans-cypermethrin in [µg]
dead: budworms killed in a trial
ntotal: total number of budworms exposed per trial

Source
References


Examples

data(budworm)

## function to calculate the empirical logits
empirical.logit<- function(nevent, ntotal) {
  y <- log((nevent + 0.5) / (ntotal - nevent + 0.5))
  y
}

# plot the empirical logits against log-dose
log.dose <- log(budworm$dose)
emp.logit <- empirical.logit(budworm$ndead, budworm$ntotal)
plot(log.dose, emp.logit, type="n", xlab="log-dose", ylab="empirical logit")
title("budworm: empirical logits of probability to die")
male <- budworm$sex=="male"
female <- budworm$sex=="female"
lines(log.dose[male], emp.logit[male], type="b", lty=1, col=1)
lines(log.dose[female], emp.logit[female], type="b", lty=2, col=2)
legend(0.5, 2, legend=c("male", "female"), lty=c(1,2), col=c(1,2))

## Not run:
* SAS example;
data budworm;
infile 'budworm.txt' firstobs=2;
input sex dose ndead ntotal;
run;

## End(Not run)

---

**getkr**  
*Extract (or "get") components from a KRmodcomp object.*

**Description**

Extract (or "get") components from a KRmodcomp object, which is the result of the KRmodcomp function.
**Usage**

```r
getKR(
  object,
  name = c("ndf", "ddf", "Fstat", "p.value", "F.scaling", "FstatU", "p.valueU", "aux")
)
```

**Arguments**

- **object**
  A `KRmodcomp` object, which is the result of the `KRmodcomp` function.
- **name**
  The available slots. If `name` is missing or `NULL` then everything is returned.

**Author(s)**

Søren Højsgaard <sorenh@math.aau.dk>

**References**


**See Also**

- `KRmodcomp`, `PBmodcomp`, `vcovAdj`

**Examples**

```r
data(beets, package='pbkrtest')
lg <- lmer(sugpct ~ block + sow + harvest + (1|block:harvest),
  data=beets, REML=FALSE)
sm <- update(lg, .~. - harvest)
modcomp <- KRmodcomp(lg, sm)
getKR(modcomp, "ddf") # get denominator degrees of freedom.
```

---

**Description**

`get_ddf_Lb`

*Adjusted denominator degrees freedom for linear estimate for linear mixed model.*

Get adjusted denominator degrees freedom for testing Lb=0 in a linear mixed model where L is a restriction matrix.
Usage

get_Lb_ddf(object, L)

## S3 method for class 'lmerMod'
get_Lb_ddf(object, L)

get_ddf_Lb(object, Lcoef)

## S3 method for class 'lmerMod'
get_ddf_Lb(object, Lcoef)

Lb_ddf(L, V0, Vadj)

ddf_Lb(VVa, Lcoef, VV0 = VVa)

Arguments

object A linear mixed model object.

L A vector with the same length as fixef(object) or a matrix with the same number of columns as the length of fixef(object)

Lcoef Linear contrast matrix

V0, Vadj Unadjusted and adjusted covariance matrix for the fixed effects parameters. Unadjusted covariance matrix is obtained with vcov() and adjusted with vcovAdj().

VVa Adjusted covariance matrix

VV0 Unadjusted covariance matrix

Value

Adjusted degrees of freedom (adjustment made by a Kenward-Roger approximation).

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>

References


See Also

KRmodcomp, vcovAdj, model2remat, remat2model
Examples

(fmLarge <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
## removing Days
(fmSmall <- lmer(Reaction ~ 1 + (Days|Subject), sleepstudy))
anova(fmLarge, fmSmall)
KRmodcomp(fmLarge, fmSmall)  ## 17 denominator df's
get_Lb_ddf(fmLarge, c(0,1))  ## 17 denominator df's

# Notice: The restriction matrix L corresponding to the test above
# can be found with
L <- model2remat(fmLarge, fmSmall)
L

internal                  Internal functions for the pbkrtest package

Description

These functions are not intended to be called directly.

internal-pbkrtest         pbkrtest internal

Description

pbkrtest internal

kr-modcomp                 F-test and degrees of freedom based on Kenward-Roger approximation

Description

An approximate F-test based on the Kenward-Roger approach.

Usage

KRmodcomp(largeModel, smallModel, betaH = 0, details = 0)

## S3 method for class 'lmerMod'
KRmodcomp(largeModel, smallModel, betaH = 0, details = 0)
Arguments

- `largeModel` An `lmer` model
- `smallModel` An `lmer` model or a restriction matrix
- `betaH` A number or a vector of the beta of the hypothesis, e.g. $L \beta = L \beta_H$. $\beta_H = 0$ if `modelSmall` is a model not a restriction matrix.
- `details` If larger than 0 some timing details are printed.

Details

The model object must be fitted with restricted maximum likelihood (i.e. with `REML=TRUE`). If the object is fitted with maximum likelihood (i.e. with `REML=FALSE`) then the model is refitted with `REML=TRUE` before the p-values are calculated. Put differently, the user needs not worry about this issue.

An F test is calculated according to the approach of Kenward and Roger (1997). The function works for linear mixed models fitted with the `lmer` function of the `lme4` package. Only models where the covariance structure is a sum of known matrices can be compared.

The `largeModel` may be a model fitted with `lmer` either using `REML=TRUE` or `REML=FALSE`. The `smallModel` can be a model fitted with `lmer`. It must have the same covariance structure as `largeModel`. Furthermore, its linear space of expectation must be a subspace of the space for `largeModel`. The model `smallModel` can also be a restriction matrix $L$ specifying the hypothesis $L \beta = L \beta_H$, where $L$ is a $k \times p$ matrix and $\beta$ is a $p$ column vector the same length as `fixef(largeModel)`.

The $\beta_H$ is a $p$ column vector.

Notice: if you want to test a hypothesis $L \beta = c$ with a $k$ vector $c$, a suitable $\beta_H$ is obtained via $\beta_H = L c$ where $L_n$ is a g-inverse of $L$.

Notice: It cannot be guaranteed that the results agree with other implementations of the Kenward-Roger approach!

Note

This functionality is not thoroughly tested and should be used with care. Please do report bugs etc.

Author(s)

Ulrich Halekoh <uhalekoh@health.sdu.dk>, Søren Højsgaard <sorenh@math.aau.dk>

References


See Also

getKR, lmer, vcovAdj, PBmodcomp
Examples

(fmLarge <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
## removing Days
(fmSmall <- lmer(Reaction ~ 1 + (Days|Subject), sleepstudy))
anova(fmLarge, fmSmall)
KRmodcomp(fmLarge, fmSmall)

## The same test using a restriction matrix
L <- cbind(0,1)
KRmodcomp(fmLarge, L)

## Same example, but with independent intercept and slope effects:
m.large <- lmer(Reaction ~ Days + (1|Subject) + (0+Days|Subject), data = sleepstudy)
m.small <- lmer(Reaction ~ 1 + (1|Subject) + (0+Days|Subject), data = sleepstudy)
anova(m.large, m.small)
KRmodcomp(m.large, m.small)

---

kr-vcov

Adjusted covariance matrix for linear mixed models according to Kenward and Roger

Description

Kenward and Roger (1997) describe an improved small sample approximation to the covariance matrix estimate of the fixed parameters in a linear mixed model.

Usage

vcovAdj(object, details = 0)

## S3 method for class 'lmerMod'
vcovAdj(object, details = 0)

Arguments

object
An lmer model
details
If larger than 0 some timing details are printed.

Value

phiA
the estimated covariance matrix, this has attributed P, a list of matrices used in KR_adjust and the estimated matrix W of the variances of the covariance parameters of the random effects

SigmaG
list: Sigma: the covariance matrix of Y; G: the G matrices that sum up to Sigma; n.ggamma: the number (called M in the article) of G matrices
Note

If $N$ is the number of observations, then the `vcovAdj()` function involves inversion of an $N \times N$ matrix, so the computations can be relatively slow.

Author(s)

Ulrich Halekoh <uhalekoh@health.sdu.dk>, Søren Højsgaard <sorenh@math.aau.dk>

References


See Also

getKR, KRmodcomp, lmer, PBmodcomp, vcovAdj

Examples

```r
fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy)
class(fm1)

## Here the adjusted and unadjusted covariance matrices are identical,
## but that is not generally the case:

v1 <- vcov(fm1)
v2 <- vcovAdj(fm1, details=0)
v2 / v1

## For comparison, an alternative estimate of the variance-covariance
## matrix is based on parametric bootstrap (and this is easily
## paralleled):

## Not run:
nsim <- 100
sim <- simulate(fm.ml, nsim)
B <- lapply(sim, function(newy) try(fixef(refit(fm.ml, newresp=newy))))
B <- do.call(rbind, B)
v3 <- cov.wt(B)$cov
v2/v1
v3/v1

## End(Not run)
```
Conversion between a model object and a restriction matrix

Description

Testing a small model under a large model corresponds imposing restrictions on the model matrix of the larger model and these restrictions come in the form of a restriction matrix. These functions converts a model to a restriction matrix and vice versa.

Usage

```r
model2remat(largeModel, smallModel, sparse = FALSE)
remat2model(largeModel, L, REML = TRUE, ...)
make_modelmat(X, L)
make_remat(X, X2)
```

Arguments

- `largeModel, smallModel`: Model objects of the same "type". Possible types are linear mixed effects models and linear models (including generalized linear models)
- `sparse`: Should the restriction matrix be sparse or dense?
- `L`: A restriction matrix; a full rank matrix with as many columns as 'X' has.
- `REML`: Controls if new model object should be fitted with REML or ML.
- `...`: Additional arguments; not used.
- `X, X2`: Model matrices. Must have same number of rows.

Details

'make_remat' Make a restriction matrix. If span(X2) is in span(X) then the corresponding restriction matrix 'L' is returned.

Value

- `model2remat`: A restriction matrix.
- `remat2model`: A model object.

Note

That these functions are visible is a recent addition; minor changes may occur.

Author(s)

Ulrich Halekoh <uhalekoh@health.sdu.dk>, Søren Højsgaard <sorenh@math.aau.dk>
References


See Also

PBmodcomp, PBrefdist, KRmodcomp

Examples

library(pbkrtest)
data("beets", package = "pbkrtest")
sug <- lm(sugpct ~ block + sow + harvest, data=beets)
sug.h <- update(sug, .~.- harvest)
sug.s <- update(sug, .~.- sow)

## Construct restriction matrices from models
L.h <- model2remat(sug, sug.h); L.h
L.s <- model2remat(sug, sug.s); L.s

## Construct submodels from restriction matrices
mod.h <- remat2model(sug, L.h); mod.h
mod.s <- remat2model(sug, L.s); mod.s

## Sanity check: The models have the same fitted values and log likelihood
plot(fitted(mod.h), fitted(sug.h))
plot(fitted(mod.s), fitted(sug.s))
logLik(mod.h)
logLik(sug.h)
logLik(mod.s)
logLik(sug.s)

pb-modcomp

Model comparison using parametric bootstrap methods.

Description

Model comparison of nested models using parametric bootstrap methods. Implemented for some commonly applied model types.

Usage

PBmodcomp(
  largeModel,  # large model
  smallModel,  # small model
  nsim = 1000,  # number of bootstrap samples
  ref = NULL,  # reference model
  seed = NULL)  # random seed
## S3 method for class 'merMod'

PBmodcomp(
  largeModel,
  smallModel,
  nsim = 1000,
  ref = NULL,
  seed = NULL,
  cl = NULL,
  details = 0
)

## S3 method for class 'lm'

PBmodcomp(
  largeModel,
  smallModel,
  nsim = 1000,
  ref = NULL,
  seed = NULL,
  cl = NULL,
  details = 0
)

seqPBmodcomp(largeModel, smallModel, h = 20, nsim = 1000, cl = 1)

Arguments

largeModel  A model object. Can be a linear mixed effects model or generalized linear mixed effects model (as fitted with lmer() and glmer() function in the lme4 package) or a linear normal model or a generalized linear model. The largeModel must be larger than smallModel (see below).

smallModel  A model of the same type as largeModel or a restriction matrix.

nsim  The number of simulations to form the reference distribution.

ref  Vector containing samples from the reference distribution. If NULL, this vector will be generated using PBrefdist().

seed  A seed that will be passed to the simulation of new datasets.

c1  A vector identifying a cluster; used for calculating the reference distribution using several cores. See examples below.

details  The amount of output produced. Mainly relevant for debugging purposes.

h  For sequential computing for bootstrap p-values: The number of extreme cases needed to generate before the sampling process stops.
Details

The model object must be fitted with maximum likelihood (i.e. with \texttt{REML=FALSE}). If the object is fitted with restricted maximum likelihood (i.e. with \texttt{REML=TRUE}) then the model is refitted with \texttt{REML=FALSE} before the p-values are calculated. Put differently, the user needs not worry about this issue.

Under the fitted hypothesis (i.e. under the fitted small model) \( nsim \) samples of the likelihood ratio test statistic (LRT) are generated.

Then p-values are calculated as follows:

- **LRT**: Assuming that LRT has a chi-square distribution.
- **PBtest**: The fraction of simulated LRT-values that are larger or equal to the observed LRT value.
- **Bartlett**: A Bartlett correction of LRT is calculated from the mean of the simulated LRT-values
- **Gamma**: The reference distribution of LRT is assumed to be a gamma distribution with mean and variance determined as the sample mean and sample variance of the simulated LRT-values.
- **F**: The LRT divided by the number of degrees of freedom is assumed to be F-distributed, where the denominator degrees of freedom are determined by matching the first moment of the reference distribution.

Note

It can happen that some values of the LRT statistic in the reference distribution are negative. When this happens one will see that the number of used samples (those where the LRT is positive) are reported (this number is smaller than the requested number of samples).

In theory one can not have a negative value of the LRT statistic but in practice one can: We speculate that the reason is as follows: We simulate data under the small model and fit both the small and the large model to the simulated data. Therefore the large model represents - by definition - an overfit; the model has superfluous parameters in it. Therefore the fit of the two models will for some simulated datasets be very similar resulting in similar values of the log-likelihood. There is no guarantee that the the log-likelihood for the large model in practice always will be larger than for the small (convergence problems and other numerical issues can play a role here).

To look further into the problem, one can use the \texttt{PBrefdist()} function for simulating the reference distribution (this reference distribution can be provided as input to \texttt{PBmodcomp()}). Inspection sometimes reveals that while many values are negative, they are numerically very small. In this case one may try to replace the negative values by a small positive value and then invoke \texttt{PBmodcomp()} to get some idea about how strong influence there is on the resulting p-values. (The p-values get smaller this way compared to the case when only the originally positive values are used).

Author(s)

Søren Højsgaard <sorenh@math.aau.dk>

References

See Also

KRmodcomp, PBrefdist

Examples

data(beets, package="pbkrtest")
head(beets)

NSIM <- 50 ## Simulations in parametric bootstrap

## Linear mixed effects model:
sug <- lmer(sugpct ~ block + sow + harvest + (1|block:harvest),
            data=beets, REML=FALSE)
sug.h <- update(sug, .~. -harvest)
sug.s <- update(sug, .~. -sow)

anova(sug, sug.h)
PBMmodcomp(sug, sug.h, nsim=NSIM, cl=1)
anova(sug, sug.h)
PBMmodcomp(sug, sug.s, nsim=NSIM, cl=1)

## Linear normal model:
sug <- lm(sugpct ~ block + sow + harvest, data=beets)
sug.h <- update(sug, .~. -harvest)
sug.s <- update(sug, .~. -sow)

anova(sug, sug.h)
PBMmodcomp(sug, sug.h, nsim=NSIM, cl=1)
anova(sug, sug.s)
PBMmodcomp(sug, sug.s, nsim=NSIM, cl=1)

## Generalized linear model
counts <- c(18, 17, 15, 20, 10, 20, 25, 13, 12)
outcome <- gl(3, 1, 9)
treatment <- gl(3, 3)
d.AD <- data.frame(treatment, outcome, counts)

head(d.AD)
glm.D93 <- glm(counts ~ outcome + treatment, family = poisson())
glm.D93.o <- update(glm.D93, .~. -outcome)
glm.D93.t <- update(glm.D93, .~. -treatment)

anova(glm.D93, glm.D93.o, test="Chisq")
PBMmodcomp(glm.D93, glm.D93.o, nsim=NSIM, cl=1)
anova(glm.D93, glm.D93.t, test="Chisq")
PBMmodcomp(glm.D93, glm.D93.t, nsim=NSIM, cl=1)

## Generalized linear mixed model (it takes a while to fit these)
## Not run:
(gm1 <- glmer(cbind(incidence, size - incidence) ~ period + (1 | herd),
              data = cbpp, family = binomial))
(gm2 <- update(gm1, .~.-period))
anova(gm1, gm2)
PBMmodcomp(gm1, gm2, cl=2)

## End(Not run)

## Not run:
(fmLarge <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
## removing Days
(fmSmall <- lmer(Reaction ~ 1 + (Days|Subject), sleepstudy))
anova(fmLarge, fmSmall)
PBMmodcomp(fmLarge, fmSmall, cl=1)

## The same test using a restriction matrix
L <- cbind(0,1)
PBMmodcomp(fmLarge, L, cl=1)

## Vanilla
PBmodcomp(beet0, beet_no.harv, nsim=NSIM, cl=1)

## Simulate reference distribution separately:
refdist <- PBrefdist(beet0, beet_no.harv, nsim=1000)
PBMmodcomp(beet0, beet_no.harv, ref=refdist, cl=1)

## Do computations with multiple processors:
## Number of cores:
(nc <- detectCores())
## Create clusters
cl <- makeCluster(rep("localhost", nc))

## Then do:
PBMmodcomp(beet0, beet_no.harv, cl=cl)

## Or in two steps:
refdist <- PBrefdist(beet0, beet_no.harv, nsim=NSIM, cl=cl)
PBMmodcomp(beet0, beet_no.harv, ref=refdist)

## It is recommended to stop the clusters before quitting R:
stopCluster(cl)

## End(Not run)

## Linear and generalized linear models:
m11 <- lm(dist ~ speed + I(speed^2), data=cars)
m10 <- update(m11, ~.-I(speed^2))
anova(m11, m10)
PBMmodcomp(m11, m10, cl=1, nsim=NSIM)
PBMmodcomp(m11, ~.-I(speed^2), cl=1, nsim=NSIM)
PBMmodcomp(m11, c(0, 0, 1), cl=1, nsim=NSIM)

m21 <- glm(dist ~ speed + I(speed^2), family=Gamma("identity"), data=cars)
pb-refdist

Calculate reference distribution using parametric bootstrap

Description

Calculate reference distribution of likelihood ratio statistic in mixed effects models using parametric bootstrap

Usage

PBrefdist(
  largeModel,  
  smallModel, 
  nsim = 1000,  
  seed = NULL, 
  cl = NULL, 
  details = 0  
)

## S3 method for class 'lm'
PBrefdist(
  largeModel,  
  smallModel, 
  nsim = 1000,  
  seed = NULL, 
  cl = NULL, 
  details = 0  
)

## S3 method for class 'merMod'
PBrefdist(
  largeModel,  
  smallModel, 
  nsim = 1000,  
  seed = NULL, 
  cl = NULL, 
  details = 0  
)
Arguments

largeModel  A linear mixed effects model as fitted with the \texttt{lmer}() function in the \texttt{lme4} package. This model must be larger than smallModel (see below).

smallModel  A linear mixed effects model as fitted with the \texttt{lmer}() function in the \texttt{lme4} package. This model must be smaller than largeModel (see above).

nsim        The number of simulations to form the reference distribution.

seed        Seed for the random number generation.

cl          Used for controlling parallel computations. See sections 'details' and 'examples' below.

details     The amount of output produced. Mainly relevant for debugging purposes.

Details

The model object must be fitted with maximum likelihood (i.e. \texttt{REML=FALSE}). If the object is fitted with restricted maximum likelihood (i.e. \texttt{REML=TRUE}) then the model is refitted with \texttt{REML=FALSE} before the p-values are calculated. Put differently, the user needs not worry about this issue.

The argument 'cl' (originally short for 'cluster') is used for controlling parallel computations. 'cl' can be \texttt{NULL} (default), positive integer or a list of clusters.

Special care must be taken on Windows platforms (described below) but the general picture is this:

The recommended way of controlling cl is to specify the component pbcl in \texttt{options()} with e.g. \texttt{options("pbcl"=4)}.

If cl is \texttt{NULL}, the function will look at if the pbcl has been set in the options list with \texttt{getOption("pbcl")}.

If cl=N then N cores will be used in the computations. If cl is \texttt{NULL} then the function will look for

Value

A numeric vector

Author(s)

Søren Højsgaard <sorenh@math.aau.dk>

References


See Also

PBmodcomp, KRmodcomp
Examples

data(beets)
head(beets)
beet0 <- lmer(sugpct ~ block + sow + harvest + (1|block:harvest), data=beets, REML=FALSE)
beet_no.harv <- update(beet0, . ~ . - harvest)
rd <- PBrefdist(beet0, beet_no.harv, nsim=20, cl=1)
rd
## Not run:
## Note: Many more simulations must be made in practice.

# Computations can be made in parallel using several processors:

# 1: On OSs that fork processes (that is, not on windows):
# -----------------------------------------------------------

if (Sys.info()["sysname"] != "Windows"){
  N <- 2 ## Or N <- parallel::detectCores()

  # N cores used in all calls to function in a session
  options("mc.cores"=N)
  rd <- PBrefdist(beet0, beet_no.harv, nsim=20)

  # N cores used just in one specific call (when cl is set,
  # options("mc.cores") is ignored):
  rd <- PBrefdist(beet0, beet_no.harv, nsim=20, cl=N)
}

# In fact, on Windows, the approach above also work but only when setting the
# number of cores to 1 (so there is to parallel computing)

# In all calls:
# options("mc.cores"=1)
# rd <- PBrefdist(beet0, beet_no.harv, nsim=20)
# Just once
# rd <- PBrefdist(beet0, beet_no.harv, nsim=20, cl=1)

# 2. On all platforms (also on Windows) one can do
# -------------------------------------------------
library(parallel)
N <- 2 ## Or N <- detectCores()
clus <- makeCluster(rep("localhost", N))

# In all calls in a session
options("pb.cl"=clus)
rd <- PBrefdist(beet0, beet_no.harv, nsim=20)

# Just once:
rd <- PBrefdist(beet0, beet_no.harv, nsim=20, cl=clus)
stopCluster(clus)

## End(Not run)
Description

An approximate F-test based on the Satterthwaite approach.

Usage

```r
SATmodcomp(
  largeModel,
  smallModel,
  details = 0,
  eps = sqrt(.Machine$double.eps)
)
```

```r
## S3 method for class 'lmerMod'
SATmodcomp(
  largeModel,
  smallModel,
  details = 0,
  eps = sqrt(.Machine$double.eps)
)
```

Arguments

- `largeModel`: An `lmerMod` model.
- `smallModel`: An `lmerMod` model, a restriction matrix or a model formula. See example section.
- `details`: If larger than 0 some timing details are printed.
- `eps`: A small number.

Details

Notice: It cannot be guaranteed that the results agree with other implementations of the Satterthwaite approach!

Note

This code is greatly inspired by code in the lmerTest package. This is a recent addition to the pbkrtest package; please report unexpected behaviour.

Author(s)

Søren Højsgaard, <sorenh@math.aau.dk>
References


See Also

getKR, lmer, vcovAdj, PBmodcomp

Examples

(fm1 <- lmer(Reaction ~ Days + (Days|Subject), sleepstudy))
L1 <- cbind(0, 1)
SATmodcomp(fm1, L1)

(fm2 <- lmer(Reaction ~ Days + I(Days^2) + (Days|Subject), sleepstudy))
## Test for no effect of Days. There are three ways of using the function:
## 1) Define 2-df contrast - since L has 2 (linearly independent) rows
## the F-test is on 2 (numerator) df:
L2 <- rbind(c(0, 1, 0), c(0, 0, 1))
SATmodcomp(fm2, L2)

## 2) Use two model objects
fm3 <- update(fm2, ~. - Days - I(Days^2))
SATmodcomp(fm2, fm3)

## 3) Specify restriction as formula
SATmodcomp(fm2, ~. - Days - I(Days^2))
Index

* datasets
  data-beets, 2
  data-budworm, 3

* inference
  get_ddf_Lb, 5
  kr-modcomp, 7
  kr-vcov, 9
  pb-modcomp, 12
  pb-refdist, 17
  sat-modcomp, 20

* models
  get_ddf_Lb, 5
  kr-modcomp, 7
  kr-vcov, 9
  pb-modcomp, 12
  pb-refdist, 17
  sat-modcomp, 20

* utilities
  getkr, 4
  model-coerce, 11
  %>% (internal), 7
  as.data.frame.XXmodcomp (internal), 7
  beets (data-beets), 2
  budworm (data-budworm), 3
  data-beets, 2
  data-budworm, 3
  ddf_Lb (get_ddf_Lb), 5
  get_ddf_Lb, 5
  get_Lb_ddf (get_ddf_Lb), 5
  get_SigmaG (kr-vcov), 9
  getKR, 8, 10, 21
  getKR (getkr), 4
  getkr, 4
  getLRT (pb-modcomp), 12
  internal, 7
  internal=pbkrtest, 7
  kr-modcomp, 7
  kr-vcov, 9
  KRmodcomp, 5, 6, 10, 12, 15, 18
  KRmodcomp (kr-modcomp), 7
  KRmodcomp_init (internal), 7
  KRmodcomp_internal (kr-modcomp), 7
  Lb_ddf (get_ddf_Lb), 5
  lmer, 8, 10, 21
  LMM_Sigma_G (kr-vcov), 9
  make_modelmat (model-coerce), 11
  make_remat (model-coerce), 11
  model-coerce, 11
  model2remat, 6
  model2remat (model-coerce), 11
  pb-modcomp, 12
  pb-refdist, 17
  PBmodcomp, 5, 8, 10, 12, 18, 21
  PBmodcomp (pb-modcomp), 12
  PBrefdist, 12, 15
  PBrefdist (pb-refdist), 17
  plot.PBmodcomp (internal), 7
  plot.XXmodcomp (pb-modcomp), 12
  print.KRmodcomp (internal), 7
  print.PBmodcomp (internal), 7
  print.summaryPB (internal), 7
  remat2model, 6
  remat2model (model-coerce), 11
  sat-modcomp, 20
  SATmodcomp (sat-modcomp), 20
  seqPBmodcomp (pb-modcomp), 12
  summary.KRmodcomp (internal), 7
  summary.PBmodcomp (internal), 7
  tidy (internal), 7
  vcovAdj, 5, 6, 8, 10, 21

22
INDEX

vcovAdj (kr-vcov), 9
vcovAdj0 (kr-vcov), 9
vcovAdj2 (kr-vcov), 9
vcovAdj_internal (kr-vcov), 9