Package 'varTestnlme'

July 22, 2025

Type Package

Title Variance Components Testing for Linear and Nonlinear Mixed Effects Models

Version 1.3.5

URL https://github.com/baeyc/varTestnlme/

BugReports https://github.com/baeyc/varTestnlme/issues

Maintainer Charlotte Baey <charlotte.baey@univ-lille.fr>

Description An implementation of the Likelihood ratio Test (LRT) for testing that, in a (non)linear mixed effects model, the variances of a subset of the random effects are equal to zero. There is no restriction on the subset of variances that can be tested: for example, it is possible to test that all the variances are equal to zero. Note that the implemented test is asymptotic.
This package should be used on model fits from packages 'nlme', 'lmer', and 'saemix'.
Charlotte Baey and Estelle Kuhn (2019) <doi:10.18637/jss.v107.i06>.

License GPL (>= 2)

Encoding UTF-8

Imports mvtnorm, lmeresampler, alabama, Matrix, merDeriv, anocva, corpcor, quadprog, lme4, nlme, saemix, msm, foreach, methods, doParallel, parallel

RoxygenNote 7.2.3

Suggests knitr, rmarkdown, EnvStats

VignetteBuilder knitr

NeedsCompilation no

Author Charlotte Baey [aut, cre] (ORCID: <https://orcid.org/0000-0002-1413-1058>), Estelle Kuhn [aut]

Repository CRAN

Date/Publication 2023-09-22 15:20:02 UTC

Contents

alt.desc . . . 2
approxWeights . . . 3
bootinvFIM . . . 4
bootinvFIM.lme . . . 4
bootinvFIM.merMod . . . 5
bootinvFIM.SaemixObject . . . 5
dfChiBarSquare . . . 6
extractFIM.lme . . . 6
extractStruct . . . 7
extractStruct.lme . . . 7
extractStruct.merMod . . . 8
extractStruct.SaemixObject . . . 8
extractVarCov . . . 9
extractVarCov.lme . . . 9
extractVarCov.merMod . . . 9
fim.vctest . . . 10
null.desc . . . 10
objFunction . . . 11
pckName . . . 12
print.desc.message . . . 12
print.res.message . . . 13
print.vctest . . . 13
summary.vctest . . . 14
varCompTest . . . 14
weightsChiBarSquare . . . 17

Index **18**

`alt.desc` *alt.desc*

Description

create alternative description

Usage

```
alt.desc(msdata)
```

Arguments

`msdata`	a list containing the structure of the model and data, as an output from `extractStruct.<package_name>` functions

`approxWeights` *Monte Carlo approximation of chi-bar-square weights*

Description

The chi-bar-square distribution $\bar{\chi}^2(I, C)$ is a mixture of chi-square distributions. The function provides a method to approximate the weights of the mixture components, when the number of components is known as well as the degrees of freedom of each chi-square distribution in the mixture, and given a vector of simulated values from the target $\bar{\chi}^2(I, C)$ distribution. Note that the estimation is based on (pseudo)-random Monte Carlo samples. For reproducible results, one should fix the seed of the (pseudo)-random number generator.

Usage

```
approxWeights(x, df, q)
```

Arguments

`x`	a vector of i.i.d. random realizations of the target chi-bar-square distribution
`df`	a vector containing the degrees of freedom of the chi-squared components
`q`	the empirical quantile of `x` used to choose the $p - 2$ values $c_1, \ldots, c_{p-2}$ (see Details)

Details

Let us assume that there are p components in the mixture, with degrees of freedom between n_1 and n_p. By definition of a mixture distribution, we have :

$$P(\bar{\chi}^2(I, C) \leq c) = \sum_{i=n_1}^{n_p} w_i P(\chi_i^2 \leq c)$$

Choosing $p-2$ values $c_1, \ldots, c_{p-2}$, the function will generate a system of $p-2$ equations according to the above relationship, and add two additional relationships stating that the sum of all the weights is equal to 1, and that the sum of odd weights and of even weights is equal to 1/2, so that we end up with a system a p equations with p variables.

Value

A vector containing the estimated weights, as well as their covariance matrix.

Author(s)

Charlotte Baey <<charlotte.baey@univ-lille.fr>>

`bootinvFIM` *Approximation of the inverse of the Fisher Information Matrix via parametric bootstrap*

Description

When the FIM is not available, this function provides an approximation of the FIM based on an estimate of the covariance matrix of the model's parameters obtained via parametric bootstrap.

Usage

```
bootinvFIM(m, B = 1000, seed = 0)
```

Arguments

`m`	a fitted model that will be used as the basis of the parametric bootstrap (providing the initial maximum likelihood estimate of the parameters and the modelling framework)
`B`	the size of the bootstrap sample
`seed`	a seed for the random generator

Value

the empirical covariance matrix of the parameter estimates obtained on the bootstrap sample

Author(s)

Charlotte Baey <<charlotte.baey@univ-lille.fr>>

`bootinvFIM.lme` *Compute the inverse of the Fisher Information Matrix using parametric bootstrap*

Description

Compute the inverse of the Fisher Information Matrix using parametric bootstrap

Usage

```
## S3 method for class 'lme'
bootinvFIM(m, B = 1000, seed = 0)
```

Arguments

`m`	the model under H1
`B`	the bootstrap sample size
`seed`	a seed for the random generator

`bootinvFIM.merMod` *Compute the inverse of the Fisher Information Matrix using parametric bootstrap*

Description

Compute the inverse of the Fisher Information Matrix using parametric bootstrap

Usage

```
## S3 method for class 'merMod'
bootinvFIM(m, B = 1000, seed = 0)
```

Arguments

`m`	the model under H1
`B`	the bootstrap sample size
`seed`	a seed for the random generator

`bootinvFIM.SaemixObject` *Compute the inverse of the Fisher Information Matrix using parametric bootstrap*

Description

Compute the inverse of the Fisher Information Matrix using parametric bootstrap

Usage

```
## S3 method for class 'SaemixObject'
bootinvFIM(m, B = 1000, seed = 0)
```

Arguments

`m`	the model under H1
`B`	the bootstrap sample size
`seed`	a seed for the random generator

dfChiBarSquare *Chi-bar-square degrees of freedom computation*

Description

Computation of the degrees of freedom of the chi-bar-square

Usage

```
dfChiBarSquare(msdata)
```

Arguments

msdata	a list containing the structure of the model and data, as an output from extractStruct.<package_name> functions

Value

a list containing the vector of the degrees of freedom of the chi-bar-square and the dimensions of the cone of the chi-bar-square distribution

extractFIM.lme *Extract FIM*

Description

Extract FIM

Usage

```
extractFIM.lme(m, struct)
```

Arguments

m	the model to extract the FIM from
struct	the structure of the covariance matrix (either 'full', 'diag', or 'blockdiag)

extractStruct *Extracting models' structures*

Description

Functions extracting the structure of the models under both hypothesis: the number of fixed and random effects, the number of tested fixed and random effects, and the residual dimension, as well as the random effects covariance structure

Usage

```
extractStruct(m1, m0, randm0)
```

Arguments

m1	the model under H1
m0	the model under H0
randm0	a boolean stating whether the model under H0 contains any random effect

Value

A list with the following components:

detailStruct	a data frame containing the list of the parameters and whether they are tested or not
nameVarTested	the name of the variance components being tested
nameFixedTested	the name of the fixed effects being tested
dims	a list with the dimensions of fixed and random effects, tested or not tested
structGamma	the structure of the covariance matrix of the random effects diag, full or blockDiag

extractStruct.lme *Extract model structure*

Description

Extract model structure

Usage

```
## S3 method for class 'lme'
extractStruct(m1, m0, randm0)
```

Arguments

m1	the fit under H1
m0	the fit under H0
randm0	a boolean indicating whether random effects are present in m0

---

extractStruct.merMod *Extract model structure*

Description

Extract model structure

Usage

```
## S3 method for class 'merMod'
extractStruct(m1, m0, randm0)
```

Arguments

m1	the fit under H1
m0	the fit under H0
randm0	a boolean indicating whether random effects are present in m0

---

extractStruct.SaemixObject *Extract model structure*

Description

Extract model structure

Usage

```
## S3 method for class 'SaemixObject'
extractStruct(m1, m0, randm0)
```

Arguments

m1	the fit under H1
m0	the fit under H0
randm0	a boolean indicating whether random effects are present in m0

extractVarCov *Extract covariance matrix*

Description

Extract covariance matrix of the random effects for a model fitted with lme4.

Usage

```
extractVarCov(m)
```

Arguments

m	a fit from lme4 package (either linear or nonlinear)

extractVarCov.lme *Extract covariance matrix*

Description

Extract covariance matrix of the random effects for a model fitted with nlme.

Usage

```
## S3 method for class 'lme'
extractVarCov(m)
```

Arguments

m	a fit from nlme package (either linear or nonlinear)

extractVarCov.merMod *Extract covariance matrix*

Description

Extract covariance matrix of the random effects for a model fitted with lme4.

Usage

```
## S3 method for class 'merMod'
extractVarCov(m)
```

Arguments

m	a fit from lme4 package (either linear or nonlinear)

fim.vctest *Extract the Fisher Information Matrix*

Description

Extract the Fisher Information Matrix

Usage

```
fim.vctest(object)
```

Arguments

object	an object of class vctest

null.desc *null.desc*

Description

create null.value description

Usage

```
null.desc(msdata)
```

Arguments

msdata	a list containing the structure of the model and data, as an output from extractStruct.<package_name> functions

Details

Useful intern functions

objFunction *Internal functions for constrained minimization*

Description

Groups of functions used for the constrained minimization problem arising in the computation of the likelihood ratio test statistics.

Usage

```
objFunction(x, cst)

gradObjFunction(x, cst)

symMatrixFromVect(x)

ineqCstr(x, cst)

jacobianIneqCstr(x, cst)

eqCstr(x, cst)

jacobianEqCstr(x, cst)
```

Arguments

x	A vector
cst	A list of constants to be passed to the optimisation function

Value

value of the objective function, its gradient, and the set of inequality and equality constraints

Functions

- `objFunction()`: objective function to be optimized
- `gradObjFunction()`: gradient of the objective function
- `symMatrixFromVect()`: function creating a symmetric matrix from its unique elements stored in a vector
- `ineqCstr()`: set of inequality constraints
- `jacobianIneqCstr()`: jacobian of the inequality constraints
- `eqCstr()`: set of equality constraints
- `jacobianEqCstr()`: jacobian of the inequality constraints

pckName *Extract package name from a fitted mixed-effects model*

Description

Extract package name from a fitted mixed-effects model

Usage

```
pckName(m)
```

Arguments

m	a model with random effects fitted with nlme, lme4 or saemix

Value

a string giving the name of the package

print.desc.message *print.desc.message*

Description

print a message to indicate the null and alternative hypotheses

Usage

```
## S3 method for class 'desc.message'
print(msdata)
```

Arguments

msdata	a list containing the structure of the model and data, as an output from extractStruct.<package_name> functions

print.res.message *print.res.message*

Description

print a message with the results

Usage

```
## S3 method for class 'res.message'
print(results)
```

Arguments

results	an object of class vctest

print.vctest *Print*

Description

Print

Usage

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

Arguments

x	an object of class vctest
...	additional arguments

summary.vctest *Summary*

Description

Summary

Usage

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

Arguments

object	an object of class vctest
...	additional arguments

varCompTest *Variance component testing*

Description

Perform a likelihood ratio test to test whether a subset of the variances of the random effects are equal to zero. The test is defined by two hypotheses, H0 and H1, and the model under H0 is assumed to be nested within the model under H1. These functions can be used on objects of class lme-, nlme-, mer-, lmerMod, glmerMod, nlmerMord or SaemixObject.

It is possible to tests if any subset of the variances are equal to zero. However, the function does not currently support nested random effects, and assumes that the random effects are Gaussian.

Usage

```
varCompTest(
  m1,
  m0,
  control = list(M = 5000, parallel = T, nb_cores = 1, B = 1000),
  pval.comp = "bounds",
  fim = "extract",
  output = TRUE
)

## S3 method for class 'lme'
varCompTest(
  m1,
  m0,
  control = list(M = 5000, parallel = FALSE, nb_cores = 1, B = 1000),
```

```
  pval.comp = "bounds",
  fim = "extract",
  output = TRUE
)

## S3 method for class 'merMod'
varCompTest(
  m1,
  m0,
  control = list(M = 5000, parallel = FALSE, nb_cores = 1, B = 1000),
  pval.comp = "bounds",
  fim = "extract",
  output = TRUE
)

## S3 method for class 'SaemixObject'
varCompTest(
  m1,
  m0,
  control = list(M = 5000, parallel = FALSE, nb_cores = 1, B = 1000),
  pval.comp = "bounds",
  fim = "extract",
  output = TRUE
)
```

Arguments

m1	a fit of the model under H1, obtained from nlme, lme4 or saemix
m0	a fit of the model under H0, obtained from the same package as m0
control	(optional) a list of control options for the computation of the chi-bar-weights (see Details section)
pval.comp	(optional) the method to be used to compute the p-value, one of: "bounds" (the default), "approx" or "both" (see Details section)
fim	(optional) the method to compute the Fisher Information Matrix. Options are: fim="extract" to extract the FIM computed by the package which was used to fit the models, fim="compute" to evaluate the FIM using parametric bootstrap, and fim=I with I a positive semidefinite matrix, for a FIM provided by the user.
output	a boolean specifying if any output should be printed in the console (default to TRUE)

Details

The asymptotic distribution of the likelihood ratio test is a chi-bar-square, with weights that need to be approximated by Monte Carlo methods, apart from some specific cases where they are available explicitly. Therefore, the p-value of the test is not exact but approximated. This computation can be time-consuming, so the default behaviour of the function is to provide bounds on the exact p-value, which can be enough in practice to decide whether to reject or not the null hypothesis. This is

triggered by the option `pval.comp="bounds"`. To compute an approximation of the exact p-value, one should use the option `pval.comp="approx"` or `pval.comp="both"`.

When `pval.comp="approx"` or `pval.comp="both"`, the weights of the chi-bar-square distribution are computed using Monte Carlo, which might involve a larger computing time.

The `control` argument controls the options for chi-bar-square weights computation. It is a list with the following elements: `M` the size of the Monte Carlo simulation, i.e. the number of samples generated, `parallel` a boolean to specify if parallel computing should be used, and `nbcores` the number of cores to be used in case of parallel computing. Default is `M=5000`, `parallel=FALSE` and `nb_cores=1`. If `parallel=TRUE` but the value of `nb_cores` is not given, then it is set to the number of detected cores minus 1

Value

An object of class `htest` with the following components:

- `statistic` the likelihood ratio test statistics
- `null.value`
- `alternative`
- `parameters` the parameters of the limiting chi-bar-square distribution: the degrees of freedom and the weights of the chi-bar-square components and the Fisher Information Matrix
- `method` a character string indicating the name of the test
- `pvalue` a named vector containing the different p-values computed by the function: using the (estimated) weights, using the random sample from the chi-bar-square distribution, and the two bounds on the p-value.

Author(s)

Charlotte Baey <<charlotte.baey@univ-lille.fr>>

References

Baey C, Cournède P-H, Kuhn E, 2019. Asymptotic distribution of likelihood ratio test statistics for variance components in nonlinear mixed effects models. *Computational Statistics and Data Analysis* 135:107-122.

Silvapulle MJ, Sen PK, 2011. Constrained statistical inference: order, inequality and shape constraints.

Examples

```
# load lme4 package and example dataset
library(lme4)
data(Orthodont, package = "nlme")

# fit the two models under H1 and H0
m1 <- lmer(distance ~ 1 + Sex + age + age*Sex +
(0 + age | Subject), data = Orthodont, REML = FALSE)
m0 <- lm(distance ~ 1 + Sex + age + age*Sex, data = Orthodont)
```

```
# compare them (order is important: m1 comes first)
varCompTest(m1,m0,pval.comp="bounds")

# using nlme
library(nlme)
m1 <- lme(distance ~ 1 + Sex + age + age*Sex,
random = pdSymm(Subject ~ 1 + age), data = Orthodont, method = "ML")
m0 <- lme(distance ~ 1 + Sex, random = ~ 1 | Subject, data = Orthodont, method = "ML")

varCompTest(m1,m0)
```

`weightsChiBarSquare` *Monte Carlo approximation of chi-bar-square weights*

Description

The function provides a method to approximate the weights of the mixture components, when the number of components is known as well as the degrees of freedom of each chi-square distribution in the mixture, and given a vector of simulated values from the target $\bar{\chi}^2(V, C)$ distribution. Note that the estimation is based on (pseudo)-random Monte Carlo samples. For reproducible results, one should fix the seed of the (pseudo)-random number generator.

Usage

```
weightsChiBarSquare(df, V, dimsCone, orthan, control)
```

Arguments

`df`	a vector with the degrees of freedom of the chi-square components of the chi-bar-square distribution
`V`	a positive semi-definite matrix
`dimsCone`	a list with the dimensions of the cone C, expressed on the parameter space scale
`orthan`	a boolean specifying whether the cone is an orthan
`control`	(optional) a list of control options for the computation of the chi-bar-weights, containing two elements: `parallel` a boolean indicating whether computation should be done in parallel (FALSE by default), `nb_cores` the number of cores for parallel computing (if `parallel=TRUE` but no value is given for `nb_cores`, it is set to number of detected cores minus 1), and `M` the Monte Carlo sample size for the computation of the weights.

Value

A list containing the estimated weights, the standard deviations of the estimated weights and the random sample of `M` realizations from the chi-bar-square distribution

Index

alt.desc, 2
approxWeights, 3

bootinvFIM, 4
bootinvFIM.lme, 4
bootinvFIM.merMod, 5
bootinvFIM.SaemixObject, 5
bootstrap *(*bootinvFIM*)*, 4

dfChiBarSquare, 6

eqCstr *(*objFunction*)*, 11
extractFIM.lme, 6
extractStruct, 7
extractStruct.lme, 7
extractStruct.merMod, 8
extractStruct.SaemixObject, 8
extractVarCov, 9
extractVarCov.lme, 9
extractVarCov.merMod, 9

fim.vctest, 10

gradObjFunction *(*objFunction*)*, 11

ineqCstr *(*objFunction*)*, 11
invFIM *(*bootinvFIM*)*, 4

jacobianEqCstr *(*objFunction*)*, 11
jacobianIneqCstr *(*objFunction*)*, 11

null.desc, 10

objFunction, 11

pckName, 12
print.desc.message, 12
print.res.message, 13
print.vctest, 13

summary.vctest, 14
symMatrixFromVect *(*objFunction*)*, 11

varCompTest, 14

weightsChiBarSquare, 17