Package ‘varComp’

February 20, 2015

Type Package
Title Variance Component Models
Version 0.1-360
Date 2015-02-06
Author Long Qu (long.qu@wright.edu)
Maintainer Long Qu <long.qu@wright.edu>
Description Variance component models: REML estimation, testing fixed effect contrasts through Satterthwaite or Kenward-Roger methods, testing the nullity of variance components through (linear or quadratic) score tests or likelihood ratio tests.
Depends R (>= 3.0.0)
Imports quadprog, Matrix, MASS, CompQuadForm, RLRsim, SPA3G, mvtnorm, nlme, stats
LazyData no
ByteCompile yes
License GPL-3
NeedsCompilation no
Repository CRAN
Date/Publication 2015-02-06 18:45:44

R topics documented:

varComp-package .................................................. 2
anova.varComp .................................................. 3
Chi-bar-square ................................................... 6
cholRoot .......................................................... 7
coef.varComp ..................................................... 8
defunct Functions ............................................... 9
get.seed .......................................................... 10
Internal formula manipulation ................................. 10
internals .......................................................... 11
is.formula ......................................................... 13
Variance Component Models

Description

This package includes functions for fitting and testing variance component models, i.e., linear mixed-effect models with linear variance-covariance structures.

Details

- varComp: fitting variance component models.
- varComp.test: testing variance components.
- fixef.varComp: testing fixed-effect parameters by the Satterthwaite or Kenward-Roger method.

Author(s)

Long Qu

Maintainer: Long Qu <long.qu@wright.edu>
anova.varComp

References

See Also
nlme::lme

Examples

```r
### Oxide/Semiconductor example
library(nlme)
data(Oxide)
lmef = lme(Thickness~Source, Oxide, ~1|Lot/Wafer)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
VarCov(lmef)
coef(vcf, 'varComp') ## same values as above

vcf0 = varComp(Thickness~1, Oxide, ~Lot/Wafer)
fixef(vcf)
anova(vcf, vcf0)
anova(vcf)
anova(vcf, L=diag(1,2)) ## notice the difference in the last row

### Genetics example
trt=g1(2, 15)
set.seed(2340)
data.frame(trt=trt)
dat$SNP=matrix(sample(0:2, 120, replace=TRUE), 30)
dat$Y = as.numeric(trt)+rnorm(30) + dat$SNP%*%rnorm(4)
(vcfo0 = varComp(Y-trt, dat, ~ibs(SNP)))
(vcfo0 = varComp(Y-trt, dat, varcov = list('ibs(SNP)'~ibs(dat$SNP)))) ## same as above
(vcfo1 = varComp(Y-trt, dat, ~ibs(SNP):trt)) ## two variance components
summary(vcfo0)

varComp.test(vcfo0, vcfo1)
varComp.test(vcfo1, null=1)
varComp.test(Y-trt, dat, random1=~ibs(SNP), random2=~trt:ibs(SNP))
```

---

anova.varComp  
ANOVA-type analysis of fixed effect parameters

Description

fixef.varComp and anova.varComp test for fixed effect contrasts, as well as providing standard errors, etc.
Usage

```r
## S3 method for class 'varComp'
anova(object, ..., test = c("KR", "Satterthwaite"), L)
## S3 method for class 'varComp'
fixef(object, Lmat, alpha=.05, test=c("KR", "Satterthwaite"), ...)
## S3 method for class 'varCompFixEf'
print(x, ...)
```

Arguments

- `object`: An object of class `varComp`.
- `Lmat`: A matrix specifying the linear combinations of fixed effect parameters to be tested for nullity. Each row is a linear combination. If the column names are non-null, the columns will be re-ordered according to the column names of `model.matrix(object)`. In case of a single row matrix, it may also be given as a vector.
- `alpha`: The requested level of test.
- `test`: A character scalar requesting the type of the tests to be performed.
- `...`: For `anova.varComp`, this may be optional additional `varComp` objects. For `fixef.varComp` and `print.varCompFixEf`, this is not used.

Value

For `anova.varComp`, this is a data frame with the following columns:

- **F value**: The scaled $F$-statistic;
- **Scale**: The multiplicative factor in the scaled $F$-statistic;
- **numDF**: The numerator degrees of freedom;
- **denDF**: The denominator degrees of freedom;
- **Pr(>|F|)**: The $p$-value.

When `...` is missing but `L` is given, each row of `L` matrix will have a corresponding row in the result, testing the nullity of linear contrasts specified by that row of `L`. Additionally, an "Overall" row will be the last row, testing the nullity of linear contrasts of all rows of `L`.

When `...` is not missing, `L` will be ignored and the result will have the same number of rows as the number of `varComp` objects given (including the first `object` argument). The order of rows is such that the rank of the fixed effect design matrix is non-decreasing, i.e., smaller models first. The results will be the comparison between the current model vs. the model in the preceding row, except for the first row. For the first row, the model being compared is a null model. If the fixed effect design matrix for the model in first row includes the intercept in its column space, the null model includes only an intercept. If the intercept is not in the column space of the first model, the null model will contain no fixed effects.

When both `...` and `L` are missing, `anova.varComp` will test each fixed effect parameters together with a last "Overall" row. However, this last row will remove the intercept from the model, if there is one. See example below.
For `fixef.varComp`, the result is an object with class `varCompFixEf`. This is a named numeric vector, providing estimates of fixed effect contrasts specified by `Lmat`. The name will be the row names of `Lmat`. It has the `anova` attribute, a numeric matrix with the following columns:

- **Std. Error**: the standard error of estimate. If `test='KR'`, this is the adjusted standard error. If `test='Satterthwaite'`, this is the plug-in estimate.
- **Lower**: the lower limit of the $1-\alpha$ confidence interval.
- **Upper**: the upper limit of the $1-\alpha$ confidence interval.
- **t value**: the scaled $t$-statistic testing nullity of this contrast.
- **Scale**: a multiplicative factor in the scaled $t$-statistic.
- **Df**: the degrees of freedom for the $t$-statistic.
- **Pr(>|t|)**: the $p$-value.

Additionally, the `anova` attribute also has the `Overall` attribute, which is a single-row matrix with the same columns as the results of `anova.varComp`, testing for the overall nullity of linear contrasts specified by `Lmat`.

`print.varCompFixEf` prints the result from `fixef.varComp` and return the argument invisibly.

**Warning**

1. When neither `N` nor `L` is given for `anova.varComp`, the current implementation will test each fixed effect parameter separately. This behavior might be changed in future releases.

2. When `L` is not explicitly given, the current implementation will rely on `getOption("digits")` when trying to find the `L` matrix that test the difference between adjacent models.

**Author(s)**

Long Qu

**References**


**See Also**

`satterth.varComp, KR.varComp`
### Chi-bar-square

**Chi-bar-square distribution with nonnegativity cone constraints**

**Description**

`pchibarsq` is the distribution function of chi-bar-square distribution with nonnegativity cone constraint.

`wchibarsq` computes the mixing proportions for the chi-bar-square distribution.

`mchibarsq` computes the moments of the chi-bar-square distribution.

**Usage**

```r
pchibarsq(q, V, lower.tail = TRUE, log.p = FALSE)
mchibarsq(V, order = 1:2)
wchibarsq(V)
```

**Arguments**

- `q`  
  A vector of quantiles, as in `stats::pchisq`.  
- `V`  
  A positive-definite matrix, defining the distance measure used when projecting onto the cone.  
- `lower.tail`  
  logical, the same as in `stats::pchisq`.  
- `log.p`  
  logical, the same as in `stats::pchisq`.  
- `order`  
  A positive integer vector of the order of moments to be computed.

**Value**

`pchibarsq` gives the distribution function, `wchibarsq` gives the mixing proportion, and `mchibarsq` gives the moments.

**Author(s)**

Long Qu
References


See Also

`stats::pchisq`

Examples

```r
set.seed(203490L)
V =tcrossprod(matrix(rnorm(25), 5))
V =solve(V)
L =t(chol(V))
chibarsq =replicate(1e3L, -2*quadprog::solve.QP(V, V*%*%(L*%*%rnorm(5)),
  diag(1, 5), rep(0, 5))[['value']] )
chibarsq =sort(chibarsq)
p =pchibarsq(chibarsq, V)

## Not run:
plot(ecdf(chibarsq))
lines(chibarsq, p, col=4, lwd=3, lty=3)

## End(Not run)
mean(chibarsq); mean(chibarsq^2)
mchibarsq(V)
```

---

**cholRoot**

*Lower Cholesky Root*

**Description**

This function is similar to `base::chol`, but it returns the lower root and possibly remove unnecessary columns.

**Usage**

`cholRoot(x)`

**Arguments**

`x` A positive semidefinite matrix.

**Details**

This function calls `base::chol` with pivoting and returns the lower root. In the case of less than full rank `x`, unnecessary columns are removed. `tcrossprod(cholRoot(x))` will reproduce `x` (up to numerical errors).
Value

A numeric matrix, not necessarily a square matrix.

Author(s)

Long Qu

See Also

Matrix::chol, base::chol.

Examples

set.seed(3456L)
A = tcrossprod(matrix(rnorm(10L), 5L))
cholRoot(A)
range(A - tcrossprod(cholRoot(A)))


c coef.varComp  Obtaining parameter estimates from a varComp object

Description

c coef computes and/or extracts fixed-effect parameters, variance-component parameters, or ratios of variance component parameters to the error variance.

Usage

## S3 method for class 'varComp'
coef(object, what = c("fixed", "beta", "random", "varComp", "var.ratio", "tau"), ...)

Arguments

object

An object of class varComp

what

Character vector (only the first component will be used) specifying what parameters are requested. See details.

...  

Not used.

Details

"fixed" (default) and "beta" are equivalent, requesting fixed-effect parameters to be returned.
"random" and "varComp" are equivalent, requesting variance components (including the error variance) to be returned.
"var.ratio" and "tau" are equivalent, requesting the ratio of variance components to the error variance to be returned.
defunct Functions

Value
A named numeric vector of requested parameter estimates.

Author(s)
Long Qu

See Also
varComp

Examples
library(nlme)
data(Oxide)
cvf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
coef(cvf, 'varComp')
coef(cvf)  ## same as coef(cvf, 'fixed')
coef(cvf, 'var.ratio')

defunct Functions  Defunct functions

Description
Functions to be removed from future release.

Usage
varComp.LinScore.LC12(null.fit, X, Y, K, null, w, ...)
varComp.LinScore.LC12Boundary (null.fit, X, Y, K, null, ...)
varComp.LinScore.approximation(approximation, ...)

Arguments
null.fit,X,Y,K,null,w,approximation,....
To be described later, if indeed needed.
**get.seed**

*Recording pseudo-random number seeds*

**Description**

get.seed obtains current pseudo-random number seeds.

**Usage**

get.seed()

**Details**

This function obtains the `.Random.seed` object in the global environment. If it is absent, `runif(1L)` is called and then the seed is obtained.

**Value**

A numeric vector of `.Random.seed`, with `RNGkind` attribute being the result from calling `RNGkind()`.

**Author(s)**

Long Qu

**See Also**

base::.Random.seed

**Examples**

```r
set.seed(2034L)
all.equal(get.seed(), .Random.seed, check.attributes = FALSE)
```

---

**Internal formula manipulation**

*Internal formula term manipulation functions*

**Description**

These are helper function used in the formula interface of `varComp` and `varComp.test`. They either sort or split the formula terms involving ":`.

**Usage**

```r
callList2terms(cl)
splitTerm(term)
sortTerm(term, priority)
```
Arguments

- cl: A list converted from a call.
- term: A character scalar of a formula term to be re-ordered alphanumerically with respect to the components separated by ":".
- priority: A character vector whose elements have higher priority to be ordered to the front of the term.

Details

The main use of `sortTerm` is to transform an interaction term like "B:A" into ordered form "A:B" such that redundant terms are easier to be identified. The results will depend on the system locale.

Value

- `callList2terms` returns a character vector of individual terms (possibly with duplicates).
- `splitTerm` returns a character vector of unique individual components in the term.
- `sortTerm` returns a character scalar with individual components sorted.

Author(s)

Long Qu

See Also

- `stats::formula`

Examples

- `sortTerm("sex:ibs(SNP)")`  ## ibs(SNP) is moved to the front in most locales
- `sortTerm("sex:ibs(SNP)", "sex")`  ## the same as input
- `splitTerm("sex:ibs(SNP)")`  ## two components
- `sortTerm("ns(x, df=3):a:b:ibs(SNP)", "b")`

Description

These internal functions are mainly used by `varComp.fit` and `varComp.test.Common`. These functions make reference to and make changes to non-local objects. Thus, they are not supposed to be called directly, unless their environments are set properly.
Usage

AEI()
AOI()
EI()
OI()
WAI()
PREML()
preprocPREML(tau)
obj(tau)
obj2(ltau)
score()
updateLI(tau)
updateL1kLI()
updateL1y()
updateNegGrad()
updateNegHess()
updateNums()
updateNums2()
updateNumsPart()
updateTr1()
updateTr2()
V(tau)
updateDenom()
hess(tau)
gradients(tau)

varComp.test.nulDoTest(null.fit, additional.varcov, test = "LinScore",
control = varCompTest.control(test), alt.fit = NULL, ...)
varComp.test.2modelDoTest(null.fit, alt.fit, test = "LinScore",
control = varCompTest.control(test), ...)
varComp.test.altDoTest(alt.fit, null = integer(0L), test = "LinScore",
control = varCompTest.control(test), null.fit = NULL, ...)
varComp.test.Common()

varComp.LinScore.Normal(all.scores, lin.form, infoMat, null, w, tr1, n, LIkLI,
tau.idx, ...)
varComp.LinScore.Satterthwaite(all.scores, infoMat, null, w, tr1, n, ...)
varComp.LinScore.SSAS155(all.scores, lin.form, infoMat, null, w, tr1, n,
LIkLI, tau.idx, non.pd, control, ...)
varComp.LinScore.test(control, n, tau.idx, LIkLI, tr1, infoMat, all.scores, non.pd)
varComp.RLRT.test(control, negHess, alt.fit, null.fit, tau.idx)
varComp.VM03.test(control, infoMat, tau.idx, LIkLI, tr1, n, LIy, all.scores)
varComp.SS95.test(control, infoMat, tau.idx, LIkLI, LIy, tr1, n, all.scores)
is.formula

Arguments

tau, ltau, null.fit, additional.varcov, test, control, alt.fit, ..., null, all.scores, lin.form, infoMat, w, t, Q,
To be described later, if indeed needed.

Description

This function checks if the object inherits the 'formula' class.

Usage

is.formula(x)

Arguments

x An R object.

Value

A logical scalar.

Author(s)

Long Qu

See Also

stats::formula

Examples

is.formula(~a+b:a:b)
is.formula('~a+b:a:b')
Kernel functions useful for genetic associations

Description
These are the functions that might be used in computing pairwise inter-individual similarities based on their single nucleotide polymorphism (SNP) genotypes.

Usage

am(x)
AM(x)
ibs(x)
IBS(x)
lin0(x)
Lin0(x)
quad1(x)
Quad1(x)
Minkowski(x, p = 1)
minkowski(x, p = 1)
polyk(x, c=0, d=1)
Polyk(x, c=0, d=1)

Arguments

x A numeric matrix encoding genotypes. Each row corresponds to an individual and each column corresponds to a genetic marker. Usually, allele-counting coding is used, but others are allowed.

p The exponent defining the Minkowski distance. The same as in stats::dist.

c The constant added to cross-products before raising to the power of d.

d The exponent defining the polynomial kernel. When c=0 and d=1, this is equivalent to lin0. When c=1 and d=2, this is equivalent to quad1.

Details
These functions compute the pairwise similarities among rows of x. Lower-case versions are more useful in the formula interface to specify random genetic effects. Upper-case versions can be used to directly compute the genetic similarity matrix.

am and AM calculate the allele-matching kernel, and AM is based on SPA3G::KERNEL.
ibs and IBS compute the identity-by-descent (IBS) kernel. IBS is computed as 1 - as.matrix(dist(x, method='manhattan')) * .5 /max(1, ncol(x)) .
lin0 and Lin0 compute the linear kernel with zero intercept. Lin0 is computed as normalizeTrace(tcrossprod(x)/max(1,ncol(x)))).
quad1 and Quad1 compute the quadratic kernel with offset 1. Quad1 is computed as normalizeTrace((base:tcrossprod(x)+1)^2).
minkowski and Minkowski compute the similarity based on the Minkowski distance. Minkowski is computed as $1 - \text{as.matrix(dist(x, method='minkowski', p=p))} * .5 / \max(1, \text{ncol}(x))^{(1/p)}$.

**Value**

The functions starting with an upper-case letter returns an n-by-n symmetric similarity matrix, where n equals nrow(x). The corresponding functions starting with a lower-case letter returns a matrix L such that tcrossprod(L) equals the value from their upper-case counterparts. The number of rows is n, but the number of columns is the rank of the similarity matrix.

**Author(s)**

Long Qu

**See Also**

cholRoot, normalizeTrace, stats::dist, SPA3G::KERNEL, varComp

**Examples**

```r
c> set.seed(2345432L)
c> x=matrix(sample(2, 50L, replace=TRUE), 10L)
c> IBS(x)
c> range(tcrossprod(ibs(x)) - IBS(x) )
c> AM(x)
c> range(tcrossprod(am(x)) - AM(x) )
c> Lin0(x)
c> range(tcrossprod(lin0(x)) - Lin0(x) )
c> range(Lin0(x) - Polyk(x, 0, 1))

c> Quad1(x)
c> range(tcrossprod(quad1(x)) - Quad1(x) )
c> range(Quad1(x) - Polyk(x, 1, 2))

c> Minkowski(x)
c> range(tcrossprod(minkowski(x)) - Minkowski(x) )
c> range(tcrossprod(minkowski(x)) - IBS(x) )

c> # Use in formulas
ncol(model.matrix(~0+ibs(x))
c> range(tcrossprod(model.matrix(~0+ibs(x))) - IBS(x))
```
### logLik.varComp

**Extracting Profiled Restricted Log Likelihood**

**Description**

Extracting maximized profiled restricted log likelihood (PREML) from a varComp object

**Usage**

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

**Arguments**

- `object` A varComp object
- `...` Not used

**Details**

The likelihood value is the profiled restricted log likelihood. The actual value depends on the residual contrast being chosen, but the location of maximum does not. Thus it is only comparable when the same residual contrast is used. This value is not to be compared to results reported using functions in other packages, e.g., `nlme::lme` or SAS. The degree of freedom reported is the number of variance components not on the boundary of parameter space, whereas the decision is based on `object$control$boundary.eps`.

**Value**

A numeric value of class `logLik`.

**Author(s)**

Long Qu

**References**


**See Also**

`varComp`. 

---
Examples

```r
library(nlme)
data(Oxide)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
logLik(vcf)
```

Description

This function computes the minimum norm quadratic unbiased estimate of variance components. Typically this is used as starting values of REML.

Usage

```r
minque(y, varcov, start = rep(0, length(k)), lower.bound = -Inf, restricted = TRUE)
```

Arguments

- **y**: A numeric vector of zero-mean response variable.
- **varcov**: A list of variance-covariance matrices, the same as in `varComp` with no random argument.
- **start**: A numeric vector of prior values of the ratio of variance components to the error variance.
- **lower.bound**: A numeric value of the ratio of variance components to the error variance. If this is -Inf (default), minque solves the linear equation as in Rao (1972). If this is zero, quadratic programming is used to minimize the squared error between the two sides of the linear equation.
- **restricted**: Not used. Currently only restricted likelihood is supported.

Value

A numeric vector of estimates of ratio of variance components to error variance.

Author(s)

Long Qu

References


See Also

`varComp.fit`
model.matrix.varComp

Examples

library(nlme)
data(Oxide)
vcf0 = varComp(Thickness~Source, Oxide, ~Lot/Wafer, control=varComp.control(nlminb=nlminb.control(iter.max=0L)))
coef(vcf0, 'var.ratio')
(st=minque(vcf0$residual.contrast, vcf0$working.cor, lower.bound = 0))
(vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer, control=varComp.control(start=st)))
coef(vcf, 'var.ratio')

model.matrix.varComp Extracting model matrices

Description

This function extracts the fixed-effect design matrix, random-effect design matrix, or the list of variance-covariance matrices whose weighted sum being the variance-covariance matrix of the response variable.

Usage

## S3 method for class 'varComp'
model.matrix(object, what = c("fixed", "random", "varcov", "X", "K", "Z"), ...)

Arguments

object A varComp object
what A character object (only the first element will be used) specifying what kind of design matrix is requested. See details.
... Not used.

Details

"fixed" and "X" are equivalent, requesting the fixed effect design matrix.
"random" and "Z" are equivalent, requesting the random effect design matrix. Note that this is an equivalent version of the design matrices such that the tcrossprod will be the contribution to the marginal variance-covariance. This is not necessarily the one computed directly from the random argument passed to varComp. These are actually computed from cholRoot of K matrices.
"varcov" and "K" are equivalent, requesting the contribution of each random effect to the marginal correlation matrix. These are not necessarily the same value passed to the varcov argument of varComp, because the input value will be treated as the "G" matrices when random is not missing, but the result here will always be "K" matrices. See varComp for notations.
normalizeTrace

Value
If what="fixed" or "X", a single numeric matrix of fixed-effect design matrix. Otherwise, a list of requested matrices.

Note
See details on possible confusions.

Author(s)
Long Qu

See Also
varcomp

Examples
library(nlme)
data(Oxide)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
model.matrix(vcf, 'fixed')
model.matrix(vcf, 'random')
model.matrix(vcf, 'varcov')

normalizeTrace

Rescales a square matrix such that the trace is the same as the dimension.

Description
Rescales a square matrix such that the trace is the same as the dimension.

Usage
normalizeTrace(x)

Arguments
x A square numeric matrix to be rescaled.

Details
This is computed as \{x=as.matrix(x); x/mean(diag(x))\}.

Value
A numeric matrix.
Author(s)

Long Qu

See Also

base::diag

Examples

set.seed(3456L)
A=tcrossprod(runif(10L))
A0 = normalizeTrace(A)
sum(diag(A0)) - nrow(A)

Description

Functions to be implemented in future releases.

Usage

varComp::LinScore::beta(...)
varComp::LinScore::cumulant3(...)
varComp::RWD::test(control,...)
varComp::Wald::test(control,...)
varComp::LinScore::pboot(...)
varComp::LinScore::saddlepoint(...)

Arguments

control list of control parameters.

... Place holder.
p.value.varComp.test  Extracting p-values

Description

p.value is a generic function for extracting p-values of tests.

Usage

p.value(x)
## Default S3 method:
p.value(x)
## S3 method for class 'htest'
p.value(x)
## S3 method for class 'varComp.test'
p.value(x)
## S3 method for class 'varCompFixEf'
p.value(x)

Arguments

x  An R object from which to extract p-values.

Details

By default, p.value extract the $p.value component from an htest object.

The p.value.varComp.test function extracts p-values from all tests in the object. The p.value.varCompFixEf function extracts individual p-values of tests of fixed effect with the overall p-value as the Overall attribute.

Value

A numeric vector of p-value(s)

Author(s)

Long Qu

See Also

varComp.test
Examples

```r
library(nlme)
data(Oxide)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
coef(vcf, 'varComp')
p.value(varComp.test(vcf))  ## test both Lot and Wafer in Lot
p.value(varComp.test(vcf, null=1))  ## test Wafer in Lot
```

print.varComp  

Summary and printing of varComp objects

Description

Summary and printing functions of varComp objects.

Usage

```r
## S3 method for class 'varComp'
print(x, ...)
## S3 method for class 'varComp'
summary(object, ...)
```

Arguments

- `x, object`: A varComp object
- `...`: Not used in print.varComp. For summary.varComp, the ... are passed to fixef.varComp.

Details

print.varComp will print the fixed effect estimates, variance parameter estimates, and the number of observations.

summary.varComp will expand the fixef component by calling fixef.varComp.

Value

invisible(x) for print.varComp.

An object of class c("summary.varComp", "varComp") for summary.varComp.

Author(s)

Long Qu
print.varComp.test  

Print variance component test results

Description
Currently only the p-values are printed.

Usage
```r
## S3 method for class 'varComp.test'
print(x, ...)
```

Arguments
- `x`: An object of class `varComp.test`
- `...`: Not used.

Value
Invisible `p.value(x)`

Author(s)
Long Qu

safeseq  

A “safer” `seq`

Description
This is the same as `seq(from, to, by)` except that when the sign of `by` is wrong, `safeseq` returns a zero-length vector rather than throwing an error.

Usage
```
safeseq(from = 1L, to = 1L, by = 1L, ...)  
```

Arguments
- `from`: The same as the `from` in `base::seq`.
- `to`: The same as the `to` in `base::seq`.
- `by`: The same as the `by` in `base::seq`.
- `...`: The same as in `base::seq`.  


Value

A vector of the same class as by*(from-to), but possibly with length being zero.

Author(s)

Long Qu

See Also

base::seq

Examples

identical(integer(0L), safeseq(1L, 0L, 1L))

symbols

Some symbols that might be used in formulas

Description

Some symbols that might be used in formulas: "+", ":", "\*", "~".

Usage

plus
colon
star
tilde

Examples

identical(plus, as.symbol("+"))
identical(colon, as.symbol("::"))
identical(star, as.symbol("\*"))
identical(tilde, as.symbol("~"))
**Description**

`varComp` and `varComp.fit` fit linear mixed-effect models where the marginal variance-covariance matrix is linear in known positive semidefinite matrices. `varComp` uses the usual formula interface, whereas `varComp.fit` is the underlying working horse. `dofit.varComp` performs model fitting if the object has been previously created by setting `dofit = FALSE` when calling `varComp`.

**Usage**

```r
varComp(fixed, data, random, varcov, weights, subset,  
family = stats::gaussian('identity'), na.action, offset,  
control = varComp.control(...), doFit = TRUE,  
normalizeTrace = TRUE, contrasts = NULL,  
model = TRUE, X = TRUE, Y = TRUE, K = TRUE, ...)  

varComp.fit(Y, X = matrix(0, length(Y), 0L), K, control = varComp.control())

dofit.varComp(object)
```

**Arguments**

- **fixed**: A two-sided formula specifying the fixed effects of the model. This is the same as in `stats::lm` or `nlme::lme`.
- **data**: an optional data frame, list or environment containing the variables in the model. If not found in data, the variables are taken from `environment(fixed)`, typically the environment from which `varComp` is called. This is the same as the data argument of `stats::lm` or `nlme::lme`.
- **random**: An optional one-sided formula specifying additive random effects, of the form `~ z1 + z2 + z3`. Interactions are allowed. The error variance component should not be included. A warning will be given when `environment(fixed)` and `environment(random)` do not match. See details.
- **varcov**: An optional list of symmetric positive semidefinite matrices. If `random` is nonmissing, these matrices represent the correlation matrix of each random effect. Thus the number of the random effects in `random` must be equal to the length of `varcov`. If `varcov` is named, the names will be matched to those used in `random`, following the same rule of arguments matching in function calls. If `varcov` is unnamed, it is assumed that the order is the same as in `random`. If `random` is missing, the weighted sum of these matrices represent the contribution of random effects to the marginal variance of the response variable, with unknown weights representing variance components.
- **weights**: An optional nonnegative vector of the same length as the response variable specified in `fixed`. When it is given, it is inversely proportional to the error variances not captured by `random` and `varcov`. This is similar to the `weights` argument in `stats::lm`. 

subset An optional vector specifying a subset of observations to be used in the fitting process.

family The same as the family argument of stats::glm. However, only gaussian('identity') is supported currently.

na.action The same as in stats::lm.

offset The same as in stats::glm. These offsets are assumed as known fixed effects.

control An object from calling varComp.control.

doFit A logical scalar, indicating whether model fitting should be performed.

normalizeTrace A logical scalar, indicating whether the individual variance-covariance matrices should be normalized such that variance components are on the same scale.

contrasts The same as in stats::lm.

model A logical scalar, indicating whether the model frame will be included in the result.

X For varComp, this is a logical scalar, indicating whether the fixed-effect design matrix should be included in the result. For varComp.fit this is the optional numeric fixed effect design matrix for the model. If X is missing or a matrix with zero columns, it is assumed that Y has zero mean.

Y For varComp, this is a logical scalar, indicating whether the response variable should be included in the result. For varComp.fit, this is a numeric vector of response variables.

K For varComp, this is a logical scalar, indicating whether the list of variance-covariance matrices should be included in the result. These matrices are computed by pre-and-post multiplying the design matrices of random effects specified by random and the corresponding matrices specified by varcov. For varComp.fit, this is a list of variance-covariance matrices.

... When control is given, this is ignored. Otherwise, these arguments are passed to varComp.control and the result will be used as the control argument.

object An object of class varComp.

Details
The variance component model is of form

\[ \mathbf{Y} = \mathbf{X}\beta + \mathbf{e} \]

where \( \mathbf{e} \) is multivariate normally distributed with mean zero and variance-covariance matrix \( \mathbf{V} \) being

\[ \mathbf{V} = \sum_{j=1}^{R} \sigma_j^2 \mathbf{K}_j + \sigma_e^2 \mathbf{W} \]

in which \( \mathbf{K}_j \) are known positive semidefinite matrices and \( \mathbf{W} \) is a known diagonal positive definite matrix. In the case of random-effect modeling, the \( \mathbf{K} \) matrices are further given by

\[ \mathbf{K}_j = \mathbf{Z}_j \mathbf{G}_j \mathbf{Z}_j^\top \]
where $Z_j$ is the design matrix for the $j$th random-effect factor and $G_j$ the known variance-covariance matrix for the $j$th random-effect vector.

In the varComp formula interface, the X matrix and response variable are specified by the fixed argument. The optional random argument specifies the Z matrices. When random is missing, they are assumed to be identity matrices. The varcov argument specifies the G matrices. The weights argument specifies the W matrix.

Note that in random, kernel functions are allowed. For example, random $\sim$ ibs(SNP) is allowed to specify the similarities contributed by a matrix SNP through the identity-by-descent (IBS) kernel. Fixed-by-random interactions are allowed in random. If F is a fixed factor with 2 levels, and F has already appeared in the fixed argument, then random $\sim$ ibs(SNP) + F:ibs(SNP) will specify two random effects. The first contributes to the marginal variance-covariance matrix through IBS(SNP); the other contributes to the marginal variance-covariance matrix through tcrossprod(X_F2) * IBS(SNP), where X_F2 is the 2nd column of the fixed-effect design matrix of factor F under the sum-to-zero contrast. Note that sum-to-zero contrast will be used even if another contrast is used for F in the fixed argument. This behavior corresponds to what we usually mean by fixed-by-random interaction. If one indeed needs to avoid sum-to-zero contrast in this case, it is only possible to do so by providing varcov directly without specifying random.

Additionally, the fixed-by-random interactions in random specified by the “:” symbol are actually treated as if they are specified by “*”. In other words, random $\sim$ ibs(SNP) + F:ibs(SNP), random $\sim$ F:ibs(SNP), and random $\sim$ F*ibs(SNP) are actually treated as equivalent when F has already appeared on the right-side of fixed. Again this corresponds to the usual notion of fixed-by-random interaction. If such behavior is not wanted, one can directly give the intended varcov and leave random as missing.

Finally, intercept is not included in the random interface.

The model fitting process uses profiled restricted maximum likelihood (PREML), where the error variance is always profiled out of the REML likelihood. The non-negativity constraints of variance components are always imposed. See varComp.control for arguments controlling the modeling fitting.

Value

The value of any of the three functions is a list with class varComp. Depending on whether doFit is TRUE or FALSE, the components in the list might differ. In either case, the result from varComp always contains the following elements:

- na.action: the na.action used in the model frame.
- offset: the same as input.
- contrasts: the contrast used in the fixed-effect design matrix.
- xlevels: the levels of both fixed-effect and random-effect factors.
- terms: both fixed-effect and random-effect terms.
- call: the actual call.
- nobs: the number of observations.
- control: the same as input.
- random.labels: the labels used to differentiate random effects. Note that the error variance is not included here. It is safe to check the number of variance components specified by the model by checking the length of random.labels.
• doFit: The value of doFit is either TRUE or a call. If the doFit argument of varComp is set to FALSE, this component will be a call that will do the model fitting when being evaluated. If the model fitting has already been done, this component will always be TRUE.

In the case where doFit is TRUE, the following components will also be present in the result of varComp and doFit:

• parms: a named numeric vector of parameter estimates. These are the estimated ratio of each variance component relative to the error variance.
• gradients: a named numeric vector of gradient of the PREML function at the final parameter estimate. Because of non-negativity constraints, the gradients are not necessarily all zero at convergence.
• hessian: a numeric matrix of Hessian matrix at parameter estimate. This is always computed through the observed information, no matter how the information argument of varComp.control is set.
• sigma2: the estimated error variance.
• varComps: a named numeric vector of variance components. These are the same as parms times sigma2.
• n.iter: the number of iterations during fitting.
• PREML: the final maximized PREML function value.
• X.Q2: a matrix, when left-multiplied to the response variable, produces the residual contrasts.
• residual.contrast: a numeric vector of residual contrasts with length(Y) - qr(X)$rank elements.
• working.cor: the list of individual variance-covariance matrices, whose weighted sum is the marginal variance-covariance matrix of the residual contrast.

If varComp.fit is called directly, then doFit is always TRUE, but the result will not contain any of na.action, offset, contrasts, xzlevels, and terms, as they do not apply.

For the varComp interface, if any of model, X, Y or K is TRUE, the corresponding part will be included in the result. If weights is non-missing, it will be included in the result.

Note

Currently, available S3 methods for the varComp class are:

• coef: retrieving fixed-effect estimates, variance component estimates, or estimates of the ratios of variance components to the error variance.
• model.matrix: retrieving the fixed-effect design matrix, equivalent random-effect design matrix, or the list of individual variance-covariance matrices.
• vcov: computing variance-covariance matrix of fixed-effect parameters, variance parameters, or the response variable.
• fixef: retrieving fixed-effect parameter estimates with standard errors and default hypothesis tests against zero. See also satterth.varComp and KR.varComp for computing approximate denominator of degrees of freedom for the F-statistics.
• anova: similar to fixef, but returning a data frame.
varComp

- varComp.test: testing for nullity of random components (other than the error variance).
- logLik: retrieving the maximized PREML value with degrees of freedom.
- print: showing the object.
- summary: summarize the object by providing standard errors and default tests against zero.

Author(s)

Long Qu

References


See Also

varComp.test, coef.varComp, model.matrix.varComp, vcov.varComp, fixef.varComp, satterth.varComp, KR.varComp, logLik.varComp, print.varComp, summary.varComp, nlme::lme, stats::lm

Examples

```r
### Oxide/Semiconductor data example
library(nlme)
data(Oxide)
lmef = lme(Thickness~Source, Oxide, ~1|Lot/Wafer)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
VarCorr(lmef)
coef(vcf, 'varComp') ## same values as above
k0 = tcrossprod(model.matrix(~0+Lot, Oxide))
k1 = tcrossprod(Oxide$Source==1) * k0
k2 = tcrossprod(Oxide$Source==2) * k0
k3 = tcrossprod(model.matrix(~0+Lot:Wafer, Oxide))

## unequal variance across Source for Lot effects, in a preferred parameterization:
(vcf1 = varComp(Thickness~Source, Oxide, varcov=list(S1Lot=k1, S2Lot=k2, 'Lot:Wafer'=k3))

## unequal variance across Source for Lot effects, in a different parameterization:
(vcf2 = varComp(Thickness~Source, Oxide, varcov=list(Lot=k0, S2Lot=k2, 'Lot:Wafer'=k3))

## unequal variance across Source for Lot effects, but in a poor parameterization that
## turns out to be the same as vcf after fitting:
(vcf3 = varComp(Thickness~Source, Oxide, varcov=list(Lot=k0, S1Lot=k1, 'Lot:Wafer'=k3))
logLik(vcf)
logLik(vcf1)
logLik(vcf2) ## the same as vcf1
logLik(vcf3) ## the same as vcf

## fixef-effect only
vcf0 = varComp(Thickness~Source, Oxide)
summary(vcf0)
summary(lmflm(Thickness~Source, Oxide))

vcf00 = varComp(Thickness~0, Oxide)
summary(vcf00)
summary(lmflm(Thickness~0, Oxide))
```
### Genetics example

```r
trt = gl(2, 15)
set.seed(2340)
dat = data.frame(trt = trt)
dat$SNP = matrix(sample(0:2, 120, replace = TRUE), 30)
dat$Y = as.numeric(trt) + rnorm(30) + dat$SNP * rnorm(4)

(vcf0 = varComp(Y ~ trt, dat, ~ibs(SNP)))
(vcf00 = varComp(Y ~ trt, dat, varcov = list(~ibs(SNP) ~ IBS(dat$SNP))))

(vcf01 = varComp(Y ~ trt, dat, ~ibs(SNP):trt))

dat$trt[1] = NA
varComp(Y ~ trt, dat, ~ibs(SNP)) ## 29 observations compared to 30 in vcf0

varComp(Y ~ trt, dat, ~ibs(SNP)) ## still, 29 observations, as ibs handles sporadic NA

varComp(Y ~ trt, dat, ~ibs(SNP)) ## 28 observations, as no genotype for the 3rd obs
```

---

### varComp.control

*Functions controlling the model fitting of varComp*

#### Description

`varComp.control` provides detailed control of model fitting for `varComp`. `nlminb.control` provides detailed control of the optimization procedure. `informationTypes` is a character vector containing the types of information matrices.

#### Usage

```r
varComp.control(verbos = FALSE, start = NULL, REML = TRUE,
information = informationTypes, boundary. = 5e-04,
nlminb = nlminb.control(iter.max = 200L, eval.max = 500L),
plot.it = FALSE, keepXYK = TRUE)

nlminb.control(eval.max = 200L, iter.max = 150L, trace = 0L, abs.tol = 0,
rel.tol = 1e-10, x.tol = 1.5e-08, xf.tol = 2.2e-14, step.min = 1,
step.max = 1, sing.tol = rel.tol, ...)
```

#### Arguments

- `verbose` A logical scalar, indicating whether excessive messages are printed.
- `start` An optional numeric vector of starting values for the ratio of variance components to the error variance. If missing, `minque` will be called with zero as the prior value and with non-negativity constraints. The result of `minque` will be used as the starting value.
varComp.control

REML A logical scalar. Currently, only TRUE is supported.

information A character vector (only the first element will be used) specifying the information matrix to be used during optimization. Available choices are given in informationTypes. See details.

boundary.eps A small positive number. If the parameter estimates fall below this number, additional checking will be performed to examine if the parameter estimate hits the lower boundary (0) of the parameter space.

nlminb A list of arguments to be passed to stats::nlminb. This is most conveniently specified through nlminb.control.

plot.it A logical scalar, indicating whether the profiled REML function is plotted around the fitted maximum. This is only effective when the number of variance component is one (not counting the error variance).

keepXYK A logical scalar, indicating whether the fixed-effect design matrix (X), response variable (Y) and the list of variance-covariance matrices (K) are included in the result. See varComp for notations.

eval.max, iter.max, trace, abs.tol, rel.tol, x.tol, xf.tol, step.min, step.max, sing.tol, ...
The same as in stats::nlminb.

Details

The objective function is always the profiled restricted log likelihood function, where the error variance is profiled out from the REML function. The parameterization being used is the ratio of each variance component to the error variance. Non-negativity constraints are imposed on such ratios.

The information argument specifies the (approximate) Hessian matrix to be used during optimization. Available choices are "EI" (expected information), "OI" (observed information), "AEI" (approximate expected information), "AOI" (approximate observed information), and "WAI" (weighted average information). The approximate versions of "EI" and "OI" replace the more computationally demanding trace terms by their realized counter-parts that are unbiased for the trace, and the "WAI" is similar to the average-information of Gilmour (1995) that cancels such terms, except that the latter reference is based on unprofiled REML function rather than profiled REML. The choice of information only affects the speed and the stability of fitting. As long as the objective function is well behaved, the parameter estimates should not be affected by which information matrix is used during fitting.

Both varComp.control and nlminb.control are simply a wrapper of the input arguments.

Value

A named list of the input arguments.

Note

Setting iter.max=0L is a trick to get residual contrast and working correlation matrices computed, but not fitting the models completely.

Author(s)

Long Qu
References


See Also

*varComp*

Examples

```r
library(nlme)
data(Oxide)

## Using fisher scoring algorithm:
varComp(Thickness~Source, Oxide, ~Lot/Wafer,
      control=varComp.control(information='EI'))

## Using Newton-Raphson algorithm:
varComp(Thickness~Source, Oxide, ~Lot/Wafer,
      control=varComp.control(information='0I'))

## Computing MINQUE0, residual contrasts, and marginal correlations of residual contrasts
varComp(Thickness~Source, Oxide, ~Lot/Wafer,
      control=varComp.control(nlminb=nlminb.control(iter.max=0L)))
```

---

**varComp.test**

*Tests for Nullity of Variance Components*

Description

These are the user-interface functions to test the nullity of variance components, using linear score tests, projected quadratic score tests, or restricted likelihood ratio tests.

Usage

```r
varComp.test(object, ...)
## S3 method for class 'formula'
varComp.test(object, data, random1, varcov1,
             random2, varcov2, fit.control, test.control, ...)

## S3 method for class 'varComp'
varComp.test(object, object2, additional.varcov, null,
              test = "LinScore", control = varCompTest.control(test), ...)
```
Arguments

object, object2

An R object. In varComp.test.formula, this is the fixed-effect formula. In
varComp.test.varComp these are the varComp objects specifying the null model
and the alternative models. object2 could be missing, but then either additional.varcov
or null needs to be given.

data

The same as in varComp.

random1, random2

The random effect formulas for the first and the second models, respectively.
Specification is the same as in varComp.

varcov1, varcov2

The lists of variance-covariance matrices for the first and the second models,
respectively. Specification is the same as in varComp.

fit.control

test.control, control

A list of arguments of class varComp.control.

A list of arguments of class varCompTest.control.

additional.varcov

An optional list of variance-covariance matrices. It is only used when object2
is missing. When this is given, object will be treated as the null model. And
the test of interest is the nullity of variance components corresponding to these
additional.varcov. null needs to be missing when additional.varcov is
given.

null

An optional integer vector, indicating the variance components in the null model.
It is only used when object2 is missing, and is incompatible with additional.varcov.
When null is given, object is treated as the alternative model. And the test of
interest is the nullity of variance components not included in null. For ex-
ample, if object has two variance components (other than the error variance).
Setting null=1L will test the nullity of the second components, whereas set-
ing null=integer(0L) (the default) will test the simultaneous nullity of both
components.

test

A character vector, specifying the tests to be performed. This does not need to
be a vector of length 1. Available choices are given in varCompTests. Currently,
this includes "LinScore", "VM03", "SS95", "HP01", and "RLRT". However, not
all test methods are applicable to all models/hypotheses.

Details

The formula interface varComp.test.formula is primarily used to test two variance component
models with the same fixed-effect formula. The two models need to be nested.

In the varComp.test.varComp interface, exactly one of object2, additional.varcov and null
needs to be given. If object2 is given, two-model comparison is performed against object. If
additional.varcov is given, object is treated as the null and additional.varcov is treated as
the additional variance components to be tested. If null is given, object is treated as the alternative
and the components not in null will be tested.
Value

A list of class varComp.test of the same length as test. Each component is either an object of htest or a list of htest objects, depending upon whether multiple methods are requested under the same test. See varCompTest.control for more details on controlling the tests to be performed.

Author(s)

Long Qu

References


See Also

varCompTest.control, varComp

Examples

### Oxide/Semiconductor data example
library(nlme)
data(Oxide)
lmeF = lme(Thickness~Source, Oxide, ~1|Lot/Wafer)
vcF = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
VarCorr(lmeF)
coef(vcF, 'varComp') ## same values as above
varComp.test(vcF) ## test against linear model
varComp.test(vcF, null=1) ## test against model with only Lot random effect

### Genetics example
trt=g1(2, 15)
set.seed(2340)
dat=data.frame(trt=trt)
dat$SNP=matrix(sample(0:2, 120, replace=TRUE), 30)
varCompTest.control  
Control of variance component testing

Description

varCompTest.control provide a list of arguments that control the details of the testing method for testing the nullity of variance components in varComp.test.

Usage

```r
varCompTest.control(test = "LinScore", LinScore.wt = "InvSTD", LinScore.acc = 1e-08,
LinScore.lim = 1e+06L, LinScore.method = c("AS155", "SSAS155"),
VM03.method = c("SSChiBarSq", "pboot"), VM03.nsims = 10000L,
SS95.method = c("pboot", "pboot"), SS95.nsims = 10000L,
RLRT.method = c("exact", "pboot"), RLRT nsims = 10000L,
information = 'EI'
)
```

Arguments

- **test**: A character vector, specifying the tests to be performed. Available choices are given in varCompTests. Currently, the following tests are supported: 'LinScore' (linear score tests of Qu et al. [2013]), 'VM03' (projected quadratic score test of Verbeke and Molenberghs [2003]), 'SS95' (projected quadratic score test of Silvapulle and Silvapulle [1995] and Hall and Praestgaard [2001]), 'HP01' (equivalent to 'SS95'), 'RLRT' (restricted likelihood ratio test of Crainiceanu and Ruppert [2003] or its pseudo-likelihood heuristic variant of Greven et al [2008]).
- **LinScore.wt**: A character vector, specifying the weighting method for the linear score test. Available choices are given in LinScoreWeightingMethods. Currently, the following are supported: 'EqWt' (equal weight for every variance component), 'InvSTD' (inverse standard deviation weighting), 'InvSqrtV' (inverse square root of covariance matrix) and 'MinVar' (weights that minimize the variance of the resulting test statistic). See Qu et al (2013) for detailed explanations.
<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinScore.acc</td>
<td>A single small positive number, specifying the accuracy when computing p-values from the linear score test. This is the same as the \texttt{acc} argument of \texttt{CompQuadForm::davies}.</td>
</tr>
<tr>
<td>LinScore.lim</td>
<td>A large positive integer, specifying the maximum number of integration terms when computing the p-value from the linear score test. This is the same as the \texttt{lim} argument of \texttt{CompQuadForm::davies}.</td>
</tr>
<tr>
<td>LinScore.method</td>
<td>A character vector with two elements. The first component specifies the method for obtaining null distributions when the null hypothesis contains no variance components other than the error variance; the second component specifies the method for obtaining null distributions when the null hypothesis contains additional variance parameters other than the error variance. Available choices are 'AS155' (Applied Statistics algorithm 155 of Davies (1980), i.e., \texttt{CompQuadForm::davies}), 'exact' (synonym of 'AS155'), 'Davies' (another synonym of 'AS155'), 'SSAS155' (shifted and scaled version of 'AS155', applicable to the 2nd component only), 'Satterthwaite' (scaled Chi-square approximation, applicable to the 2nd component only), and 'Normal' (normal approximation, applicable to the 2nd component only).</td>
</tr>
<tr>
<td>VM03.method</td>
<td>A character vector with two components for the VM03 test, each component being a character string. The first component specifies the method for obtaining null distributions when the null hypothesis contains no variance components other than the error variance; the second component specifies the method for obtaining null distributions when the null hypothesis contains additional variance parameters other than the error variance. Currently, the 2nd component is discarded. Available choices are 'ChiBarSq' (asymptotic null distribution of Chi-bar-squares; c.f., \texttt{pchibarsq}) and 'pboot' (parametric bootstrap, i.e., the Monte Carlo approximation to the finite-sample null distribution; c.f., Qu et al [2013]).</td>
</tr>
<tr>
<td>VM03.nsim</td>
<td>A large positive integer, specifying the number of simulations used to approximate the null, when VM03.method[1] is set to 'pboot'.</td>
</tr>
<tr>
<td>SS95.method</td>
<td>A vector of two components for the SS95 test, each component being a character string. The first component specifies the method for obtaining null distributions when the null hypothesis contains no variance components other than the error variance; the second component specifies the method for obtaining null distributions when the null hypothesis contains additional variance parameters other than the error variance. Currently available choices are &quot;ChiBarSq&quot; (Chi-bar-square asymptotic null; c.f., \texttt{pchibarsq}) and &quot;pboot&quot; (parametric bootstrap, i.e., the Monte Carlo approximation to the finite-sample null distribution; c.f., Qu et al [2013])</td>
</tr>
<tr>
<td>SS95.nsim</td>
<td>A large positive integer, specifying the number of simulations used to approximate the null, when SS95.method[1] is set to 'pboot'.</td>
</tr>
<tr>
<td>RLRT.method</td>
<td>A vector of two components for the RLRT test, each component being a character string. The first component specifies the method for obtaining null distributions when the null hypothesis contains no variance components other than the error variance; the second component specifies the method for obtaining null distributions when the null hypothesis contains additional variance parameters</td>
</tr>
</tbody>
</table>
other than the error variance. Currently available choices are "exact" (for either the exact null of Crainiceanu and Ruppert [2003] or the pseudo-likelihood heuristic method of Greven et al [2008]) and "pboot" (for the 1st component, this is an alias of "exact"; for the 2nd component, this is not yet implemented). See also RLRsim::RLRTSim.

**RLRT.nsim**
A large positive integer, specifying the number of simulations used to approximate the null, when RLRT.method[1] is set to 'exact'.

**information**
A character scalar, specifying the information matrix to be used in score-based tests. Available choices are given in informationTypes. By default, the expected information ("EI") is used. C.f., Verbeke and Molenberghs (2007).

**Value**
A list of class varCompTest.control with length equal to the length test argument. This can be passed as the control or fit.control arguments to varComp.test

**Author(s)**
Long Qu

**References**


**See Also**
varComp.test, CompQuadForm::davies, RLRsim::RLRTSim
vcov.varComp

Extracting Variance-Covariance Matrices

Description

Extracting (approximate) variance-covariance matrices for fixed-effect parameters, variance components, ratios of variance components to error variance, or the response variable.

Usage

```r
## S3 method for class 'varComp'
vcov(object, what = c("fixed", "beta", "random", "varComp", "var.ratio", "tau", "response", "y"), drop = TRUE, beta.correction=TRUE, ...)
```

Arguments

- `object`: An object of class `varComp`.
- `what`: A character vector (only the first element will be used) specifying what variance-covariance matrices are requested. "fixed" or "beta" request approximate variance-covariance of fixed-effect parameters (see details). "random" or "varComp" request the approximate variance-covariance matrix of variance components computed from the expected information matrix. "var.ratio" or "tau" requests approximate variance-covariance matrix of ratio of variance components to the error variance computed from the observed information matrix. "response" or "y" request the marginal variance of the response variable computed from the plug-in estimate.
- `drop`: A logical scalar, indicating whether zero variance components should be dropped from the results.
- `beta.correction`: A logical scalar, only applicable when `what`='beta', indicating whether the variance-covariance matrix for fixed effect estimates is corrected according to Kackar and Harville (1984).

Details

For fixed-effect parameters, the results is the plug-in estimate variance of generalized least squares estimates when `beta.correction`=FALSE; Otherwise, the Kackar and Harville (1984) correction will be used (default). For ratios of variance components to error variance, the result is the Hessian matrix. For response variable, the result is the plug-in estimate of the marginal variance. For variance components, the result is the plug-in estimate of inverse expected information matrix from the restricted likelihood.

Value

A numeric matrix of the requested variance-covariance.
Wald

Author(s)

Long Qu

References


See Also

*varComp* for the *varComp* object; *KR.varComp* for testing fixed effect parameters accounting for uncertainty in variance parameter estimates.

---

**Description**

`satterth.varComp` computes the denominator degrees of freedom for testing fixed-effect parameters under the variance component model using the Satterthwaite-type method. *KR.varComp* computes the denominator degrees of freedom and tests fixed-effect parameters under the variance component model using the Kenward-Roger method. These functions are most conveniently used by *fixef.varComp* and *anova.varComp*.

**Usage**

```r
satterth.varComp(object, Lmat, Vbet, svd.VLbet, X, K, V, ...)
KR.varComp(object, Lmat, Vbet, svd.VLbet, X, K, V, ...)
```

**Arguments**

- `object` An object of class varComp.
- `Lmat` A matrix specifying the linear combinations of fixed effect parameters to be tested for nullity. Each row is a linear combination.
- `Vbet` An optional matrix of variance-covariance matrix of fixed-effect parameter estimates.
- `svd.VLbet` An optional singular value decomposition of `Lmat` times `Vbet`.
- `X` Optional fixed-effect design matrix
- `K` Optional list of K matrices (the same as in *varComp.test*).
- `V` Optional covariance matrix of the response variable.
- `...` Place holder, not used.
Value

A numeric scalar of denominator degree of freedom. For `satterth.varComp`, the 'individual.df' attribute will contain a vector of numerator degrees of freedom for each row of `Lmat`. For `KR.varComp`, the result will contain the following attributes:

- `numDF`: The numerator degree of freedom, i.e., the rank of `Lmat`.
- `Scale`: A positive numeric scalar, to be multiplied to raw F-statistics before calculating p-values.
- `F value`: The scaled F-statistic, after adjusting for the variance estimate of fixed effect parameter estimates, and multiplied by the scaling factor.
- `Pr(>F)`: A numeric scalar of Kenward-Roger p-value.
- `vcov.beta`: The adjusted variance-covariance estimate of fixed effect parameter estimates.

Author(s)

Long Qu

References


See Also

`fixef.varComp`, `anova.varComp`

Examples

```r
library(nlme)
data(Oxide)
lmef = lme(Thickness~Source, Oxide, ~1|Lot/Wafer)
anova(lmef)
vcf = varComp(Thickness~Source, Oxide, ~Lot/Wafer)
KR.varComp(vcf, matrix(c(0,1), 1))  # test Source effect
satterth.varComp(vcf, matrix(c(0,1), 1))  # d.f. for testing Source effect
```
Index

*Topic algebra
    cholRoot, 7
*Topic array
    normalizeTrace, 19
*Topic datasets
    symbols, 24
*Topic distribution
    Chi-bar-square, 6
    get.seed, 10
*Topic htest
    anova.varComp, 3
    p.value.varComp.test, 21
    varComp.test, 32
    varCompTest.control, 35
    Wald, 39
*Topic manip
    safeseq, 23
*Topic methods
    print.varComp, 22
*Topic models
    coef.varComp, 8
    Internal formula manipulation, 10
    is.formula, 13
    logLik.varComp, 16
    minite, 17
    model.matrix.varComp, 18
    vcov.varComp, 38
*Topic multivariate
    kernels, 14
    varComp, 25
*Topic nonlinear
    vcov.varComp, 38
*Topic optimize
    minite, 17
    varComp.control, 30
*Topic package
    varComp-package, 2
*Topic print
    anova.varComp, 3
    print.varComp, 22
    print.varComp.test, 23
*Topic regression
    anova.varComp, 3
    coef.varComp, 8
    varComp, 25
*Topic smooth
    varComp, 25

AEI (internals), 11
AM (kernels), 14
am (kernels), 14
anova.varComp, 3, 39, 40
AOI (internals), 11
base::.Random.seed, 10
base::chol, 7, 8
base::diag, 20
base::seq, 23, 24

callList2Terms (Internal formula manipulation), 10
Chi-bar-square, 6
cholRoot, 7, 15, 18
coef.varComp, 8, 29
colon (symbols), 24
CompQuadForm::davies, 36, 37
defunct Functions, 9
doFit.varComp (varComp), 25

EI (internals), 11
fixed.effects (anova.varComp), 3
fixef (anova.varComp), 3
fixef.varComp, 2, 22, 29, 39, 40

g_base.seed, 10
gradFunc (internals), 11
hess (internals), 11
IBS, 27
IBS (kernels), 14
ibs (kernels), 14
informationTypes (varComp.control), 30
Internal formula manipulation, 10
internals, 11
is.formula, 13
kernels, 14
KR.varComp, 5, 28, 29, 39
KR.varComp (Wald), 39
Lin0 (kernels), 14
lin0 (kernels), 14
LinScoreWeightingMethods (varCompTest.control), 35
logLik.varComp, 16, 29
Matrix::chol, 8
mchibarsq (Chi-bar-square), 6
Minkowski (kernels), 14
minkowski (kernels), 14
minkaue, 17, 30
model.matrix.varComp, 18, 29

nlme::lme, 3, 16, 25, 29
nlminb.control, 31
nlminb.control (varComp.control), 30
normalizeTrace, 15, 19
Not Implemented Functions, 20

obj (internals), 11
obj2 (internals), 11
OI (internals), 11

p.value (p.value.varComp.test), 21
p.value.varComp.test, 21
pchibarsq, 36
pchibarsq (Chi-bar-square), 6
plus (symbols), 24
Polyk (kernels), 14
polyk (kernels), 14
PREML (internals), 11
preprocPREML (internals), 11
print.varComp, 22, 29
print.varComp.test, 23
print.varCompFixEf (anova.varComp), 3
Quad1 (kernels), 14
quad1 (kernels), 14

RLRsim::RLRTSim, 37
safeseq, 23
satterth.varComp, 5, 28, 29
satterth.varComp (Wald), 39
score (internals), 11
sortTerm (Internal formula manipulation), 10
SPA3G::KERNEL, 14, 15
splitTerm (Internal formula manipulation), 10
star (symbols), 24
stats::dist, 14, 15
stats::formula, 11, 13
stats::glm, 26
stats::lm, 25, 26, 29
stats::nlminb, 31
stats::pchisq, 6, 7
summary.varComp, 29
summary.varComp (print.varComp), 22
symbols, 24

tcrossprod, 18
tilde (symbols), 24

updateDenom (internals), 11
updateLI (internals), 11
updateLIKLI (internals), 11
updateLIy (internals), 11
updateNegGrad (internals), 11
updateNegHess (internals), 11
updateNums (internals), 11
updateNums2 (internals), 11
updateNumsPart (internals), 11
updateTr1 (internals), 11
updateTr2 (internals), 11

V (internals), 11
varComp, 2, 9, 10, 15–19, 25, 30–34, 39
varComp-package, 2
varComp.control, 26–28, 30, 33
varComp.fit, 17
varComp.LinScore.approximation (defunct Functions), 9
varComp.LinScore.beta (Not Implemented Functions), 20
varComp.LinScore.cumulant3 (Not Implemented Functions), 20
varComp.LinScore.LC12 (defunct Functions), 9
INDEX

varComp.LinScore.LC12Boundary (defunct Functions), 9
varComp.LinScore.Normal (internals), 11
varComp.LinScore.pboot (Not Implemented Functions), 20
varComp.LinScore.saddlepoint (Not Implemented Functions), 20
varComp.LinScore.Satterthwaite (internals), 11
varComp.LinScore.SSAS155 (internals), 11
varComp.LinScore.test (internals), 11
varComp.RLRT.test (internals), 11
varComp.RWD88.test (Not Implemented Functions), 20
varComp.SS95.test (internals), 11
varComp.test, 2, 10, 21, 29, 32, 35, 37, 39
varComp.test.2modelDoTest (internals), 11
varComp.test.altDoTest (internals), 11
varComp.test.Common (internals), 11
varComp.test.nulDoTest (internals), 11
varComp.VM03.test (internals), 11
varComp.Wald.test (Not Implemented Functions), 20
varCompScoreTests
   (varCompTest.control), 35
varCompTest.control, 33, 34, 35
varCompTests, 33
varCompTests (varCompTest.control), 35
vcov.varComp, 29, 38

WAI (internals), 11
Wald, 39
wchibarsq (Chi-bar-square), 6