Package 'coxme'

January 4, 2018

Title Mixed Effects Cox Models

Priority optional

Version 2.2-7

Depends survival (>= 2.36.14), methods, bdsmatrix(>= 1.3)

Imports nlme, Matrix (>= 1.0)

Suggests mvtnorm, kinship2

LinkingTo bdsmatrix

LazyData Yes

LazyLoad Yes

Description Cox proportional hazards models containing Gaussian random effects, also known as frailty models.

License LGPL-2

NeedsCompilation yes

Author Terry M. Therneau [aut, cre]

Maintainer Terry M. Therneau <therneau@mayo.edu>

Repository CRAN

Date/Publication 2018-01-04 14:49:58 UTC

R topics documented:

`anova.coxme` *Analysis of Deviance for a Cox model.*

Description

Compute an analysis of deviance table for one or more Cox model fits.

Usage

```
## S3 method for class 'coxme'
anova(object, ...,  test = 'Chisq')
```

Arguments

`object`	An object of class `coxme` or `coxph`
`...`	Further `coxme` objects
`test`	a character string. The appropriate test is a chisquare, all other choices result in no test being done.

Details

Specifying a single object gives a sequential analysis of deviance table for that fit. That is, the reductions in the model log-likelihood as each term of the formula is added in turn are given in as the rows of a table, plus the log-likelihoods themselves.

If more than one object is specified, the table has a row for the degrees of freedom and loglikelihood for each model. For all but the first model, the change in degrees of freedom and loglik is also given. (This only make statistical sense if the models are nested.) It is conventional to list the models from smallest to largest, but this is up to the user.

The table will optionally contain test statistics (and P values) comparing the reduction in loglik for each row.

Value

An object of class `"anova"` inheriting from class `"data.frame"`.

Warning

The comparison between two or more models by anova or will only be valid if they are fitted to the same dataset. This may be a problem if there are missing values.

See Also

coxme, anova.

Examples

```
# Testing a shrunken estimate of ECOG performace status
fit1 <- coxph(Surv(time, status) ~ age + sex, data=lung,
              subset=(!is.na(ph.ecog)))
fit2 <- coxme(Surv(time, status) ~ age + sex + (1|ph.ecog), lung)
anova(fit1,fit2)
```

---

coxme *Fit a mixed effects Cox model*

Description

Fit a Cox model containing mixed (random and fixed) effects. Assume a Gaussian distribution for the random effects.

Usage

```
coxme(formula, data, weights, subset, na.action, init, control,
ties = c("efron", "breslow"),
varlist, vfixed, vinit,  x = FALSE, y = TRUE,
refine.n = 0, random, fixed, variance, ...)
```

Arguments

formula	a two-sided formula with a survival object as the left hand side of a ~ operator and the fixed and random effects on the right.
data	an optional data frame containing the variables named in the formula.
subset, weights, na.action	further model specifications arguments as in lm; see there for details.
init	optional initial values for the fixed effects.
control	optional list of control options. See coxme.control for details.
ties	method for handling exact ties in the survival time.
varlist	the variance family to be used for each random term. If there are multiple terms it will be a list of variance functions. The default is coxmeFull. Alternatively it can be a list of matrices, in which case the coxmeMlist function is used.

vfixed	optional named list or vector used to fix the value of one or more of the variance terms at a constant.
vinit	optional named list or vector giving suggested starting values for the variance.
x	if TRUE the X matrix (fixed effects) is included in the output object
y	if TRUE the y variable (survival time) is included in the output object
refine.n	number of samples to be used in a monte-carlo estimate of possible error in the log-likelihood of the fitted model due to inadequacy of the Laplace approximation.
fixed, random, variance	In the preliminary version of coxme the fixed and random effects were separate arguments. These arguments are included for backwards compatability, but are depreciated. The variance argument is a depreciated alias for vfixed.
...	any other arguments are passed forward to coxme.control.

Value

An object of class coxme.

Author(s)

Terry Therneau

References

S Ripatti and J Palmgren, Estimation of multivariate frailty models using penalized partial likelihood, Biometrics, 56:1016-1022, 2000.

T Therneau, P Grambsch and VS Pankratz, Penalized survival models and frailty, J Computational and Graphical Statistics, 12:156-175, 2003.

See Also

coxmeFull, coxmeMlist, coxme.object

Examples

```
# A non-significant institution effect
fit1 <- coxph(Surv(time, status) ~ ph.ecog + age, data=lung,
              subset=(!is.na(inst)))
fit2 <- coxme(Surv(time, status) ~ ph.ecog + age + (1|inst), lung)
anova(fit1, fit2)

# Shrinkage effects (equivalent to ridge regression)
temp <- with(lung, scale(cbind(age, wt.loss, meal.cal)))
rfit <- coxme(Surv(time, status) ~ ph.ecog + (temp | 1), data=lung)
```

coxme.control *Auxillary parameters for controlling coxme fits.*

Description

Auxillary function which packages the optional parameters of a coxme fit as a single list.

Usage

```
coxme.control(eps = 1e-08, toler.chol = .Machine$double.eps^0.75,
iter.max = 20, inner.iter = Quote(max(4, fit0$iter+1)),
sparse.calc = NULL,
optpar = list(method = "BFGS", control=list(reltol = 1e-5)),
refine.df=4, refine.detail=FALSE, refine.method="control",
sparse=c(50, .02),
varinit=c(.02, .1, .4, .8)^2, corinit = c(0, .3))
```

Arguments

eps	convergence criteria for the partial likelihood
toler.chol	tolerance for the underlying Cholesky decomposition. This is used to detect singularity (redundant variables).
iter.max	maximum number of iterations for the final fit
inner.iter	number of iterations for the 'inner loop' fits, i.e. when the partial likelihood is the objective function of optim. The default is to use one more iteration than the baseline coxph model fit0. The baseline model contains only the fixed effects, and is as part of the setup by the main program. The minimum value of 4 applies most often to the case where there are no fixed effects.
sparse.calc	choice of method 1 or 2 for a particular portion of the calculation. This can have an effect on run time for problems with thousands of random effects.
optpar	parameters passed forward to the optim routine.
refine.df	the degrees of freedom for the t-distribution used to draw random samples for the refine.n option
refine.detail	this option is mostly for debugging. If TRUE then an extra component refine.detail will be present in the output which contains intermediate variables from the iterative refinement calculation.
refine.method	method by which the control calculations are done. This is a current research/development question, the option will likely disappear at some future date, and users should ignore it.
sparse	rule for deciding sparsity of a random effect, see details below.
varinit	the default set of starting values for variances, used if no vinit argument is supplied in the coxme call.
corinit	the default set of starting values for correlations.

Details

The main flow of coxme is to use the optim routine to find the best values for the variance parameters. For any given trial value of the variance parameters, an inner loop maximizes the partial likelihood to select the regression coefficients beta (fixed) and b (random). Within this loop cholesky decomposition is used. It is critical that the convergence criteria of inner loops be less than outer ones, thus toler.chol < eps < reltol.

If no starting values are supplied for the variances of the random effects then a grid search is performed to select initial values for the main iteration loop. The default values given here are based on experience but without any formal arguments for their optimality. We have found that the estimated standard deviation of a random effect is often between .1 and .3, corresponding to exp(.1)= 1.1 to exp(.3)= 1.35 fold "average" relative risks associated with group membership. This is biologically reasonable for a latent trait. Other common solutions ane a small random effect corresponding to only 1–5% change in the hazard or likelihood that is maximized at the boundary value of 0 variance. Variances greater than 2 are very unusual. Because we use the log(variance) as our iteration scale the 0–.001 portion of the variance scale is stretched out giving a log-likelihood surface that is almost flat; a Newton-Raphson iteration starting at log(.2) may have log(.0001) as its next guess and get stuck there, never finding a true maximum that lies in the range of .01 to .05. Corrleation paramters seem to need fewer starting points.

The sparse option controls a sparse approximation in the code. Assume we have a mixed effects model with a random intercept per group, and there are 1000 groups. In a Cox model (unlike a linear mixed effects model) the resulting second derivative matrix used during the solution will be 1000 by 1000 with no zeros, and fitting the model can consume a large amount of both time and memory. However, it is almost sparse, in that elements off the diagonal are very small and can often be ignored. Computation with a sparse matrix approximation will be many times faster. Luckily, as the number of groups increases the accuracy of the approximation also increases. If `sparse=c(50, .03)` this states that sparse approximation will be employed for any grouping variable with 50 or more levels, and off diagonal elements that relate any two levels both of which represent .03 of less of the total sample will be treated as zero.

Value

a list of control parameters

Author(s)

Terry Therneau

See Also

coxme

`coxme.object` *Coxme regression output object*

Description

This contains further description of the output object created by a coxme call. Most components can be accessed with extractor functions, which is the safer route since details of the object will likely change over time.

Details

The structure of each element of the random effects coefficients (obtained with ranef) and variances (VarCorr) depend on the variance functions, i.e., the functions used in the varlist argument. Since users can write their own variance functions this format can never be completely known.

Value

coefficients	the coefficients of the fixed effects. Use the fixef function to extract them.
frail	the coefficients of the random effects. Use the ranef function to extract them. These are always stored as a list with one member per random effect; each parenthesised term in the model is a random effect. In a linear mixed effects model the fixed effects and the variances of the random effects can be obtained without explicitly computing the coefficients of the random effects, the latter are called BLUP estimates and are computed later if at all. This is not the case for a Cox model, there the random effect coefficients are a required part of the iteration process and so are always present in the final model.
vcoef	the variances of the random effects. Use the VarCorr function to extract them. These are always stored as a list with one member per random effect.
variance	the variance-covariance matrix of the coefficient vector, including both fixed and random terms. The random effects are listed first. This will often be a sparse matrix. The vcov function will extract the fixed effects portion, which is always dense.
loglik	the log-likelihood vector from the fit. The first element is the loglik at the initial values, the second is the integrated partial likelihood at the solution (IPL), the third is the penalized partial likelihood at the solution(PPL).
df	degrees of freedom for the IPL and the PPL solutions.
hmat	sparse Cholesky factorization of the information matrix.
iter	outer and inner iterations performed. For each trial value of the variance parameters an Cox model partial likelihood must be solved; the outer iterations is the reported number from the optim() routine which handles the variance parameters, the inner iterations is the cumulative number of partial likelihood iterations.
control	a copy of the coxph.control parameters used in the fit.
ties	the computational method used for ties.
u	the vector of first derivatives of the PPL, at the solution.
means,scale	means and scale for each predictor, used internally to scale the problem.
linear.predictor	the vector of linear predictors.
n	vector containing the number of events and the number of observations in the fitting data set.

terms	the terms object from the fixed effects of the model formula. Access using the terms function.
formulaList	the fixed and random portions of the formula, separated
na.action	the missing value attributes of the data, if any
x,y,model	optional: the x matrix, response, for model frame. These depend on the corresponding arguments in the call.
call	a copy of the call to the routine

---

coxmeFull *Variance family function for coxme fits.*

Description

This function sets up the default variance family information for a mixed effects survival model fit with coxme.

Usage

```
coxmeFull(collapse = FALSE)
```

Arguments

collapse	Form for fitting a nested effect, either standard or collapsed. The latter appears to be more numerically stable (still under research).

Details

Coxme variance families create a list with three functions: initialize, generate, and wrapup, that determine how the variance structure of a fit is modeled.

Value

an object of class coxvar.

Author(s)

Terry Therneau

See Also

coxme

coxmeMlist *Coxme variance function*

Description

This variance function accepts a list of matrices, which define a correlation structure for a coxme fit.

Usage

```
coxmeMlist(varlist, rescale = FALSE, pdcheck = TRUE, positive = TRUE)
```

Arguments

varlist	a list containing one or more matrix or bdsmatrix objects.
rescale	if TRUE, each input matrix is rescaled to have a diagonal of 1. (Kinship matrices for instance are often generated with a diagonal of .5 and would be multiplied by 2).
pdcheck	check each matrix to ensure that it is positive definite
positive	constrain coefficients to be positive. This may also be a vector of the same length as varlist

Details

If two matrices A and B were given, this fits the variance structure $V = \sigma_1^2 A + \sigma_2^2 B$, where the variances σ_1^2 and σ_2^2 are parameters that will be optimized by coxme, treating A and B as fixed.

Value

a coxme variance family object, used by coxme in the fitting process.

Author(s)

Terry Therneau

See Also

coxme

eortc *Simulated data set based on an EORTC trial*

Description

This is a simulated surival data set for investigating random center effects. To make it realistic, the number of centers and their sizes is based on an EORTC cancer trial.

Usage

```
data(eortc)
```

Format

A data frame with 2323 observations on the following 4 variables.

`y` survival time

`uncens` 0=alive, 1=dead

`center` enrolling center, a number from 1 to 37

`trt` treatment arm, 0 or 1

Details

This is used in the test suite for the code.

Source

PhD thesis work of Jose Cortinas Abrahantes

References

Cortinas Abrahantes, Jose; Burzykowski, Tomasz (2002), A version of the EM algorithm for proportional hazards models with random effects , Published in: Lecture Notes of the ICB Seminars. p. 15-20

Examples

```
data(eortc)
coxme(Surv(y, uncens) ~ trt + (trt| center) + strata(center), eortc)
```

`expand.nested` *Expand nested factors*

Description

Expand out the data frame for a nested factor such as (1| a/b). This is used by the variance function routines of `coxme`.

Usage

```
expand.nested(x)
```

Arguments

`x`	A data frame containing the nesting variables

Details

The initialize function of a `coxme` variance family is passed, as one of its arguments, a data frame `G` containing the grouping variables, each of which is a factor.. Assume a nested factor `(1| a/b)` in the model formula and a data set whose first few rows are:

a	b
1	1
1	2
2	1

The function will replace the second column with a variable named `a/b` and values of 1/1, 1/2, 2/1, etc.

Value

an updated data frame

Author(s)

Terry Therneau

See Also

`coxme`, `coxmeMlist`

fixed.effects *Import from package* **nlme**

Description

The `fixed.effects` and `fixef` methods are imported from package **nlme**. Help is available here: `nlme::fixed.effects`.

fixef.coxme *Extraction functions for Coxme*

Description

Extract the fixed effects, randome effects, variance of the fixed effects, or variance of the random effects from a coxme model.

Usage

```
## S3 method for class 'coxme'
fixef(object, ...)
## S3 method for class 'coxme'
ranef(object, ...)
## S3 method for class 'coxme'
vcov(object, ...)
## S3 method for class 'coxme'
VarCorr(x, ...)
```

Arguments

`object`	an object inheriting from class `coxme` representing the result of a mixed effects Cox model.
`x`	an object inheriting from class `coxme` representing the result of a mixed effects Cox model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Value

the fixed effects are a vector and the variance of the fixed effects is a matrix. The random effects will be a list with one element for each random effects terms, as will be their variance.

Author(s)

Terry Therneau

See Also

coxme, random.effects, fixed.effects

Examples

```
rat1 <- coxme(Surv(time, status) ~ rx + (1|litter), rats)
fixed.effects(rat1)
vcov(rat1)
random.effects(rat1)[[1]]  #one value for each of the 50 litters
VarCorr(rat1)
```

`fixef.lmekin` *Extraction functions for Lmekin*

Description

Extract the fixed effects, random effects, variance of the fixed effects, or variance of the random effects from a linear mixed effects model fit with lmekin.

Usage

```
## S3 method for class 'lmekin'
fixef(object, ...)
## S3 method for class 'lmekin'
ranef(object, ...)
## S3 method for class 'lmekin'
vcov(object, ...)
## S3 method for class 'lmekin'
VarCorr(x, ...)
## S3 method for class 'lmekin'
logLik(object, ...)
```

Arguments

`object`	an object inheriting from class `lmekin` representing the result of a mixed effects model.
`x`	an object inheriting from class `lmekin` representing the result of a mixed effects model.
`...`	some methods for this generic require additional arguments. None are used in this method.

Details

For the random effects model $y = X\beta + Zb + \epsilon$, let σ^2 be the variance of the error term ϵ. Let $A = \sigma^2 P$ be the variance of the random effects b. There is a computational advantage to solving the problem in terms of P instead of A, and that is what is stored in the returned lmekin object. The `VarCorr` function returns elements of P; the print and summary functions report values of A. Pinhiero and Bates call P the precision factor.

Value

the fixed effects are a vector and vcov returns their variance/covariance matrix. The random effects are a list with one element for each random effect. The ranef component contains the coefficients and VarCorr the estimated variance/covariance matrix. The logLik method returns the loglikelihood along with its degrees of freedom.

Author(s)

Terry Therneau

References

J Pinheiro and D Bates, Mixed-effects models in S and S-Plus. Springer, 2000.

See Also

lmekin, random.effects, fixed.effects, link{vcov}, VarCorr

Examples

```
data(ergoStool, package="nlme")  # use a data set from nlme
efit <-  lmekin(effort ~ Type + (1|Subject), ergoStool)
ranef(efit)
```

---

lmekin *Fit a linear mixed effects model*

Description

The lmekin function fits a linear mixed effects model, with random effects specified in the same structure as in the coxme function.

Usage

```
lmekin(formula, data, weights, subset, na.action, control,
varlist, vfixed, vinit, method = c("ML", "REML"),
x = FALSE, y = FALSE, model=FALSE,
random, fixed, variance, ...)
```

Arguments

formula	a two-sided formula with the response as the left hand side of a ~ operator and the fixed and random effects on the right.
data	an optional data frame containing the variables named in the formula.
subset, weights, na.action	further model specifications arguments as in lm; see there for details.
control	optional list of control options. See coxme.control for details.

varlist	the variance family to be used for each random term. If there are multiple terms it will be a list of variance functions. The default is coxmeFull. Alternatively it can be a list of matrices, in which case the coxmeMlist function is used.
vfixed	optional named list or vector used to fix the value of one or more of the variance terms at a constant.
vinit	optional named list or vector giving suggested starting values for the variance.
method	fit using either maximum likelihood or restricted maximum likelihood
x	if TRUE the X matrix (fixed effects) is included in the output object
y	if TRUE the y variable is included in the output object
model	if TRUE the model frame is included in the output object
fixed, random, variance	In an earlier version of lmekin the fixed and random effects were separate arguments. These arguments are included for backwards compatability, but are depreciated. The variance argument is a depreciated alias for vfixed.
...	any other arguments are passed forward to coxme.control.

Details

Let $A = \sigma^2 B$ be the variance matrix of the random effects where σ^2 is the residual variance for the model. Internally the routine solves for the parameters of B, computing A at the end. The vinit and vfixed parmaters refer to B, however.

It is possible to specify certain models in lmekin that can not be fit with lme, in particular models with familial genetic effects, i.e., a *kinship* matrix, and hence the name of the routine. Using user-specified variance functions an even wider range of models is possible. For simple models the specification of the random effects follows the same form as the lmer function. For any model which can be fit by both lmekin and lmer, the latter routine would normally be prefered due to a much wider selection of post-fit tools for residuals, prediction and plotting.

Much of the underlying model code for specification and manipulation of the random effects is shared with the coxme routine. In fact lmekin was originally written only to provide a test routine for those codes, and no expectation that it would find wider utility.

Value

An object of class lmekin.

Author(s)

Terry Therneau

See Also

lmekin.object, coxme

Examples

```
  data(ergoStool, package="nlme")  # use a data set from nlme
  fit1 <- lmekin(effort ~ Type + (1|Subject), data=ergoStool)
  ## Not run:
  # gives the same result
  require(nlme)
  fit2 <- lme(effort ~ Type, data=ergoStool, random= ~1|Subject,
            method="ML")

## End(Not run)
```

`lmekin.control` *Auxillary parameters for controlling lmekin fits.*

Description

Auxillary function which packages the optional parameters of a `lmekin` fit as a single list.

Usage

```
lmekin.control(
optpar = list(method = "BFGS", control=list(reltol = 1e-8)),
varinit=c(.02, .1, .8, 1.5)^2, corinit = c(0, .3))
```

Arguments

`optpar`	parameters passed forward to the `optim` routine.
`varinit`	the default grid of starting values for variances, used if no `vinit` argument is supplied in the `lmekin` call.
`corinit`	the default grid of starting values for correlations.

Details

The main flow of `lmekin` is to use the `optim` routine to find the best values for the variance parameters. For any given trial value of the variance parameters, a subsidiary computation maximizes the likelihood to select the regression coefficients beta (fixed) and b (random).

If no starting values are supplied for the variances of the random effects then a grid search is performed to select initial values for the main iteration loop. The variances and correlations are all scaled by σ^2, making these starting estimates scale free, e.g., replacing y by 10*y in a data set will change σ but not the internal representation of any other variance parameters. Because we use the log(variance) as our iteration scale the 0–.001 portion of the variance scale is stretched out giving a log-likelihood surface that is almost flat; a Newton-Raphson iteration starting at log(.2) may have log(.0001) as its next guess and get stuck there, never finding a true maximum that lies in the range of .01 to .05. Corrleation paramters seem to need fewer starting points.

Value

a list of control parameters

Author(s)

Terry Therneau

See Also

lmekin

`lmekin.object` *lmekin object*

Description

This class of object is returned by the `lmekin` function to represent a fittd mixed effect linear model. Objects of this class currently have methods for `print` and `residuals`.

Value

A list with the folling components:

`coefficients`	a list with components `fixed` and `random`; the first will be NULL for a model with no fixed effects. The `random` component is itself a list, with an element for each random effect.
`var`	the variance matrix of the fixed effects
`vcoef`	the parameters of the variance matrix of the random effects.
`residuals`	vector of residuals from the fit
`method`	either "ML" or "REML"
`loglik`	the log-likelihood for the fitted model
`sigma`	the estimated residual error
`n`	number of observations used
`call`	a copy of the call
`na.action`	this will be present if any observations were removed due to missing values

Author(s)

Terry Therneau

See Also

lmekin, coxmeFull, coxmeMlist

logLik.coxme *The logLik method for coxme objects*

Description

logLik is most commonly used for a model fitted by maximum likelihood, and some uses, e.g. by AIC. This method allows generic functions to easily extract the log-likelihood of a coxme model.

Usage

```
## S3 method for class 'coxme'
logLik(object, type = c("penalized", "integrated"), ...)
```

Arguments

object	a fitted coxme model
type	which of the two types of partial likelihood to extract
...	used by other methods

Details

The likelihood for a mixed effects Cox model can be viewed in two ways: the ordinarly partial likelihood, where the random effects act only as a penalty or constraint, or a partial likelihood where the random effect has been integrated out. Both are valid.

Value

Returns an object of class logLik.

See Also

logLik

predict.coxme *Predictions for a coxme object.*

Description

Return predicted values from a coxme fit.

Usage

```
## S3 method for class 'coxme'
predict(object, newdata, type = c("lp", "risk"))
```

Arguments

object	an object of class coxme, from the fit of a mixed effects survival model
newdata	new data set, not yet supported
type	type of prediction

Value

a vector of predicted values

See Also

coxme

---

print.coxme *Print method for a coxme fit.*

Description

Print out the result of a coxme fit.

Usage

```
## S3 method for class 'coxme'
print(x, rcoef=FALSE, digits = options()$digits, ...)
## S3 method for class 'coxme'
summary(object, ...)
```

Arguments

x	an object of class coxme, from the fit of a mixed effects survival model.
rcoef	print the random (penalized) coefficients, as well as the fixed ones.
digits	number of significant digits to print
object	an object of class coxme, from the fit of a mixed effects survival model.
...	optional arguments

Note

The summary function is currently identical to the print function. The default in R is for print() to be the short form printout and summary() the long form.

Author(s)

Terry Therneau

See Also

coxme

print.lmekin *Print function for lmekin*

Description

Print out the result of an lmekin fit.

Usage

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

Arguments

x	an object of class lmekin.
...	generic arguments to print, unused.

Details

The print function current has no options. This should one day improve.

Author(s)

Terry Therneau

See Also

lmekin

ranef *Import from package* **nlme**

Description

The ranef and random.effects methods are imported from package **nlme**. Help is available here: nlme::random.effects.

VarCorr *Import from package* **nlme**

Description

The VarCorr method is imported from package **nlme**. Help is available here: nlme::VarCorr.

Index