Package: gremlin (via r-universe)

September 8, 2024

Type Package

Title Mixed-Effects REML Incorporating Generalized Inverses

Version 1.0.2

URL http://github.com/matthewwolak/gremlin

BugReports http://github.com/matthewwolak/gremlin/issues

Depends Matrix

Imports methods, nlme

LazyData yes

NeedsCompilation yes

Description Fit linear mixed-effects models using restricted (or residual) maximum likelihood (REML) and with generalized inverse matrices to specify covariance structures for random effects. In particular, the package is suited to fit quantitative genetic mixed models, often referred to as 'animal models'. Implements the average information algorithm as the main tool to maximize the restricted log-likelihood, but with other algorithms available.

License GPL-3 | file LICENSE

Encoding UTF-8

RoxygenNote 7.1.0

Repository https://matthewwolak.r-universe.dev

RemoteUrl https://github.com/matthewwolak/gremlin

RemoteRef HEAD

RemoteSha fb14161332666819e51310d3c564d9ac096d855b

Contents

`gremlin-package` *Mixed-Effects REML Incorporating Generalized Inverses*

Description

Fit linear mixed-effects models using restricted (or residual) maximum likelihood (REML) and with generalized inverse matrices to specify covariance structures for random effects. In particular, the package is suited to fit quantitative genetic mixed models, often referred to as 'animal models' (Henderson 1973). Implements the average information algorithm (Johnson & Thompson 1995; Gilmour et al. 1995; Meyer & Smith 1996) as the main tool to maximize the restricted log-likelihood, but with other algorithms available.

Details

The average information algorithm combined with sparse matrix techniques can potentially make model fitting very efficient.

Author(s)

Maintainer: Matthew Wolak <matthewwolak@gmail.com> (ORCID)

References

Henderson, C.R. 1973. Sire evaluation and genetic trends. Journal of Animal Science 1973:10-41.

Johnson, D.L. and R. Thompson. 1995. Restricted maximum likelihood estimation of variance components for univariate animal models using sparse matrix techniques and average information. Journal of Dairy Science 78:449-456.

Gilmour, A.R., R. Thompson, and B.R. Cullis. 1995. Average information REML: An efficient algorithm for variance parameter estimation in linear mixed models. Biometrics 51:1440-1450.

Meyer, K. and S.P. Smith. 1996. Restricted maximum likelihood estimation for animal models using derivatives of the likelihood. Genetics, Selection, and Evolution 28:23-49.

Mrode, R.A. 2005. Linear Models for the Prediction of Animal Breeding Values, 2nd ed. CABI Publishing, Cambridge, MA, USA.

See Also

Useful links:

- http://github.com/matthewwolak/gremlin
- Report bugs at http://github.com/matthewwolak/gremlin/issues

Examples

```
## Not run:
  # Following the example from Mrode 2005, chapter 11.
  library(nadiv)  #<-- to construct inverse of the numerator relatedness matrix
  Ainv <- makeAinv(Mrode11[, 1:3])$Ainv

  gr11lmm <- gremlin(WWG11 ~ sex - 1,
random = ~ calf,
data = Mrode11,
ginverse = list(calf = Ainv),
Gstart = matrix(0.2), Rstart = matrix(0.4),  #<-- specify starting values
maxit = 15,    #<-- maximum iterations
     v = 2, vit = 1,  #<-- moderate screen output (`v`) every iteration (`vit`)
     algit = "AI")  #<-- only use Average Information algorithm iterations
  summary(gr11lmm)

  # Compare the model to a Linear Model with no random effects
  ## Use `update()` to update the model
  gr11lm <- update(gr11lmm, random = ~ 1)  #<-- `~1`=drop all random effects
  summary(gr11lm)

  # Do analysis of variance between the two models
  ## See AIC or evaluate likelihood ratio against a Chi-squared distribution
  anova(gr11lm, gr11lmm)

## End(Not run)
```

---

anova.gremlin *anova() for gremlin objects*

Description

REML Likelihood Ratio Tests for gremlin models using anova()

Usage

```
## S3 method for class 'gremlin'
anova(object, ..., model.names = NULL)
```

Arguments

object	An object of class 'gremlin'.
...	Additional objects of class 'gremlin'.
model.names	Optional character vector with model names to be used in the anova table

Value

A data.frame containing the nested comparison of model objects via a REML likelihood ratio test.

Author(s)

<matthewwolak@gmail.com>

Examples

```
mod11 <- gremlin(WWG11 ~ sex - 1,
random = ~ sire,
data = Mrode11)
mod11red <- gremlinR(WWG11 ~ sex - 1, data = Mrode11)
anova(mod11, mod11red)
```

---

covFun *(Co)variance parameter transformations.*

Description

Converts lists of (co)variance parameters either between list and vector format or between the theta and nu scales.

Usage

```
stTrans(x)

conTrans(Gcon, Rcon)

start2theta(Gstart, Rstart, name = NULL)

matlist2vech(theta)

vech2matlist(vech, skeleton)

theta2nu_trans(theta)

nu2theta_trans(nu)

theta2nu_lambda(theta, thetaG, thetaR)
```

```
nu2theta_lambda(nu, sigma2e, thetaG, thetaR)

nuVar2thetaVar_lambda(object)

nuAI2thetaAIinv_lambda(object)

nu2theta_noTrans(nu, thetaG, thetaR)
```

Arguments

x, theta, nu	A list of matrices containing the (co)variance parameters of the model.
Gcon, Rcon	A list of starting (co)variance constraints for the G-structure (random effects terms) or R-structure (residual).
Gstart, Rstart	A list of starting (co)variance values for the G-structure (random effects terms) or R-structure (residual).
name	An (optional) character vector containing the (co)variance component names.
vech	A vector of (co)variance parameters.
skeleton	An example structure to map vech onto.
thetaG, thetaR	A vector indexing the G-structure or R-structure components, respectively.
sigma2e	A numeric estimate of the factored out residual variance from the mixed model equations (i.e., the 'lambda' scale) σ_e^2.
object	An object of class 'gremlin'.

Details

- stTrans Transform start parameters into lower triangle matrices of class dsCMatrix.
- conTrans Transformation of starting constraints to correct format.
- start2theta Converts lists of starting values for (co)variance parameters to a theta object used to structure the (co)variance components within gremlin.
- matlist2vech Converts a list of (co)variance parameter matrices to a vector with a "skel" attribute.
- vech2matlist Converts a vector of (co)variance parameters to a list of covariance matrices.
- theta2nu_trans Transforms theta to nu scale by taking the Cholesky factor of each covariance matrix and then replacing the diagonals with their (natural) logarithms. Done to ensure matrices are positive definite.
- nu2theta_trans Back transformation from theta2nu_trans: exponentiates the diagonal elements of each matrix then calculates the cross-product.
- theta2nu_lambda Transformation that factors out a residual variance so that nu contains the 'lambda' parameterization: ratios of variance parameters with the residual variance.
- nu2theta_lambda Back transformation from theta2nu_lambda.
- nuVar2thetaVar_lambda Transformation of Sampling Variances from lambda Scale for theta.
- nuAI2thetaAIinv_lambda Transform AI matrix from lambda Scale to AI-inverse of theta.
- nu2theta_noTrans Structures theta when not transformed.

Value

Functions are specified to mostly return either a `list` of matrices (structure as defined by the "skel" attribute or in the `skeleton` object) or a `vector` containing the (co)variance parameters of the model. Additional list elements returned can be:

thetaG A `vector` indexing the G-structure components.

thetaR A `vector` indexing the R-structure components.

Alternatively, `nuVar2thetaVar_lambda` and `nuAI2thetaAIinv_lambda` return a `vector` and `matrix`, respectively, holding the sampling (co)variances of the model (co)variance parameters both on the `theta` scale. These are elements of the inverse Average Information matrix.

Author(s)

<matthewwolak@gmail.com>

Examples

```
# User-specified starting parameters
thetaOut <- start2theta(Gstart = list(matrix(1), matrix(2)),
  Rstart = matrix(3))
## convert to a vector and then back into a matrix list
thetav <- matlist2vech(thetaOut$theta)
theta <- vech2matlist(thetav, attr(thetav, "skel"))
  identical(thetaOut$theta, theta)  #<-- should be TRUE
# lambda parameterization transformation
nu <- theta2nu_lambda(theta, thetaOut$thetaG, thetaOut$thetaR)
# back-transform from (lambda scale) nu to theta
## For example, when the sigma2e estimate=0.5
theta2 <- nu2theta_lambda(nu, sigma2e = 0.5, thetaOut$thetaG, thetaOut$thetaR)
```

`deltaSE` *Delta Method to Calculate Standard Errors for Functions of (Co)variances.*

Description

Calculates the standard error for results of simple mathematical functions of (co)variance parameters using the delta method.

Usage

```
deltaSE(calc, object, scale = c("theta", "nu"))

## Default S3 method:
deltaSE(calc, object, scale = c("theta", "nu"))

## S3 method for class 'formula'
```

```
deltaSE(calc, object, scale = c("theta", "nu"))

## S3 method for class 'list'
deltaSE(calc, object, scale = c("theta", "nu"))
```

Arguments

calc	A character expression, formula, or list (of formula or expression) expressing the mathematical function of (co)variance component for which to calculate standard errors.
object	A fitted model object of class 'gremlin'.
scale	A character indicating whether to calculate the function and standard error on the original data scale ("theta") or on the underlying scale to which (co)variance components are transformed for the model fitting calculations ("nu"). Defaults to "theta" if not specified.

Details

The delta method (e.g., Lynch and Walsh 1998, Appendix 1; Ver Hoef 2012) uses a Taylor series expansion to approximate the moments of a function of parameters. Here, a second-order Taylor series expansion is implemented to approximate the standard error for a function of (co)variance parameters. Partial first derivatives of the function are calculated by algorithmic differentiation with deriv.

Though deltaSE can calculate standard errors for non-linear functions of (co)variance parameters from a fitted gremlin model, it is limited to non-linear functions constructed by mathematical operations such as the arithmetic operators +, -, *, / and ^, and single-variable functions such as exp and log. See deriv for more information.

Value

A data.frame containing the "Estimate" and "Std. Error" for the mathematical function(s) of (co)variance components.

Methods (by class)

- default: Default method
- formula: Formula method
- list: List method

Author(s)

<matthewwolak@gmail.com>

References

Lynch, M. and B. Walsh 1998. Genetics and Analysis of Quantitative Traits. Sinauer Associates, Inc., Sunderland, MA, USA.

Ver Hoef, J.M. 2012. Who invented the delta method? The American Statistician 66:124-127. DOI: 10.1080/00031305.2012.687494

See Also

deriv

Examples

```
# Calculate the sum of the variance components
  grS <- gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
  deltaSE(Vsum ~ V1 + V2, grS)
  deltaSE("V1 + V2", grS)  #<-- alternative

# Calculate standard deviations (with standard errors) from variances
  ## Uses a `list` as the first (`calc`) argument
  ### All 3 below: different formats to calculate the same values
  deltaSE(list(SD1 ~ sqrt(V1), SDresid ~ sqrt(V2)), grS)  #<-- formulas
  deltaSE(list(SD1 ~ sqrt(G.sire), SDresid ~ sqrt(ResVar1)), grS)
  deltaSE(list("sqrt(V1)", "sqrt(V2)"), grS)  #<-- list of character expressions

# Additive Genetic Variance calculated from observed Sire Variance
  ## First simulate Full-sib data
  set.seed(359)
  noff <- 5     #<-- number of offspring in each full-sib family
  ns <- 100     #<-- number of sires/full-sib families
  VA <- 1       #<-- additive genetic variance
  VR <- 1       #<-- residual variance
  datFS <- data.frame(id = paste0("o", seq(ns*noff)),
    sire = rep(paste0("s", seq(ns)), each = noff))
  ## simulate mid-parent breeding value (i.e., average of sire and dam BV)
  ### mid-parent breeding value = 0.5 BV_sire + 0.5 BV_dam
  #### var(mid-parent BV) = 0.25 var(BV_sire) + 0.25 var(BV_dam) = 0.5 var(BV)
  datFS$midParBV <- rep(rnorm(ns, 0, sqrt(0.5*VA)), each = noff)
  ## add to this a Mendelian sampling deviation to get each offspring BV
  datFS$BV <- rnorm(nrow(datFS), datFS$midParBV, sqrt(0.5*VA))
  datFS$r <- rnorm(nrow(datFS), 0, sqrt(VR))  #<-- residual deviation
  datFS$pheno <- rowSums(datFS[, c("BV", "r")])
  # Analyze with a sire model
  grFS <- gremlin(pheno ~ 1, random = ~ sire, data = datFS)
  # calculate VA as 2 times the full-sib/sire variance
  deltaSE(VAest ~ 2*V1, grFS)
  # compare to expected value and simulated value
  VA  #<-- expected
  var(datFS$BV)  #<-- simulated (includes Monte Carlo error)

# Example with `deltaSE(..., scale = "nu")
## use to demonstrate alternative way to do same calculation of inverse
## Average Information matrix of theta scale parameters when lambda = TRUE
### what is done inside gremlin::nuVar2thetaVar_lambda
  dOut <- deltaSE(thetaV1 ~ V1*V2, grS, "nu")  #<-- V2 is sigma2e
  aiFnOut <- nuVar2thetaVar_lambda(grS)[1]  #<-- variance (do sqrt below)
  stopifnot(abs(dOut[, "Std. Error"] - sqrt(aiFnOut)) < 1e-10)
```

fixef.gremlin *Fixed Effect Estimates of* class *'gremlin'*

Description

Extracts the fixed effect estimates from a model of class 'gremlin'.

Usage

```
## S3 method for class 'gremlin'
fixef(object, add.dropped = FALSE, ...)
```

Arguments

object	An object of class 'gremlin'.
add.dropped	A logical value indicating whether fixed effects dropped by gremlin, due to rank deficiencies in the fixed effect design matrix, should be included with NA values.
...	Additional arguments.

Value

A numeric vector of fixed effect estimates.

Author(s)

<matthewwolak@gmail.com>

Examples

```
fixef(gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11))
```

gremlin *Mixed-effect modeling functions.*

Description

Fit and setup functions for linear mixed-effect model (Gaussian responses).

Usage

```
gremlin(
  formula,
  random = NULL,
  rcov = ~units,
  data = NULL,
  ginverse = NULL,
  Gstart = NULL,
  Rstart = NULL,
  Bp = NULL,
  Gcon = NULL,
  Rcon = NULL,
  maxit = 20,
  algit = NULL,
  vit = 1,
  v = 1,
  control = gremlinControl(),
  ...
)

gremlinR(
  formula,
  random = NULL,
  rcov = ~units,
  data = NULL,
  ginverse = NULL,
  Gstart = NULL,
  Rstart = NULL,
  Bp = NULL,
  Gcon = NULL,
  Rcon = NULL,
  maxit = 20,
  algit = NULL,
  vit = 1,
  v = 1,
  control = gremlinControl(),
  ...
)

## S3 method for class 'gremlin'
getCall(x, ...)

## S3 method for class 'gremlin'
update(object, ...)

gremlinSetup(
  formula,
  random = NULL,
```

```
  rcov = ~units,
  data = NULL,
  ginverse = NULL,
  Gstart = NULL,
  Rstart = NULL,
  Bp = NULL,
  Gcon = NULL,
  Rcon = NULL,
  maxit = 20,
  algit = NULL,
  vit = 1,
  v = 1,
  control = gremlinControl(),
  ...
)

## S3 method for class 'gremlin'
is(x)

## S3 method for class 'grMod'
is(x)

mkModMats(
  formula,
  random = NULL,
  rcov = ~units,
  data = NULL,
  subset = NULL,
  ginverse = NULL,
  na.action = na.pass,
  offset = NULL,
  contrasts = NULL,
  Xsparse = TRUE,
  ...
)
```

Arguments

formula	A formula for the response variable(s) and fixed effects.
random	A formula for the random effects.
rcov	A formula for the residual covariance structure.
data	A data.frame in which to look for the terms in formula, random, and rcov.
ginverse	A list of (preferably sparse) inverse matrices that are proportional to the covariance structure of the random effects. The name of each element in the list should match a column in data that is associated with a random term. All levels of the random term should appear as rownames for the matrices.
Gstart	A list of matrices with starting (co)variance values for the G-structure or random effects terms.

Rstart	A list of matrices with starting (co)variance values for the R-structure or residual terms.
Bp	A prior specification for fixed effects.
Gcon, Rcon	A list of matrices with constraint codes for the G-structure/random effects or R-structure/residual effects terms, respectively. Must be a character of either "F" for fixed, "P" for positive, or "U" for unbounded.
maxit	An integer specifying the maximum number of likelihood iterations.
algit	A character vector of length 1 or more or an expression to be evaluated that specifies the algorithm to use for proposing (co)variances in the next likelihood iteration.
vit	An integer value specifying the verbosity of screen output on each iteration. A value of zero gives no iteration specific output and larger values increase the amount of information printed on the screen.
v	An integer value specifying the verbosity of screen output regarding the model fitting process. A value of zero gives no details and larger values increase the amount of information printed on the screen.
control	A list returned by gremlinControl containing specific named values from that function. See gremlinControl.
...	Additional arguments to be passed to control the model fitting.
x, object	An object of class gremlin.
subset	An expression for the subset of data to use.
na.action	What to do with NAs.
offset	Should an offset be specified.
contrasts	Specify the type of contrasts for the fixed effects.
Xsparse	Should sparse matrices be used for the fixed effects design matrix.

Value

A list containing an object of class grMod and, if a model was fit (gremlin or gremlinR) then an object containing details of the REML iterations (object itMat). An object of class grMod contains:

call The model call.

modMats A list of the model matrices used to construct the mixed model equations.

y The response vector.

ny The number of responses.

ncy The number of columns of the original response.

X The fixed effects design matrix.

nb The number of columns in X.

Zr The residual design matrix.

Zg A list of the design matrices for each random term.

nG The number of parameters in the G structure.

listGeninv A list of generalized inverse matrices.

logDetG The log-determinants of the generalized inverse matrices - necessary to calculate the log-likelihood.

rfxIncContrib2loglik A `numeric` value containing the sum of the log determinants of the random effects that do not change between log-likelihood iterations (i.e., the part of the log determinants of (co)variance matrices to be estimated that have been factored out).

ndgeninv A `logical` indicating which terms in the random formula have generalized inverses associated with them (non-diagonal matrices in the Kronecker product.

dimsZg, nminffx, rfxlvls, nminfrfx `Numeric` vectors or scalars describing the numbers of random effects or some function of random and fixed effects.

conv, bounds (Co)variance component constraints and boundaries of the allowable parameter space for each component.

thetav A `vector` of the (co)variance parameters to be estimated by REML withe the attribute "skel" giving the skeleton to recreate a list of `matrices` from this vector.

thetaG, thetaR `Vectors` indexing the random and residual (co)variances, respectively, in a list of (co)variance matrices (i.e., `theta`).

nu A `list` of transformed (co)variance matrices to be fit by REML. If a residual variance has been factored out of the mixed model equations, `nu` contains the 'lambda' parameterization with expresses the (co)variance components as ratios of variance parameters with the residual variance. The 'nu' scale (co)variances are the ones actually fit by REML.

sigma2e The estimate of the factored out residual variance from the mixed model equations (i.e., the 'lambda' scale) σ_e^2.

p An `integer` for the total number of (co)variances to be estimated.

lambda A `logical` indicating whether the 'lambda' scale parameterization has been used.

uni A `logical` to indicate if the model is univariate or not.

W, tWW, RHS, Bpinv Sparse matrices of class `Matrix` that form the mixed model equations and do not change between iterations of REML. These are the column bound 'X' and 'Z' design matrices for fixed and random effects, the cross-product of `W`, the Right-Hand Side of the mixed model equations, and the inverse of the fixed effect prior matrix (zeroes on the diagonal if no priors have been specified). Note, these may be `NULL` if `lambda=FALSE`, because the `NULL` objects are not used or do change between REML iterations.

sLc A `Matrix` containing the symbolic Cholesky factorization of the coefficient matrix of the Mixed Model Equations.

sln A one column `matrix` of solutions in the mixed model equations.

Cinv_ii A one column `matrix` of variances for the solutions to the mixed model equations. These are obtained from the diagonal of the inverse Coefficient matrix in the mixed model equations. If lambda is `TRUE` then these are on the lambda scale and must be multiplied by `sigma2e` to be converted to the original data scale.

r A one column `matrix` of residual deviations, response minus the values expected based on the solutions, corresponding to the order in `modMats$y`.

AI A `matrix` of values containing the Average Information matrix, or second partial derivatives of the likelihood with respect to the transformed (co)variance components ('nu'). The inverse of this matrix gives the sampling (co)variances of these transformed (co)variance components.

dLdnu A single column `matrix` of first derivatives of the transformed (co)variance parameters ('nu') with respect to the log-Likelihood.

maxit See the parameter described above.

algit A `character` vector of REML algorithms to use in each iteration.

vit See the parameter described above.

v See the parameter described above.

cctol A `numeric` vector of convergence criteria thresholds. See `gremlinControl` for more details.

ezero, einf `numeric` values for the effective numbers to use as "zero" and maximum negative or positive numbers. Values less than `ezero` are treated as zero and fixed to this value. Values less than `-1*einf` or greater than `einf` are restricted to these values. See `gremlinControl` for more details.

step A `numeric` value indicating the step-reduction to use. See `gremlinControl` for more details.

itMat A `matrix` of details about each iteration. Rows indicate each REML iteration (rownames reflect the REML algorithm used) and columns contain:

nu, theta (Co)variance parameters.

sigma2e See 'sigma2e' described above.

tyPy, logDetC Estimates for two these two components of the log of the REML likelihoods. These are obtained from Cholesky factorization of the coefficient matrix of the mixed model equations.

loglik The REML log-likelihood.

itTime Time elapsed for each REML iteration.

Functions

- `mkModMats`: Generates model matrices.

Author(s)

<matthewwolak@gmail.com>

Examples

```
grSire <- gremlin(WWG11 ~ sex, random = ~ sire, data = Mrode11)
# Now drop sire random effects and use the `anova` method to compare models
grLM <- update(grSire, random = ~ 1)  #<-- use `~1` to drop all random effects
  anova(grSire, grLM)

# Modular functions
## get model matrices for a mixed model
mM11 <- mkModMats(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)

## setup model, but do not evaluate the log-likelihood
grSetup <- gremlinSetup(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
## maximize the restricted maximum likelihood
grOut <- remlIt(grSetup)
summary(grOut)

## Not run:
  # Following the example from Mrode 2005, chapter 11.
  library(nadiv)  #<-- to construct inverse of the numerator relatedness matrix
```

```
  Ainv <- makeAinv(Mrode11[, 1:3])$Ainv

  gr11lmm <- gremlin(WWG11 ~ sex - 1,
random = ~ calf,
data = Mrode11,
ginverse = list(calf = Ainv),
Gstart = matrix(0.2), Rstart = matrix(0.4),  #<-- specify starting values
maxit = 15,    #<-- maximum iterations
     v = 2, vit = 1,  #<-- moderate screen output (`v`) every iteration (`vit`)
     algit = "AI")  #<-- only use Average Information algorithm iterations
  summary(gr11lmm)

  # Compare the model to a Linear Model with no random effects
  ## Use `update()` to update the model
  gr11lm <- update(gr11lmm, random = ~ 1)  #<-- `~1`=drop all random effects
  summary(gr11lm)

  # Do analysis of variance between the two models
  ## See AIC or evaluate likelihood ratio against a Chi-squared distribution
  anova(gr11lm, gr11lmm)

## End(Not run)
```

`gremlinControl` *Advanced Options for Mixed-effect modeling functions.*

Description

Change default settings for gremlin models.

Usage

```
gremlinControl(
  cctol = c(5e-04, 1e-08, 0.001, NULL),
  ezero = 1e-08,
  einf = 1e+30,
  step = 0.3,
  lambda = TRUE,
  algorithm = NULL,
  algArgs = list()
)
```

Arguments

`cctol`	Convergence criteria tolerances (Meyer 2007, 2019).
`ezero`	Effective zero to be used, values less than this number are treated as zero and fixed to this value.
`einf`	Effective infinite value to be used, values are limited to a to this variable as a maximum.

step	A numeric value for scaling the proposed parameter updates.
lambda	A logical indicating whether a residual variance should be factored out of the mixed model equations.
algorithm	A character naming the function to use to decide subsequent parameters in the REML iterations.
algArgs	A list of function arguments to be given to functions named in the algorithm argument.

Value

A list of class gremlinControl to be used by gremlinSetup and later functions when fitting the model.

Author(s)

<matthewwolak@gmail.com>

References

Meyer, K. 2007. WOMBAT - a tool for mixed model analyses in quantitative genetics by restricted maximum likelihood (REML). Journal of Zhejiang University SCIENCE B 8(11):815-821.

Meyer, K. 2019. WOMBAT A program for mixed model analyses by restricted maximum likelihood. User Notes. 27 September 2019.

Examples

```
str(gremlinControl())
```

---

logLik.gremlin *Methods to extract log-likelihood and information criterion of a gremlin model.*

Description

Extracts the log-likelihood or AIC from a gremlin model fit.

Usage

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

npar.gremlin(object)

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

Arguments

object	An object of class 'gremlin'.
...	Additional arguments.
k	A numeric value for the penalty per parameter. Default is 2, as in classic AIC.
fxdDf	A logical indicating whether to penalize according to the number of fixed effect parameters. Since only models fit by REML can be compared, these must always be the same and so become a constant. Hence, the default is FALSE.

Details

Function npar.gremlin returns an object with attributes n.fxd and n.bndry which give additional information about the parameters estimated and contributing to the overall df of the model. n.fxd returns the total number of parameters (No. fixed effects + No. (co)variance components) minus the number of parameters constrained to a certain value. Thus, n.fxd represents the number of parameters that can vary and, as a consequence, affect the log-likelihood.

The attribute n.bndry reports the number of parameters that were restrained to stay inside the boundaries of allowable parameter space (e.g., a variance that was not allowed to be negative).

Value

numeric values for the log-likelihood, the number of parameters estimated by the model (sum of fixed effects and random effect (co)variance components), and Akaike's Information Criterion.

Author(s)

<matthewwolak@gmail.com>

Examples

```
grS <- gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
logLik(grS)
AIC(grS)
```

Mrode11 *Weight gain data.*

Description

Data from chapter 11 in Mrode 2005. The variables are as follows:

Usage

```
Mrode11
```

Format

An object of class data.frame with 5 rows and 5 columns.

Details

- calf. a factor with levels 4 5 6 7 8
- dam. a factor with levels 2 5 6
- sire. a factor with levels 1 3 4
- sex. a factor with levels male female
- WWG11. a numeric vector

Source

Mrode, R.A. 2005. Linear Models for the Prediction of Animal Breeding Values, 2nd ed. CABI Publishing, Cambridge, MA, USA.

Examples

```
data(Mrode11)
```

---

nobs.gremlin *Number of observations in data from gremlin model fit objects*

Description

Extract the number of 'observations' in a gremlin model fit.

Usage

```
## S3 method for class 'gremlin'
nobs(object, use.fallback = FALSE, ...)
```

Arguments

object	An object of class 'gremlin'.
use.fallback	logical: should fallback methods be used to try to guess the value? Included for compatibility.
...	Further arguments to be passed to the methods.

Value

A single number, usually an integer, but can be NA.

Author(s)

<matthewwolak@gmail.com>

Examples

```
grS <- gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
nobs(grS)
```

reml *REML optimization algorithms for mixed-effect models.*

Description

Evaluate the REML likelihood and algorithms for iterating to find maximum REML estimates.

Usage

```
reml(
  nu,
  skel,
  thetaG,
  sLc,
  modMats,
  W,
  Bpinv,
  nminffx,
  nminfrfx,
  rfxlvls,
  rfxIncContrib2loglik,
  thetaR = NULL,
  tWW = NULL,
  RHS = NULL
)

em(nuvin, thetaG, thetaR, conv, modMats, nminffx, sLc, ndgeninv, sln, r)

ai(nuvin, skel, thetaG, modMats, W, sLc, sln, r, thetaR = NULL, sigma2e = NULL)

gradFun(
  nuvin,
  thetaG,
  modMats,
  Cinv,
  sln,
  sigma2e = NULL,
  r = NULL,
  nminfrfx = NULL
)
```

Arguments

nu, nuvin	A `list` or `vector` of (co)variance parameters to estimate on the transformed, or nu, scale.
skel	A skeleton for reconstructing the list of (co)variance parameters.

`thetaG`, `thetaR` `Integer` vectors indexing the G-structure or R-structure of the (co)variance parameters.

`sLc` A sparse `Matrix` containing the symbolic Cholesky factorization of the coefficient matrix of the Mixed Model Equations.

`modMats` A `list` of the model matrices used to construct the mixed model equations.

`W`, `tWW` A sparse `Matrix` containing the design matrices for the fixed and random effects (`W`) and the cross-product of this (`tWW`).

`Bpinv` A matrix inverse of the matrix containing the prior specification for fixed effects.

`nminffx`, `nminfrfx`, `rfxlvls`
`Integers` specifying: (1) the difference between the number of observations and fixed effects (of the full rank fixed effects design matrix (X), (2) `nminffx` minus the total number of random effects, and (3) a `vector` of levels for each term in the random effects.

`rfxIncContrib2loglik`
A `numeric` indicating the sum of constraint contributions to the log-likelihood across all terms in the random effects that have non-diagonal generalized inverse matrices (`ginverse`).

`RHS` A sparse `Matrix` containing the Right-Hand Side to the Mixed Model Equations.

`conv` A `character` vector of (co)variance parameter constraints.

`ndgeninv` A `logical` vector indicating if each random term is associated with a generalized inverse (`ginverse`).

`sln`, `r` Sparse `Matrices` containing the solutions or residuals of the Mixed Model Equations.

`sigma2e` A `numeric` value for the residual variance estimate when it has been factored out of the Coefficient matrix of the Mixed Model Equations, thus converting the (co)variance components to ratios (represented by the variable lambda).

`Cinv` A sparse `Matrix` containing the inverse of the Coefficient matrix to the Mixed Model Equations.

Value

A `list` or `matrix` containing any of the previous parameters described above, or the following that are in addition to or instead of parameters above:

loglik The REML log-likelihood.

tyPy,logDetC Components of the REML log-likelihood derived from the Cholesky factor of the Coefficient matrix to the Mixed Model Equations.

Cinv_ii A vector containing the diagonal elements of the inverse of the Coefficient matrix to the Mixed Model Equations (i.e., the diagonal entries of `Cinv`).

AI A `matrix` of values containing the Average Information matrix, or second partial derivatives of the likelihood with respect to the transformed (co)variance components (nu). The inverse of this matrix gives the sampling variances of these transformed (co)variance components.

dLdnu A single column `matrix` of first derivatives of the transformed (co)variance parameters (nu) with respect to the log-Likelihood.

Author(s)

<matthewwolak@gmail.com>

`remlIt` *Mixed-effect model Restricted Maximum Likelihood (REML) iterations.*

Description

Conduct REML iterations to estimate (co)variance parameters of a linear mixed-effect model (Gaussian responses).

Usage

```
remlIt(grMod, ...)

## Default S3 method:
remlIt(grMod, ...)

## S3 method for class 'gremlinR'
remlIt(grMod, ...)
```

Arguments

`grMod`	A gremlin model of class `grMod`. See `gremlin` or `gremlinSetup` for the functions constructing an object of class `grMod`.
`...`	Additional arguments to be passed to control the model fitting.

Value

A `list` containing an object of class `grMod` and `matrix` containing details of the REML iterations (object `itMat`). See `gremlin` for descriptions of `grMod` and `itMat` objects.

Methods (by class)

- `default`: Default method
- `gremlinR`: gremlinR method

Author(s)

<matthewwolak@gmail.com>

Examples

```
grSsetp <- gremlinSetup(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
grS <- remlIt(grSsetp)
```

residuals.gremlin *Residuals of* class *'gremlin'*

Description

Residuals of class 'gremlin'.

Usage

```
## S3 method for class 'gremlin'
residuals(object, type = "response", scaled = FALSE, ...)
```

Arguments

object	An object of class 'gremlin'.
type	The type of residuals which should be returned. Only implement "response" currently. Can be abbreviated.
scaled	Logical value indicating whether to scale residuals by the residual standard deviation.
...	Additional arguments.

Value

A numeric vector of residuals.

Author(s)

<matthewwolak@gmail.com>

Examples

```
grS <- gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
residuals(grS)
```

runtime *Time to execute the gremlin model*

Description

Extract the length of time to fit the model.

Usage

```
runtime(object, ...)
```

Arguments

object	An object of class 'gremlin'.
...	Further arguments to be passed to the methods.

Value

A numeric of class 'difftime' with an attribute of units (e.g., seconds or minutes).

Author(s)

<matthewwolak@gmail.com>

---

summary.gremlin *Gremlin model summary.*

Description

Summarize and print results of linear mixed model fitted with gremlin.

Usage

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

## S3 method for class 'summary.gremlin'
print(x, digits = max(3, getOption("digits") - 3), ...)
```

Arguments

object, x	An object of class 'gremlin' or 'summary.gremlin'.
...	Additional arguments to be passed to control the output.
digits	An integer used for number formatting with 'signif()'.

Value

A list of class summary.gremlin or a printed value to the screen with no return values.

logLik Model log-likelihood.

formulae Function call and model fixed, random, and residual formulae.

runtime A numeric of class 'difftime' containing the length of time to run the model. See how this is handled in update.gremlin.

lambda A logical indicating if the model was transformed to the variance ratio, or lambda scale.

residQuants A named vector listing summary output for the model residuals.

varcompSummary Table of variance components and approximate standard errors (calculated from the inverse of the average information matrix). If a (co)variance component is fixed or at the boundary of its parameter space then an NA is returned for the standard error and a column with constraint types is added to the table. Alternative methods (e.g., profile likelihood CIs) should be pursued for obtaining uncertainties associated with fixed or boundary parameters.

varcompSampCor A matrix containing the sampling correlations of the (co)variance components. Note this is on the underlying nu scale that the model is fitting.

coefficients Table of fixed effects and standard errors (calculated from the corresponding diagonal elements of the inverse of the coefficient matrix, transformed where necessary).

Author(s)

<matthewwolak@gmail.com>

See Also

gremlin

Examples

```
grS <- gremlin(WWG11 ~ sex - 1, random = ~ sire, data = Mrode11)
summary(grS)
```

---

tr *Matrix trace methods.*

Description

Methods to efficiently calculate a matrix trace depending on the class of matrix.

Usage

```
tr(X, ...)

## Default S3 method:
tr(X, ...)

## S3 method for class 'dgCMatrix'
tr(X, ...)

## S3 method for class 'dsCMatrix'
tr(X, ...)
```

Arguments

X	A matrix.
...	Additional arguments.

Value

A numeric value for the sum of the diagonal elements.

Methods (by class)

- default: Default method
- dgCMatrix: Method for matrix X of class Matrix:::dgCMatrix
- dsCMatrix: Method for matrix X of class Matrix:::dsCMatrix

Author(s)

<matthewwolak@gmail.com>

Index