Package 'spaMM'

September 5, 2016

Type Package

Title Mixed Models, Particularly Spatial GLMMs

Encoding UTF-8

Version 1.10.0

Date 2016-09-04

Maintainer François Rousset <francois.rousset@umontpellier.fr>

Imports methods, stats, graphics, Matrix, MASS, lpSolveAPI (>= 5.5.0.14), proxy, geometry (>= 0.3.6), Rcpp (>= 0.11.0), nlme, mvtnorm, nloptr

LinkingTo Rcpp, RcppEigen

Suggests maps, testthat, lme4, rsae, ff, rasterVis, rgdal, rcdd

Depends R (>= 3.1.0)

NeedsCompilation yes

Description
Inference in mixed models, including GLMMs with spatial correlations and models with non-Gaussian random effects (e.g., Beta Binomial, or negative-binomial mixed models). Heteroscedasticity can further be fitted by a linear model. The algorithms are currently various Laplace approximations methods for ML or REML, in particular h-likelihood and penalized-likelihood methods.

License CeCILL-2

URL http://www.r-project.org,
http://kimura.univ-montp2.fr/~rousset/spaMM.htm

Author François Rousset [aut, cre, cph],
Jean-Baptiste Ferdy [aut, cph],
Alexandre Courtiol [ctb],
Dirk Eddelbuettel [ctb] (ziggurat rnorm sources),
GSL authors [ctb] (src/gsl_bessel.*)

Repository CRAN

Date/Publication 2016-09-05 04:17:26

R topics documented:

adjacency — *Fitting autoregressive models*

Description

This describes the autoregressive models (AR) considered by spaMM and how to fit them. This documentation is mainly a stub, as more implementation of some these models (in particular, of simultaneous AR, i.e., SAR) is to be expected in the future. The only model currently implemented is the adjacency model (a conditional AR, i.e., CAR).

Details

The most widely applicable fitting strategy in spaMM is to fit possibly complex spatial models by numerical maximization over correlation parameters. However, more efficient algorithms are available for CAR and SAR models, and have been widely used in particular in the econometric literature (e.g., LeSage and Pace 2009). Currently such an algorithm is implemented only for one CAR model, the adjacency model. The taxonomy of AR models is not well settled, but adjacency should retain its current meaning as a specific CAR model in future versions of spaMM.

In the adjacency model, the covariance matrix of random effects **u** can be described as $\lambda(\mathbf{I}-\rho\,\mathbf{W})^{-1}$ where **W** is the (symmetric) adjacency matrix. If you use corrHLfit, the default fitting method for the adjacency model is numerical maximization of the likelihood (or restricted likelihood) as function of the correlation parameter ρ. However, a call to HLCor fits more efficiently the variance and correlation parameters of random effects (see examples). The ML fits by both methods should be practically equivalent. The REML fits should slightly differ from each other, due to the fact that the REML approximation for GLMMs does not maximize a single likelihood function. Finally, a call to corrHLfit with the additional argument init.HLfit=list(rho=0) should be equivalent in speed and result to the HLCor call.

The efficient method uses the spectral decomposition **W=VDV'** where **D** is a diagonal matrix of eigenvalues d_i, the covariance of **V'u** is $\lambda(\mathbf{I}-\rho\,\mathbf{D})^{-1}$, which is a diagonal matrix with elements λ_i=λ/$(1-\rho d_i)$. Hence $1/\lambda_i$ is in the linear predictor form α+βd_i This can be used to fit λ and ρ efficiently. If the efficient fitting method is used, the results are reported as the coefficients α ((Intercept)) and β (adjd) of the predictor for $1/\lambda_i$, in addition to the resulting values of ρ and of the common λ factor.

References

LeSage, J., Pace, R.K. (2009) Introduction to Spatial Econometrics. Chapman & Hall/CRC.

Examples

```
data(scotlip)
# CAR by Laplace with 'outer' estimation of rho
if (spaMM.getOption("example_maxtime")>5) {
  corrHLfit(cases~I(prop.ag/10) +adjacency(1|gridcode)+offset(log(scotlip$expec)),
          adjMatrix=Nmatrix,family=poisson(),data=scotlip,HLmethod="ML") ## ~ 5 s.
}
```

```
# CAR by Laplace with 'inner' estimation of rho
HLCor(cases~I(prop.ag/10) +adjacency(1|gridcode)+offset(log(scotlip$expec)),
          adjMatrix=Nmatrix,family=poisson(),data=scotlip,HLmethod="ML") ## < 1 s.
```

---

arabidopsis *Arabidopsis genetic and climatic data*

Description

For 948 “accessions” from European Arabidopsis thaliana populations, this data set merges the genotypic information at four single nucleotide polmorphisms (SNP) putatively involved in adaptation to climate (Fournier-Level et al, 2011, Table 1), with 13 climatic variables from Hancock et al. (2011).

Usage

```
data(arabidopsis)
```

Format

The data frame includes 948 observations on the following variables:

pos1046738, pos5510910, pos6235221, pos8132698 Genotypes at four SNP loci

LAT latitude

LONG longitude

seasonal, tempWarmest, tempColdest, preciWettest, preciDriest, preciCV, PAR_SPRING,

growingL, conseqCold, conseqFrFree, RelHumidSp, dayLSp, aridity Thirteen climatic variables. See Hancock et al. (2011) for details about these variables.

Details

The response is binary so “HLmethod="PQL/L"” seems warranted (see Rousset and Ferdy, 2014).

Source

The data were retrieved from `http://bergelson.uchicago.edu/regmap-data/climate-genome-scan` on 22 February 2013 (they may no longer be available from there).

References

Fournier-Level A, Korte A., Cooper M. D., Nordborg M., Schmitt J., Wilczek AM (2011). A map of local adaptation in Arabidopsis thaliana. Science 334: 86-89.

Hancock, A. M., Brachi, B., Faure, N., Horton, M. W., Jarymowycz, L. B., Sperone, F. G., Toomajian, C., Roux, F., and Bergelson, J. 2011. Adaptation to climate across the Arabidopsis thaliana genome, Science 334: 83-86.

Rousset F., Ferdy, J.-B. (2014) Testing environmental and genetic effects in the presence of spatial autocorrelation. Ecography, 37: 781-790. http://dx.doi.org/10.1111/ecog.00566

Examples

```
data(arabidopsis)
if (spaMM.getOption("example_maxtime")>3) {
  HLCor(cbind(pos1046738,1-pos1046738)~seasonal+Matern(1|LAT+LONG),
            ranPars=list(rho=0.129,lambda=4.28,nu=0.291),
            family=binomial(),HLmethod="PQL/L",data=arabidopsis)
}
## the above ranPars are deduced from an hour(s?)-long analysis:
## Not run:
verylong <- corrHLfit(cbind(pos1046738,1-pos1046738)~seasonal+Matern(1|LAT+LONG),
              family=binomial(),HLmethod="PQL/L",data=arabidopsis)
summary(verylong)

## End(Not run)
```

---

blackcap — *Genetic polymorphism in relation to migration in the blackcap*

Description

This data set is extracted from a study of genetic polymorphisms potentially associated to migration behaviour in the blackcap (Sylvia atricapilla). Across different populations in Europe and Africa, the average migration behaviour was found to correlate with average allele size (dependent on the number of repeats of a small DNA motif) at the locus ADCYAP1, encoding a neuropeptide.

Usage

```
data(blackcap)
```

Format

The data frame includes 14 observations on the following variables:

latitude latitude, indeed.

longitude longitude, indeed.

migStatus migration status as determined by Mueller et al, from 0 (resident populations) to 2.5 (long-distance migratory populations)

means Mean allele sizes in each population

pos Numerical index for the populations

Details

Migration status was coded as : pure resident populations as '0', resident populations with some migratory restlessness as '0.5', partial migratory populations as '1', completely migratory populations migrating short-distances as '1.5', intermediate-distance migratory populations as '2' and distinct long-distance migratory populations as '2.5'.

Source

Data from Mueller et al. (2011), including supplementary material retrieved from http://rspb.royalsocietypublishing.org/content/suppl/2011/02/11/rspb.2010.2567.DC1.

References

Mueller, J. C., Pulido, F., and Kempenaers, B. 2011. Identification of a gene associated with avian migratory behaviour, Proc. Roy. Soc. (Lond.) B 278, 2848-2856.

Examples

```
## see 'corrHLfit' and 'fixedLRT' for examples involving these data
```

COMPoisson *Conway-Maxwell-Poisson (COM-Poisson) GLM family*

Description

The COM-Poisson family is a generalization of the Poisson family which can describe over-dispersed as well as under-dispersed count data. It is indexed by a parameter $\nu_C MP$ that quantifies such dispersion. The COM-Poisson family with given $\nu_C MP$ is here implemented as a `family` object, so that it can be fitted by `glm`, and further used to model conditional responses in mixed models fitted by this package's functions (see Examples). $\nu_C MP$ is distinct from the dispersion parameter $\nu = 1/\phi$ considered elsewhere in this package and in the GLM literature, as ν affects in a more specific way the log-likelihood. The "canonical link" $\theta(\mu)$ between the canonical GLM parameter θ and the expectation μ of the response do not have a known expression in terms of elementary functions. The link inverse is $\mu(\theta) = \sum_(i = 0)^\infty \lambda^i/(i!)^\nu$ for $\lambda = e^\theta$ (hence the link is here nicknamed `"loglambda"`).

Usage

```
COMPoisson(nu =  stop("COMPoisson's 'nu' must be specified"),
           link = "loglambda")
```

Arguments

`link`	GLM link function. Cannot be modified.
`nu`	Under-dispersion parameter $\nu_C MP$. `COMPoisson(nu=1)` is the Poisson family. For $\nu > 1$, the distribution is under-dispersed. The limit as $\nu -> 0$ is the Bernoulli distribution with expectation $\lambda/(1+\lambda)$. `COMPoisson(nu=0)` would describe a geometric distribution with parameter λ. However the code may fail to fit distributions with ν approaching 0 because this requires various sums that have to be truncated and cannot be easily approximated. $\nu = 0$ may more easily be fitted thanks to special code for this case.

Details

Fitting will be very slow when $\nu_C MP$ approaches zero, i.e. for strong overdispersion of (conditional) response. It may be faster if the value of the control parameter `COMP_maxn` is reduced by using `spaMM.options()`, but this will reduce the accuracy of the results for high overdispersion. This trade-off occurs because the link inverse function, as shown in Description, involves an infinite summation. This sum can be easily approximated for large $\nu_C MP$ but not when $\nu_C MP$ approaches zero. By default, `spaMM` truncates the sum at `spaMM.getOption(COMP_maxn)`= 10000 terms, which should be more than enough for underdispersed response.

Value

A family object. The 'simulate' member function is not yet implemented.

References

G. Shmueli, T. P. Minka, J. B. Kadane, S. Borle and P. Boatwright (2005) A useful distribution for fitting discrete data: revival of the Conway-Maxwell-Poisson distribution. Appl. Statist. 54: 127-142.

Sellers KF, Shmueli G (2010) A Flexible Regression Model for Count Data. Ann. Appl. Stat. 4: 943–961

Examples

```
## Fitting COMPoisson model with estimated nu parameter:
if (spaMM.getOption("example_maxtime")>7) {
  data(freight) ## example from Sellers & Shmueli, Ann. Appl. Stat. 4: 943961 (2010)
  fitme(broken ~ transfers,family = COMPoisson(),data=freight)
}
## Not run:
  data(freight)
  # Equivalence of poisson() and COMPoisson(nu=1):
  COMPglm <- glm(broken ~ transfers, data=freight, family = poisson())
  coef(COMPglm)
  logLik(COMPglm)
  COMPglm <- glm(broken ~ transfers, data=freight, family = COMPoisson(nu=1))
  coef(COMPglm)
  logLik(COMPglm)
  HLfit(broken ~ transfers, data=freight, family = COMPoisson(nu=1))
  # GLMM with under-dispersed conditional response
 HLfit(broken ~ transfers+(1|id), data=freight, family = COMPoisson(nu=10),HLmethod="ML")

## End(Not run)
```

`confint.HLfit` *Confidence intervals for fixed-effect parameters*

Description

This computes confidence intervals for a given fixed effect parameter, based on the p_v-based approximation of the profile likelihood ratio for this parameter. The profiling is other all other fitted parameters: other fixed effects, as well as variances of random effects and spatial correlations if these were fitted.

Usage

```
## S3 method for class 'HLfit'
confint(object, parm, level=0.95, verbose=TRUE,...)
```

Arguments

`object`	An object of class `HLfit`, such as return object of HLfit, HLCor or corrHLfit calls;
`parm`	The name of a parameter to be fitted, or its position in the the object's `$fixef` vector. Valid names are those of the object's `$fixef`;
`level`	The coverage of the interval;
`verbose`	whether to print the interval or not. As the function returns its more extensive results invisibly, this printing is the only visible output;
`...`	Additional arguments (maybe not used, but conforming to the generic definition of `confint`).

Value

A list including the confidence `interval` for the target parameter, and the fits `lowerfit` and `upperfit` giving the profile fits at the confidence bounds. This is returned invisibly.

Examples

```
if (spaMM.getOption("example_maxtime")>1) {
 data(wafers)
 wfit <- HLfit(y ~X1+(1|batch),family=Gamma(log),data=wafers,HLmethod="ML")
 confint(wfit,"X1")
}
```

`corMatern` *Matern Correlation Structure as a corSpatial object*

Description

This implements the Matérn correlation structure (see `Matern`) for use with `lme` or `glmmPQL`. Usage is as for others `corSpatial` objects such as `corGaus` or `corExp`, except that the Matérn family has an additional parameter. This function was defined for comparing results obtained with `corrHLfit` to those produced by `lme` and `glmmmPQL`. There are problems in fitting (G)LMMs in the latter way, so it is not a recommended practice.

Usage

```
corMatern(value = c(1, 0.5), form = ~1, nugget = FALSE, nuScaled = FALSE,
          metric = c("euclidean", "maximum", "manhattan"), fixed = FALSE)
```

Arguments

`value` An optional vector of parameter values, with serves as initial values or as fixed values depending on the `fixed` argument. It has either two or three elements, depending on the `nugget` argument.

If `nugget` is `FALSE`, `value` should have two elements, corresponding to the "range" and the "smoothness" ν of the Matérn correlation structure. If `value` has zero length, the default is a range of 90% of the minimum distance and a smoothness of 0.5 (exponential correlation). **Warning**: the range parameter used in `corSpatial` objects is the inverse of the scale parameter used in `MaternCorr` and thus they have opposite meaning despite both being denoted ρ elsewhere in this package or in `nlme` literature.

If `nugget` is `TRUE`, meaning that a nugget effect is present, `value` can contain two or three elements, the first two as above, the third being the "nugget effect" (one minus the correlation between two observations taken arbitrarily close together). If `value` has length zero or two, the nugget defaults to 0.1. The range and smoothness must be greater than zero and the nugget must be between zero and one.

`form` (Pasted from corSpatial) a one sided formula of the form `~ S1+...+Sp`, or `~ S1+...+Sp | g`, specifying spatial covariates `S1` through `Sp` and, optionally, a grouping factor `g`. When a grouping factor is present in `form`, the correlation structure is assumed to apply only to observations within the same grouping level; observations with different grouping levels are assumed to be uncorrelated. Defaults to `~ 1`, which corresponds to using the order of the observations in the data as a covariate, and no groups.

`nugget` an optional logical value indicating whether a nugget effect is present. Defaults to `FALSE`.

`nuScaled` If `nuScaled` is set to `TRUE` the "range" parameter ρ is divided by $2\sqrt{\nu}$. With this option and for large values of ν, `corMatern` reproduces the calculation of `corGaus`. Defaults to `FALSE`, in which case the function compares to `corGaus` with range parameter $2(\sqrt{\nu})\rho$ when ν is large.

`metric` (Pasted from corSpatial) an optional character string specifying the distance metric to be used. The currently available options are `"euclidean"` for the root sum-of-squares of distances; `"maximum"` for the maximum difference; and `"manhattan"` for the sum of the absolute differences. Partial matching of arguments is used, so only the first three characters need to be provided. Defaults to `"euclidean"`.

`fixed` an optional logical value indicating whether the coefficients should be allowed to vary in the optimization, or kept fixed at their initial value. Defaults to `FALSE`, in which case the coefficients are allowed to vary.

Details

This function is a constructor for the `corMatern` class, representing a Matérn spatial correlation structure. See `MaternCorr` for details on the Matérn family.

Value

an object of class `corMatern`, also inheriting from class `corSpatial`, representing a Matérn spatial correlation structure.

Note

The R and C code for the methods for `corMatern` objects builds on code for `corSpatial` objects, by D.M. Bates, J.C. Pinheiro and S. DebRoy, in a circa-2012 version of nlme.

References

Mixed-Effects Models in S and S-PLUS, José C. Pinheiro and Douglas M. Bates, Statistics and Computing Series, Springer-Verlag, New York, NY, 2000.

See Also

`glmmPQL`, `lme`

Examples

```
## LMM
data(blackcap)
blackcapD <-cbind(blackcap,dummy=1) ## obscure, isn't it?
## With method= 'ML' in lme, The correlated random effect is described
##  as a correlated residual error and no extra residual variance is fitted:
nlme::lme(fixed = migStatus ~ means, data = blackcapD, random = ~ 1 | dummy,
    correlation = corMatern(form = ~ longitude+latitude  | dummy),
    method = "ML")

## Binomial GLMM
## ~54s. on a laptop
if (spaMM.getOption("example_maxtime")>50) {
 data(Loaloa)
 LoaloaD <-cbind(Loaloa,dummy=1)
 MASS::glmmPQL(fixed =cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI,
        data = LoaloaD, random = ~ 1 | dummy,family=binomial,
        correlation = corMatern(form = ~ longitude+latitude | dummy))
}
```

corrHLfit *Fits a mixed model, typically a spatial GLMM.*

Description

corrHLfit performs the joint estimation of correlation parameters, fixed effect and dispersion parameters.

Usage

```
corrHLfit(formula, data, init.corrHLfit = list(),
          init.HLfit = list(), ranFix = list(), lower = list(),
          upper = list(), trace = list(file = NULL, append = TRUE),
          objective = "p_bv", resid.model = ~1, resid.formula,
          control.dist = list(), control.corrHLfit = list(),
          processed = NULL, family = gaussian(), ...)
```

Arguments

formula	Either a linear model formula (as handled by various fitting functions) or a predictor, i.e. a formula with attributes (see Predictor and examples below). See Details in spaMM for allowed terms in the formula.
data	A data frame containing the variables in the response and the model formula.
init.corrHLfit	An optional list of initial values for correlation and/or dispersion parameters, e.g. list(rho=1,nu=1,lambda=1,phi=1) where rho and nu are parameters of the Matérn family (see Matern), and lambda and phi are dispersion parameters (see Details in spaMM for the meaning of these parameters). All are optional, but giving values for a dispersion parameter changes the ways it is estimated (see Details). rho may be a vector (see make_scaled_dist) and, in that case, it is possible that some or all of its elements are NA, for which corrHLfit substitute automatically determined values.
init.HLfit	See identically named HLfit argument.
ranFix	A list similar to init.corrHLfit, but specifying fixed values of the parameters not estimated.
lower	An optional list of values of parameters specified through init.corrHLfit, used as lower values in calls to optim. See Details for default values.
upper	Same as lower, but upper values.
trace	Not for normal use. If trace is of the form trace=list(file=<filename>,append=F), some trace information is written in the file 'filename'. This file is written over by each new call of corrHLfit unless append=T. Information is written for each HLCor call. It includes the APHLs, and all given fixed random effect parameters in the HLCor call. Information about the variables is printed at the end of the file, but may be slightly inaccurate (some amount of guesswork is expected from users venturing into trace). The arguments of all HLCor calls are saved in RData files (if this is felt inappropriate, then using spaMM.options(TRACE.UNLINK=TRUE) will keep only the latest HLCor call).

`objective` The objective function maximized for estimation of parameters by `optim`. Either `"p_bv"` for restricted likelihood or `"p_v"` for marginal likelihood.

`resid.model, resid.formula`
See identically named `HLfit` arguments.

`control.dist` See `control.dist` in `HLCor`

`control.corrHLfit`
This may be used to provide control to the optimizer. In particular, `control.corrHLfit$optim` should be a list of arguments for optim, e.g. `control.corrHLfit = list(optim=list(control=list(r`

`processed` For programming purposes, not documented.

`family` Either a `family` or a `multi` value.

`...` Optional arguments passed to `HLCor`, `HLfit` or `designL.from.Corr`, for example the `distMatrix` argument of `HLCor`. Arguments that do not fit within these functions are detected and a warning is issued.

Details

Under the Matérn correlation model, corrHLfit typically performs a optimization over the ρ and ν parameters, with maximum possible values as set by `spaMM.options`.

By default `corrHLfit` will estimate correlation parameters by maximizing the `objective` value returned by `HLCor` calls wherein the dispersion parameters are estimated jointly with fixed effects for given correlation parameters. If dispersion parameters are specified in `init.corrHLfit`, they will also be estimated by maximizing the `objective` value, and `HLCor calls` will not estimate them jointly with fixed effects. This means that in general the fixed effect estimates may vary depending on `init.corrHLfit` when any form of REML correction is applied.

Correctly using `corrHLfit` for likelihood ratio tests of fixed effects may then by tricky. It is safe to perform full ML fits of all parameters (using `objective="p_v",HLmethod="ML"`) for such tests (see Examples). The higher level function `fixedLRT` is a safe interface for likelihood ratio tests using some form of REML estimation in `corrHLfit`.

`attr(<fitted object>,"optimInfo")$lower` and ...`$upper` gives the lower and upper bounds for optimization of correlation parameters. These are the default values if the user did not provide explicit values. For the adjacency model, the default values are the inverse of the maximum and minimum eigenvalues of the `adjMatrix`. For the Matérn model, the default values are not so easily summarized: they are intended to cover the range of values for which there is statistical information to distinguish among them.

Value

The return value of an `HLCor` call, with additional attributes. The `HLCor` call is evaluated at the estimated correlation parameter values. These values are included in the return object as its `$corrPars` member. The attributes added by `corrHLfit` include the original call of the function (which can be retrived by `getCall(`<fitted object>`)`, and information about the optimization call within `corrHLfit`.

See Also

See more examples on data set `Loaloa`. See `fixedLRT` for likelihood ratio tests.

Examples

```
# Example with an adjacency matrix (autoregressive model):
# see 'adjacency' documentation page

#### Examples with Matérn correlations
## A likelihood ratio test based on the ML fits of a full and of a null model.
if (spaMM.getOption("example_maxtime")>18) {
 data(blackcap)
 fullfit <- corrHLfit(migStatus ~ means+ Matern(1|latitude+longitude),data=blackcap,
                     HLmethod="ML")
 summary(fullfit)
 nullfit <- corrHLfit(migStatus ~ 1 + Matern(1|latitude+longitude),data=blackcap,
                     HLmethod="ML")
 summary(nullfit)
 ## p-value:
 1-pchisq(2*(logLik(fullfit)-logLik(nullfit)),df=1)
}

## see data set Loaloa for additional examples
```

---

corrMatrix *Using a corrMatrix argument*

Description

`corrMatrix` is an argument of `HLCor`, of calls `dist` or `matrix`, with is used if the model formula contains a term of the form `corrMatrix(1|<...>)`. It describes a correlation matrix, possibly as a `dist` object. A covariance matrix can actually be passed through this argument, but then it must be a full matrix, not a `dist` object. The way the rows and columns of the matrix are matched to the rows of the `data` depends on the nature of the grouping term <...>.

Details

The simplest case is illustrated in the first two examples below: the grouping term is identical to a single variable which is present in the `data`, whose levels match the rownames of the `corrMatrix`. As illustrated by the second example, the order of the data does not matter in that case, because the factor levels are used to match the `data` rows to the appropriate row and columns of the `corrMatrix`. The `corrMatrix` may even contain rows (and columns) in excess of the levels of the grouping term, in which case these rows are ignored.

These convenient properties no longer hold when the grouping term is not a single variable from the `data` (third example below), or when its levels do not correspond to row names of the matrix. In these cases, (1) no attempt is made to match the `data` rows to the row and column names of the `corrMatrix`. Such attempt could succeed only if the user had given names to the matrix matching those that the the called function could create from the information in the `data`, in which case the user should find easier to specify a single variable that can be matched; (2) the order of `data` and `corrMatrix` matter; Internally, a single factor variable is constructed from all levels of the variables in the grouping term (i.e., from all levels of `latitude` and `longitude`, in the third example), with levels 1,2,3... that are matched to rows 1,2,3... of the `corrMatrix`. Thus the first row of the data is always associated to the first row of the matrix; (3) further, the dimension of the matrix must match the number of levels implied by the grouping term. For example, one might consider the case of 14 response values but of correlations between only 7 levels of a random effect, with two responses for each level. Then the matrix must be of dimension 7x7.

Examples

```
data(blackcap)
## Here we manually reconstruct the correlation matrix
##  of the ML fit produced by corrHLfit:
MLcorMat <- MaternCorr(proxy::dist(blackcap[,c("latitude","longitude")]),
                        nu=0.6285603,rho=0.0544659)
blackcap$name <- as.factor(rownames(blackcap))
## (1) Single variable present in the data
HLCor(migStatus ~ means+ corrMatrix(1|name),data=blackcap,
      corrMatrix=MLcorMat,HLmethod="ML")
## (2) Same, permuted: still gives correct result
perm <- sample(14)
# Permuted matrix (with permuted names)
pmat <- as.matrix(MLcorMat)[perm,perm]
HLCor(migStatus ~ means+ corrMatrix(1|name),data=blackcap,
      corrMatrix=as.dist(pmat),HLmethod="ML")
## (3) Other grouping terms:
HLCor(migStatus ~ means+ corrMatrix(1|latitude+longitude),data=blackcap,
      corrMatrix=MLcorMat,HLmethod="ML")
```

---

designL.from.Corr *Computation of "square root" of correlation matrix*

Description

This function is not usually directly called by users, but arguments may be passed to it through higher-level calls (see Examples). For given correlation matrix **C**, it computes a "design matrix" **L** such that **C** = **L** * t(**L**). `t(chol(.))` (Cholesky factorization) is a fast method for this computation, but it is not robust numerically and may even return an error, in which cases more robust methods (`eigen` or `svd`) are used. Matrix roots are not unique (for example, they are lower triangular for `t(chol(.))`, and symmetric for `svd(.)`. As matrix roots are used to simulate samples under the fitted model (in particular in the parametric bootstrap implemented in `fixedLRT`), this implies that for given seed of random numbers, these samples will differ with these different methods (although their distribution should be identical).

Usage

```
designL.from.Corr(m, symSVD=NULL, try.chol=TRUE, try.eigen=FALSE, threshold=1e-06,
                  debug=FALSE, SVDfix=1/10)
```

Arguments

`m`	The matrix which 'root' is to be computed. This argument is ignored if `symSVD` is provided.
`symSVD`	A list representing the symmetric singular value decomposition of the matrix which 'root' is to be computed. Must have elements `$u`, a matrix of eigenvectors, and `$d`, a vector of eigenvalues.
`try.chol`	If `try.chol=TRUE`, the Cholesky factorization will be tried.
`try.eigen`	The default behavior is to try `chol`, and use `svd` if chol fails. If `try.eigen=TRUE`, the `eigen` factorization will be tried before `svd`. `eigen` is a compromise between speed and accuracy, but in our experience it may *hang* so by default it is not tried.
`threshold`	A correction threshold for low eigenvalues is the case and eigensystem or singular-value decomposition are used.
`debug`	Not documented, only for development purposes.
`SVDfix`	A solution to failures of `svd`: see Details.

Details

The function may call `svd`, for singular value decomposition (SVD) of a matrix $\mathbf{M}$. `svd` may return "error code 1 from Lapack routine 'dgesdd"' (cf. unhelpful discussions on R forums). This can be circumvented by computing the SVD of $(1-x)\mathbf{I}+x\mathbf{M}$ and deducing the singular values of $\mathbf{M}$ in a trivial way. The x value to be used in this fix is provided by the `SVDfix` argument.

`svd` errors have occurred for correlation matrices that were close to the identity matrix except for a few large non-diagonal elements. Such matrices may occur in particular for the Matérn correlation model with low ν, high ρ, and if some samples are spatially close. Then, an alternative fix to the `svd` problem may be to restrict the ν and/or ρ ranges, using the `lower` and `upper` arguments of `corrHLfit`, although one should make sure that this has no bearing on the inferences.

Value

The "square root of the input matrix". Its rows and columns are labelled according to the columns of the original matrix.

Examples

```
## Not run:
## try.chol argument passed to designL.from.Corr
## through the '...' argument of higher-level functions
## such as HLCor, corrHLfit, fixedLRT:
data(scotlip)
HLCor(cases~I(prop.ag/10) +adjacency(1|gridcode)+offset(log(scotlip$expec)),
      ranPars=list(rho=0.174),adjMatrix=Nmatrix,family=poisson(),
      data=scotlip,try.chol=FALSE)

## End(Not run)
```

`extractors` *Functions to extract various components of a fit*

Description

`logLik` extracts the log-likelihood (exact or approximated). `get_any_IC` and `AIC.HLfit` compute model selection/information criteria such as AIC. See the Details for more information about these criteria. `dev_resids` returns a vector of deviance residuals. `deviance` returns the sum of these deviance residuals. `fitted` extracts fitted values (see `fitted.values`). `fixef` extracts the fixed effects coefficients, β. `ranef` extracts the predicted random effects, u. `vcov` returns the variance-covariance matrix of the fixed-effects coefficients. `Corr` returns a correlation matrix of random effects (with restrictions, see Details). `getDistMat` extracts a distance matrix for a Matérn correlation model. `get_RLRTSim_args` extracts a list of arguments suitable for calls to `LRTSim::RLRTSim()`

Usage

```
## S3 method for class 'HLfit'
logLik(object,which,...)
## S3 method for class 'HLfit'
fitted(object,...)
## S3 method for class 'HLfit'
fixef(object,...)

## S3 method for class 'HLfit'
ranef(object,...)
## S3 method for class 'HLfit'
vcov(object,...)
## S3 method for class 'HLfit'
deviance(object,...)
getDistMat(object,scaled=FALSE)
Corr(object,...)
dev_resids(object,...)
get_any_IC(object, ..., verbose=interactive())
get_RLRTSim_args(object,...)
## S3 method for class 'HLfit'
AIC(object, ..., k, verbose=interactive())
```

Arguments

`object`	The return object of an HLfit or similar function.
`which`	Which element of the `APHLs` list to return. The default depends on the fitting method.In particular, if it was REML or one of its variants, the function returns the log restricted likelihood (exact or approximated).
`scaled`	If `FALSE`, the function ignores the scale parameter rho and returns unscaled distance.
`verbose`	Whether to print the model selection criteria or not.

`k`	For `AIC`, unused by `HLfit` method, but included to conform to the generic.
`...`	Other arguments that may be needed by some method.

Value

Return values are numeric (for `logLik`), vectors (most cases), matrices (for `vcov`), matrices or dist objects (for `getDistMat`). `ranef` returns a vector with attributes, which inherits from class `ranef` which has its own (undocumented) `print` method.

`Corr` currently returns the correlation matrix of the random effects which are described as **Lv** (see `HLfit`)

`get_any_IC` computes, optionally prints, and returns invisibly the following quantities. The **Effective degrees of freedom** for the random effects (approximately) characterizes the expectation of a goodness of fit statistic discussed by Lee and Nelder (2001), which gave a general formula for it in HGLMs. The **conditional AIC** (Vaida and Blanchard 2005) is notable in involving the conditional likelihood and the effective degrees of freedom. Lee et al. (2006) and Ha et al (2007) defined a corrected AIC [i.e., AIC(D*) in their eq. 7] that is here interpreted as the conditional AIC. The conditional AIC returned by `HLfit` includes both this effective df, the df for estimated fixed effects, and the df for estimated parameters of the variance of random effects.

Also returned are the **marginal AIC** (Akaike's classical AIC), and a focussed AIC for dispersion parameters (**dispersion AIC**) discussed by Ha et al (2007; eq.10). This diversity of criteria should encourage users to think twice before applying model selection automatically, which is no better although more fashionable than misuses of simple null hypothesis testing. Also, alternative procedures for model choice can be considered (e.g. Cox and Donnelly, 2011, p. 130-131).

`get_RLRTSim_args` extracts a list of arguments suitable for a call to `LRTSim::RLRTSim()` for a small-sample test of the presence of a random effect by an efficient simulation procedure. The test can be run by
`do.call("RLRTSim",<get_RLRTSim_args return value>)`.

References

Ha, I. D., Lee, Y. and MacKenzie, G. (2007) Model selection for multi-component frailty models. Statistics in Medicine 26: 4790-4807.

Lee, Y., Nelder, J. A. (2001) Hierarchical generalised linear models: A synthesis of generalised linear models, random-effect models and structured dispersions. Biometrika 88, 987-1006.

Lee, Y., Nelder, J. A. and Pawitan, Y. (2006) Generalized linear models with random effects: unified analysis via h-likelihood. Chapman & Hall: London.

Vaida, F., and Blanchard, S. (2005) Conditional Akaike information for mixed-effects models. Biometrika 92, 351-370.

Examples

```
data(wafers)
m1 <- HLfit(y ~X1+X2+(1|batch),
          resid.model = ~ 1 ,data=wafers,HLmethod="ML")
get_any_IC(m1)
fixef(m1)
vcov(m1)
ranef(m1)
```

`fitme` *Fitting function for fixed- and mixed-effect models with GLM response.*

Description

This is a common interface for fitting most models that spaMM can fit, therefore substituting to `corrHLfit`, `HLCor` and `HLfit`. Whene possible, it uses generic optimization methods for estimating all dispersion parameters, rather than the general, but often slow, default method in `HLfit`. Random-slope models are not yet handled by this function. Currently it is fast for some models and slow for others, with few general guidance except that `fitme` should generally be faster than the alternative fitting functions for large data sets when the residual variance model is a single constant term (no structured dispersion).

Usage

```
fitme(formula, data, family = gaussian(), init = list(), fixed = list(),
      lower = list(), upper = list(), resid.model = ~1, init.HLfit = list(),
      control = list(), control.dist = list(), method = "ML",
      HLmethod = method, processed = NULL, ...)
```

Arguments

`formula`	Either a linear model `formula` (as handled by various fitting functions) or a `predictor`, i.e. a formula with attributes (see `Predictor` and examples below). See Details in `spaMM` for allowed terms in the formula.
`data`	A data frame containing the variables in the response and the model formula.
`family`	Either a `family` or a `multi` value.
`init`	An optional list of initial values for correlation and/or dispersion parameters and/or response family parameters, e.g. `list(rho=1,nu=1,lambda=1,phi=1)` where `rho` and `nu` are parameters of the Matérn family (see `Matern`), and `lambda` and `phi` are dispersion parameters (see Details in `spaMM` for the meaning of these parameters). All are optional, but giving values for a dispersion parameter changes the ways it is estimated (see Details). `rho` may be a vector (see `make_scaled_dist`) and, in that case, it is possible that some or all of its elements are `NA`, for which `corrHLfit` substitute automatically determined values.
`fixed`	A list similar to `init`, but specifying fixed values of the parameters not estimated.
`lower`	An optional list of values of parameters specified through `init.corrHLfit`, used as lower values in calls to `optim`. See Details for default values.
`upper`	Same as `lower`, but upper values.
`resid.model`	See identically named `HLfit` argument.
`init.HLfit`	See identically named `HLfit` argument.
`control`	A list of control parameters, with two possible elements:

- `$nloptr`, itself a list of control parameters to be copied in the `opts` argument of `nloptr`. Default controls are `list(algorithm="NLOPT_LN_BOBYQA",xtol_rel=1.0e-4,maxeval=-1,print_level=0)`
- `$refit`, a boolean, or a list of booleans with possible elements `$phi` and `$lambda`. If either element is set to TRUE, then the corresponding parameters are refitted by the internal `HLfit` methods (see Details). If `$refit` is TRUE, all `phi` and `lambda` parameters are refit. By default none of them is refitted.

`control.dist` See `control.dist` in `HLCor`

`method, HLmethod`
"ML" or "REML". "ML" is the default, in contrast to "REML" for the `HLmethod` argument of other fitting functions. Other possible values of `HLfit`'s `HLmethod` argument are handled and all should give results close to the other fitting methods with the same `HLmethod` argument.

`processed` For programming purposes, not documented.

`...` Optional arguments passed to `HLCor`, `HLfit` or `designL.from.Corr`, for example the `distMatrix` argument of `HLCor`.

Details

`fitme` uses `nloptr` for optimization, where `corrHLfit` uses `optim`.

By default `corrHLfit` initiates optimization with the best of a grid of parameter values, while `fitme` performs no such grid search for initial value. The grid search may help find a better maximum (as is the case in the toy examples based on the `blackcap` data set), but may as well trap the function in an inferior local maximum.

For `phi` and `lambda`, `fitme` does not use the internal fitting methods of `HLfit`. The latter methods are well suited for structured dispersion models, but require the computation of the so-called leverages, which can be slow. However, they also provide some more information such as the "cond. SE" (abut which see warning in Details of `HLfit`). To obtain such information from a `fitme` call, use the `control$refit` argument (see Example).

Value

The return value of an `HLCor` or an `HLfit` call, with additional attributes. The `HLCor` call is evaluated at the estimated correlation parameter values. These values are included in the return object as its `$corrPars` member. The attributes added by `fitme` include the original call of the function (which can be retrived by `getCall(`<fitted object>`)`, and information about the optimization call within `fitme`.

Examples

```
## Example with 'refit'
# We simulate Gamma deviates with mean mu=3 and variance=2,
#  ie. phi= var/mu^2= 2/9 in the (mu, phi) parametrization of a Gamma
#  GLM; and shape=9/2, scale=2/3 in the parametrisation of rgamma().
#  Note that phi is not equivalent to scale:
#  shape = 1/phi and scale = mu*phi.
set.seed(123)
```

```
gr <- data.frame(y=rgamma(100,shape=9/2,scale=2/3))
fitme(y~1,data=gr,family=Gamma(log),control=list(refit=list(phi=TRUE)))

## see help("COMPoisson"), help("negbin"), help("Loaloa"), etc., for further examples.
```

---

fixedLRT *Likelihood ratio test of fixed effects.*

Description

fixedLRT performs a likelihood ratio (LR) test between two models, the "full" and the "null" models, currently differing only in their fixed effects. Parametric bootstrap p-values can be computed, either using the raw bootstrap distribution of the likelihood ratio, or a a bootstrap estimate of the Bartlett correction of the LR statistic. This function differ from LRT in its arguments (model fits for LRT, but all arguments required to fit the models for fixedLRT), and in the format of its return value.

Usage

```
fixedLRT(null.formula, formula, data,
         method, HLmethod = method, REMLformula=NULL,
         boot.repl=0, control=list(),control.boot=list(),
         fittingFunction = "corrHLfit", ...)
```

Arguments

null.formula	Either a formula (as in glm) or a predictor (see Predictor) for the null model.
formula	Either a formula or a predictor for the full model.
data	A data frame containing the variables in the model.
method	A method to fit the full and null models. See HLfit's HLmethod argument for background information about such methods. The two most meaningful values of method in fixedLRT calls are: 'ML' for an LRT based on ML fits (generally recommended); and 'PQL/L' for an LRT based on PQL/L fits (recommended for spatial binary data). Also feasible, but more tricky, and not really recommended (see Rousset and Ferdy, 2014), is 'REML'. This will perform an LRT based on two REML fits of the data, *both* of which use the same conditional (or "restricted") likelihood of residuals for estimating dispersion parameters λ and ϕ (see REMLformula argument). Further, REML will not be effective on a given dispersion parameter if a non-trivial init.corrHLfit value is provided for this parameter.
HLmethod	Kept for back-compatibility. Same as method, but wll work only for fittingFunction=corrHLfit.
REMLformula	a formula specifying the fixed effects which design matrix is used in the REML correction for the estimation of dispersion parameters, if these are estimated by REML. This formula is by default that for the *full* model.
boot.repl	the number of bootstrap replicates.

`control`	A set of control parameters for the fits of the data, mostly for development purposes. However, if an initial value is provided for a dispersion parameter, a better one may be sought if further `control=list(prefits=TRUE)` (the effect appears small, however).
`control.boot`	Same as `control`, but for the fits of the bootstrap replicates. Again, the option `control.boot=list(prefits=TRUE)` may yield a small improvement in the fits, at the expense of more computation time.
`fittingFunction`	The function used to fit each model: either `corrHLfit` or `fitme`.
`...`	Further arguments passed to or from other methods; in particular, additional arguments passed to `corrHLfit`, including mandatory ones such as `data` and those ultimately passed to `designL.from.Corr`. With respect to the latter, note that `try.chol` affects the simulation of samples for the parametric bootstrap, and although ultimate differences in performance may be small, `try.chol=FALSE` may be slightly better.

Details

Comparison of REML fits is a priori not suitable for performing likelihood ratio tests. Nevertheless, it is possible to contrive them for testing purposes (Wehlam & Thompson 1997). This function generalizes some of Wehlam & Thompson's methods to GLMMs.

See Details in `LRT` for details of the bootstrap procedures.

Value

An object of class `fixedLRT`, actually a list with as-yet unstable format, but here with typical elements (depending on the options)

`fullfit`	the HLfit object for the full model;
`nullfit`	the HLfit object for the null model;
`LRTori`	A likelihood ratio chi-square statistic
`LRTprof`	Another likelihood ratio chi-square statistic, after a profiling step, if any.
`df`	the number of degrees of freedom of the test.
`trace.info`	Information on various steps of the computation.
`bootreps`	A table of fitted likelihoods for bootstrap replicates.
`meanbootLRT`	The mean likelihood ratio chi-square statistic for boostrap replicates.

References

Rousset F., Ferdy, J.-B. (2014) Testing environmental and genetic effects in the presence of spatial autocorrelation. Ecography, 37: 781-790. http://dx.doi.org/10.1111/ecog.00566

Welham, S. J., and Thompson, R. (1997) Likelihood ratio tests for fixed model terms using residual maximum likelihood, J. R. Stat. Soc. B 59, 701-714.

See Also

See also `corrHLfit` and `LRT`.

Examples

```
if (spaMM.getOption("example_maxtime")>18) {
 data(blackcap)
 ## result comparable to the corrHLfit examples based on blackcap
 fixedLRT(null.formula=migStatus ~ 1 + Matern(1|latitude+longitude),
       formula=migStatus ~ means + Matern(1|latitude+longitude),
       HLmethod='ML',data=blackcap)
}
if (spaMM.getOption("example_maxtime")>1800) {
 ## longer version with bootstrap
 fixedLRT(null.formula=migStatus ~ 1 + Matern(1|latitude+longitude),
       formula=migStatus ~ means + Matern(1|latitude+longitude),
       HLmethod='ML',data=blackcap, boot.repl=100)
 }
```

`freight` *Freight dataset*

Description

A set of data on airfreight breakage. Data are given on 10 air shipments, each carrying 1000 ampules of some substance. For each shipment, the number of ampules found broken upon arrival, and the number of times the shipments were transferred from one aircraft to another, are recorded.

Usage

```
data(freight)
```

Format

The data frame includes 10 observations on the following variables:

broken number of ampules found broken upon arrival.

transfers number of times the shipments were transferred from one aircraft to another.

id Shipment identifier.

Source

The data set is reported by Kutner et al. (2003) and used by Sellers and Shmueli (2010) to illustrate COMPoisson analyses.

References

Kutner MH, Nachtsheim CJ, Neter J, Li W (2005, p. 35). Applied Linear Regression Models, Fourth Edition. McGraw-Hill.

Sellers KF, Shmueli G (2010) A Flexible Regression Model for Count Data. Ann. Appl. Stat. 4: 943–961

Examples

```
## see ?COMPoisson for examples
```

HLCor *Fits a (spatially) correlated mixed model, for given correlation parameters*

Description

A convenient interface for HLfit, constructing the correlation matrix of random effects from the arguments, then estimating fixed effects and dispersion parameters using HLfit.

Usage

```
HLCor(formula, data, family = gaussian(), ranPars = NULL, distMatrix,
      uniqueGeo = NULL, adjMatrix, corrMatrix,
      verbose = c(warn=TRUE,trace=FALSE,summary=FALSE),
      control.dist = list(), ...)
```

Arguments

formula	A predictor, i.e. a formula with attributes (see Predictor), or possibly simply a simple formula if an offset is not required.
ranPars	A list of values for correlation parameters (some of which are mandatory), and possibly also dispersion parameters (optional, but passed to HLfit if present). For the Matérn model, the correlation parameters are rho (scale parameter(s)), nu (smoothness parameter), and (optional) Nugget (see Matern). The rho parameter can itself be a vector with different values for different geographic coordinates. For the "adjacency" model, the only correlation parameter is a scalar rho. The dispersion parameters are the random effect variance, lambda, and the residual variance, phi.
data	The data frame to be analyzed.
family	A family object describing the distribution of the response variable. See HLfit for further information.
distMatrix	A distance matrix between geographic locations, forwarded to MaternCorr
uniqueGeo	A matrix of non-redundant geographic locations. This is useful if the rho parameter is a vector with different values for different geographic coordinates, in which case a scaled distance matrix has to be reconstructed from uniqueGeo and rho.
adjMatrix	An adjacency matrix, used if a random effect of the form y ~ adjacency(1|<location index>) is present. Its rows and columns must be ordered as increasing values of the levels of the geographic location index specifying the spatial random effect. For example, if the model formula is y ~ adjacency(1|geo.loc) and <data>$geo.loc is 2,4,3,1,... the first row/column of the matrix refers to the fourth row of the data.

`corrMatrix` A arbitrary (valid) correlation matrix, used if a random effect term of the form `corrMatrix(1|<stuff>)` is present. Each row corresponds to levels of a variable <stuff>. This allows to analyze non-spatial model by giving for example a matrix of genetic correlations. See `corrMatrix` for details.

`verbose` A vector of booleans. `trace` controls various diagnostic (possibly messy) messages about the iterations. `summary` controls whether a summary of the fit is called by `HLfit`. `warn` is for programming purposes and best ignored.

`control.dist` A list of arguments that control the computation of the distance argument of the correlation functions. Possible elements are

- **rho.mapping** a set of indices controlling which elements of the `rho` scale vector scales which dimension(s) of the space in which (spatial) correlation matrices of random effects are computed. See same argument in `make_scaled_dist` for details and examples.
- **dist.method** `method` argument of `proxy::dist` function (by default, `"Euclidean"`, but e.g. `"Geodesic"` can be used to compute spherical distances.)

`...` Further parameters passed to `HLfit` or to `designL.from.Corr`.

Details

The correlation matrix for random effects can be specified by various combination of formula terms and other arguments (see Examples):

Basic Matérn model `Matern(1|<...>)` with `ranPars` argument, using the spatial coordinates in `<...>`. This will construct a correlation matrix according to the Matérn correlation function (see `MaternCorr`);

Matérn model with given distance matrix `Matern(1|<...>)` with `distMatrix` and `ranPars` argument;

Given correlation matrix `corrMatrix(1|<...>)` with `corrMatrix` argument. See `corrMatrix` for further details.

CAR model with given adjacency matrix `adjacency(1|<...>)` with `adjMatrix` and `ranPars` argument. This will construct the correlation matrix $(\mathbf{I}-\rho$ `adjMatrix`$)^{-1}$ for a conditional autoregressive (CAR) model. This was implemented for comparison purposes as it is widely used, but is not really recommended for irregular lattices (see Wall, 2004 and Martellosio, 2012 for some insights on the implications of autoregressive models). See `adjacency` documentation for more information on fitting such models.

Value

The return value of an `HLfit` call, with the following additional attributes:

`HLCorcall` the HLCor call

`info.uniqueGeo` Unique geographic locations.

References

Wall M.M. (2004) A close look at the spatial structure implied by the CAR and SAR models: Journal of Statistical Planning and Inference 121: 311-324.

Martellosio, F. (2012) The correlation structure of spatial autoregressions, Econometric Theory 28, 1373-1391.

See Also

MaternCorr, HLfit, corrHLfit

Examples

```
# Example with an adjacency matrix (autoregressive model):
# see 'adjacency' documentation page

#### Matérn correlation using only the Matern() syntax
if (spaMM.getOption("example_maxtime")>2) {
 data(Loaloa)
 HLCor(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
              +Matern(1|longitude+latitude),data=Loaloa,
      family=binomial(),ranPars=list(nu=0.5,rho=1/0.7))
}


#### Matérn correlation using a distMatrix
data(blackcap)
MLdistMat <- as.matrix(proxy::dist(blackcap[,c("latitude","longitude")]))
HLCor(migStatus ~ means+ Matern(1|latitude+longitude),data=blackcap,
      distMatrix=MLdistMat,HLmethod="ML",ranPars=list(nu=0.6285603,rho=0.0544659))
```

HLfit *Fit mixed models with given correlation matrix*

Description

This function fits GLMMs as well as some hierarchical generalized linear models (HGLM; Lee and Nelder 2001). HLfit fits both fixed effects parameters, and dispersion parameters i.e. the variance of the random effects and the variance of the residual error. The linear predictor is of the standard form `offset+ X beta + Z v`, where X is the design matrix of fixed effects and Z is a design matrix of random effects. The function also handles a linear predictor (with only fixed effects) for the residual variance.

Usage

```
HLfit(formula, data, family = gaussian(), rand.family = gaussian(),
      resid.model = ~1, resid.formula, REMLformula = NULL,
      verbose = c(warn = TRUE, trace = FALSE, summary = FALSE),
      HLmethod = "HL(1,1)", control.HLfit = list(),
      control.glm = list(), init.HLfit = list(), ranFix = list(),
      etaFix = list(), prior.weights = NULL, processed = NULL)
## see 'rand.family' argument for inverse.Gamma
```

Arguments

formula
: A formula; or a predictor, i.e. a formula with attributes created by Predictor, if design matrices for random effects have to be provided. See Details in spaMM for allowed terms in the formula (except spatial ones).

data
: A data frame containing the variables named in the model formula.

family
: A family object describing the distribution of the response variable. Possible values include the gaussian, poisson, binomial, Gamma, negbin and COMPoisson families. Possible combinations of family and link are those allowed by each of these families (see family for the first four, and specific documentation pages for the last two).

rand.family
: A family object describing the distribution of the random effect, or a list of family objects for different random effects (see Examples). Possible options are gaussian(), Gamma(log), Gamma(identity) (see Details), Beta(logit), inverse.Gamma(-1/mu), and inverse.Gamma(log). For discussion of these alternatives see Lee and Nelder 2001 or Lee et al. 2006, p. 178-. Here the family gives the distribution of a random effect u and the link gives v as function of u (see Details). If there are several random effects and only one family is given, this family holds for all random effects.

resid.model
: **Either** a formula (without left-hand side) for the dispersion parameter phi of the residual error. A log link is assumed by default;
or a list, with at most two possible elements if its formula involves only fixed effects:

formula model formula as in formula-only case, without left-hand side

family Always Gamma, with by default a log link. Gamma(identity) can be tried but may fail because only the log link ensures that the fitted ϕ is positive.

and additional possible elements (all named as fitme arguments) if its formula involves random effects: see phiHGLM.

resid.formula
: Obsolete, for back-compatibility; will be deprecated. Same as formula in resid.model.

REMLformula
: A model formula that allows the estimation of dispersion parameters, and computation of restricted likelihood (p_bv) under a model different from the predictor formula.

For example, if only random effects are included in REMLformula, an ML fit is performed and p_bv equals the marginal likelihood (or its approximation), p_v. This ML fit can be performed more simply by setting HLmethod="ML" and leaving REMLformula at its default NULL value.

verbose
: A vector of booleans. trace controls various diagnostic (possibly messy) messages about the iterations. summary controls whether a summary of the fit is called by HLfit. warn is for programming purposes and best ignored.

HLmethod
: Allowed values are "REML", "ML", "EQL-" and "EQL+" for all models; "PQL" (="REPQL") and "PQL/L" for GLMMs only; and (only for those curious to experiment) expressions of the form "HL(<...>)", "ML(<...>)" and "RE(<...>)". HL and RE are equivalent. The default behaviour is RE(1,1), which **by default performs REML** (standard REML for LMMs, an extended definition for other

models). But it can also perform non-standard forms of REML (indeed including ML), depending on the `REMLformula` given.

EQL stands for the EQL method of Lee and Nelder (2001). The '+' version includes the d v/ d tau correction described p. 997 of that paper, and the '-' version ignores it. PQL can be seen as the version of EQL- for GLMMs. PQL/L is PQL without the leverage corrections that define REML estimation of random-effect parameters.

For **GLMs**, the default is still REML. For binomial and Poisson GLMs, this cannot be distinguished from an exact ML fit. Note that the `glm` function performs an EQL analysis, which will differ from the REML (and ML) one for Gamma GLMs.

See Details for the more general syntax.

`control.HLfit` A list of parameters controlling the fitting algorithms.

`resid.family` allows one to change the link for modeling of residual variance ϕ, which is `"log"` by default. The family is always Gamma, so the non-default possible values of `resid.family` are `Gamma(identity)` or `Gamma(inverse)`. Only the default value ensures that the fitted ϕ is positive.

Controls for the fitting algorithms should be ignored in routine use. They are

`conv.threshold`, a convergence threshold for the iterative algorithm, which controls whether linear predictor terms (fixed effects and inferred random effects), and dispersion parameter estimates have converged. Defaults to 1e-05;

`break_conv_logL`, a boolean specifying whether the iterative algorithm should terminate when log-likelihood appears to have converged (roughly, when its relative variation over on iteration is lower than 1e-8). Default is FALSE (convergence is then assessed on the parameter estimates rather than on log-likelihood).

`iter.mean.dispFix`, the number of iterations of the iterative algorithm for coefficients of the linear predictor, if no dispersion parameters are estimated by the iterative algorithm. Defaults to 200;

`iter.mean.dispVar`, the number of iterations of the iterative algorithm for coefficients of the linear predictor, if some dispersion parameter(s) is estimated by the iterative algorithm. Defaults to 50;

`max.iter`, the number of iterations of the iterative algorithm for joint estimation of dispersion parameters and of coefficients of the linear predictor. Defaults to 200. This is typically much more than necessary, unless there is little information to separately estimate λ and ϕ parameters.

`control.glm` List of parameters controlling GLM fits, passed to `glm.control`; e.g. `control.glm=list(maxit=100)`. See `glm.control` for further details.

`init.HLfit` A list of initial values for the iterative algorithm, with possible elements of the list are `fixef` for fixed effect estimates (beta), `v_h` for random effects vector **v** in the linear predictor, `lambda` for the parameter determining the variance of random effects u as drawn from the `rand.family` distribution `phi` for the residual variance. However, this argument can be ignored in routine use.

`ranFix` A list of fixed values of random effect parameters, with possible elements `lambda`, and also `phi` for gaussian and Gamma HGLMs. Inhibits the estimation of these parameters.

etaFix
: A list of fixed values of the coefficients of the linear predictor, with currently documented element `beta`. `etaFix$beta` should be a vector with names matching (a subset of) coefficient names of a fit without fixed values. It provides a convenient interface for fixing (some of) the fixed-effects coefficients (β). In contrast to an offset specification, it affects the REML correction for estimation of dispersion parameters, which depends only on which β coefficients are estimated. However, for non-standard use, REML can still be performed as if all β coefficients were estimated, by adding attribute `keepInREML=TRUE` to `etaFix$beta` (see Examples). These different behaviours will be overridden whenever a non-null `REMLformula` is provided.

prior.weights
: An optional vector of prior weights as in `glm`. This fits the data to a model with residual variance `phi/prior.weights`, so that increasing the weights by a constant factor f will yield (Intercept) estimates of `phi` also increased by f (this effect cannot be generally achieved if a non-trivial `resid.formula` with log link is used). This is not necessarily the way prior weights are interpreted in widely used packages, but this is consistent with what `glm`.

processed
: A list of preprocessed arguments, for programming purposes only (as in `corrHLfit`).

Details

I. Fitting methods: Many approximations for likelihood have been defined to fit mixed models (e.g. Noh and Lee (2007) for some overview), and this function only considers a subset of them, but it adds a new complication in terms of REML methods. For example, PQL as originally defined by Breslow and Clayton uses REML to estimate dispersion parameters, but this function allows one to use ML instead. Moreover, it allows some non-standard specification of the model formula that determines the conditional distribution used in REML.

In the more general syntax for `HLmethod`, used as e.g. `HLmethod="RE(1,1)"` the first '1' means that a first order Laplace approximation to the likelihood is used to estimate fixed effects (a '0' would instead mean that the h likelihood is used as the objective function). The second '1' means that a first order Laplace approximation to the likelihood or restricted likelihood is used to estimate dispersion parameters, including the dv/d tau term specifically discussed by Lee & Nelder 2001, p. 997 (a '0' would instead mean that these terms are ignored).

It is possible to enforce the EQL approximation for estimation of dispersion parameter by adding a third index with value 0. `"HL(0,1,0)"` is Lee & Nelder's (2001) method, i.e. `"EQL+"`.

For a Gamma GLM with log link, ML and EQL results will differ in their phi estimates, and the EQL estimate will match that from the `glm` function.

II. Random effects are constructed in several steps. first, a vector **u** of independent and identically distributed (iid) random effects is drawn from some distribution; second, a transformation v=f(u) is applied to each element (this defines **v** which elements are still iid); third, correlated random effects are obtained as **Lv** where **L** is the "square root" of a correlation matrix (this may be meaningful only for Gaussian random effects). Finally, a matrix **Z** (or sometimes **ZA**, see `Predictor`) allows to specify how the correlated random effects affect the response values. In particular, **Z** is the identity matrix if there is a single observation (response) for each location, but otherwise its elements z_{ji} are 1 for the jth observation in the ith location. The design matrix for **v** is then of the form **ZL**.

The specification of the random effects u and v handles the following cases: **Gaussian** with zero mean, unit variance, and identity link; **Beta**-distributed, where $u\,B(1/(2\lambda), 1/(2\lambda))$ with mean=1/2, and var= $\lambda/[4(1+\lambda)]$; and with logit link `v=logit(u)`; **Gamma**-distributed random effects, where

u Gamma(shape=1+1/λ,**scale=**1/λ): see Gamma for allowed links and further details; and **Inverse-Gamma**-distributed random effects, where u inverse-Gamma(shape=1+1/λ,**rate=**1/λ): see inverse.Gamma for allowed links and further details.

III. The standard errors reported may sometimes be misleading. For each set of parameters among β, λ, and ϕ parameters these are computed assuming that the other parameters are known without error. This is why they are labelled `Cond. SE` (conditional standard error). This is most uninformative in the unusual case where λ and ϕ are not separately estimable parameters. Further, the SEs for λ and ϕ are rough approximations as discussed in particular by Smyth et al. (2001; V_1 method).

Value

An object of class `HLfit`, which is a list with many elements, not all of which are documented.

A few extractor functions are available (see `extractors`), and should be used as far as possible as they should be backward-compatible from version 1.4 onwards, while the structure of the return object may still evolve. The following information will be useful for extracting further elements of the object.

Elements include **descriptors of the fit**:

- `eta` — Fitted values on the linear scale (including the predicted random effects);
- `fv` — Fitted values (μ =<inverse-link>(η)) of the response variable (returned by the `fitted` function);
- `fixef` — The fixed effects coefficients, β (returned by the `fixef` function);
- `ranef` — The random effects u (returned by the `ranef` function);
- `v_h` — The random effects on the linear scale, v;
- `phi` — The residual variance ϕ;
- `phi.object` — A possibly more complex object describing ϕ;
- `lambda` — The random effects (u) variance λ;
- `lambda.object` — A possibly more complex object describing λ;
- `corrPars` — Agglomerates information on correlation parameters, either fixed, or estimated by `HLfit`, `corrHLfit` or `fitme`;
- `APHLs` — A list which elements are various likelihood components, include conditional likelihood, h-likelihood, and the two adjusted profile h-likelihoods: the (approximate) marginal **likelihood** `p_v` and the (approximate) **restricted likelihood** `p_bv` (the latter two available through the `logLik` function). See the extractor function `get_any_IC` for information criteria (“AIC”) and effective degrees of freedom;

  The covariance matrix of β estimates is not included as such, but can be extracted by `vcov`;

Information about the input is contained in output elements named as `HLfit` or `corrHLfit` arguments (`data,family,resid.family,ranFix,prior.weights`), with the following notable exceptions or modifications:

- `predictor` — The linear predictor, including the `formula` (possibly reformatted) and several attributes;

`resid.predictor`
: Analogous to `predictor`, for the residual variance;

`rand.families`
: corresponding to the `rand.family` input;

Further miscellaneous diagnostics and descriptors of model structure:

`X.pv`
: The design matrix for fixed effects;

`ZAlist`
: Components of the design matrix for random effects. The design matrix for the random effects v is not included as such, but can be extracted by the *member_extractor* `get_ZALMatrix`.

`fixef_terms,fixef_levels`
: Futher information about fixed effect model;

`weights`
: (binomial data only) the binomial denominators;

`y`
: the response vector; for binomial data, the frequency response.

`models`
: Additional information on model structure for η, λ and ϕ;

`HL`
: A set of indices that characterize the approximations used for likelihood;

`leve_phi,lev_lambda`
: Leverages;

`warnings`
: A list of warnings for events that may have occurred during the fit.

Finally, the object includes programming tools: `call, spaMM.version` and `member_extractors`.

References

Breslow, NE, Clayton, DG. (1993). Approximate Inference in Generalized Linear Mixed Models. Journal of the American Statistical Association 88, 9-25.

Cox, D. R. and Donnelly C. A. (2011) Principles of Applied Statistics. Cambridge Univ. Press.

Lee, Y., Nelder, J. A. (2001) Hierarchical generalised linear models: A synthesis of generalised linear models, random-effect models and structured dispersions. Biometrika 88, 987-1006.

Lee, Y., Nelder, J. A. and Pawitan, Y. (2006). Generalized linear models with random effects: unified analysis via h-likelihood. Chapman & Hall: London.

Noh, M., and Lee, Y. (2007). REML estimation for binary data in GLMMs, J. Multivariate Anal. 98, 896-915.

Smyth GK, Huele AF, Verbyla AP (2001). Exact and approximate REML for heteroscedastic regression. Statistical Modelling 1, 161-175.

See Also

`HLCor` for estimation with given spatial correlation parameters; `corrHLfit` for joint estimation with spatial correlation parameters; `fitme` as an alternative to all these functions.

Examples

```
data(wafers)
## Gamma GLMM with log link

HLfit(y ~X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          resid.model = ~ X3+I(X3^2) ,data=wafers)

## Gamma - inverseGamma HGLM with log link
HLfit(y ~X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          HLmethod="HL(1,1)",rand.family=inverse.Gamma(log),
          resid.model = ~ X3+I(X3^2) ,data=wafers)
```

`inverse.Gamma` *Distribution families for Gamma and inverse Gamma-distributed random effects*

Description

For dispersion parameter λ, `Gamma` means that random effects are distributed as u Gamma(`shape=`1+1/λ,**scale=**1/λ), so u has mean 1 and variance λ. Both the log ($v = log(u)$) and identity ($v = u$) links are possible, though in the latter case the variance of u is constrained below 1 (otherwise Laplace approximations fail).

The two-parameter inverse Gamma distribution is the distribution of the reciprocal of a variable distributed according to the Gamma distribution Gamma with the same shape and scale parameters. `inverse.Gamma` implements the one-parameter inverse Gamma family with shape=1+1/λ and **rate**=1/λ) (rate=1/scale). It is used to model the distribution of random effects. Its mean=1; and its variance =$\lambda/(1-\lambda))$ if $\lambda < 1$, otherwise infinite. The default link is `"-1/mu"`, in which case `v=-1/u` is "-Gamma"-distributed with the same shape and rate, hence with mean $-(\lambda+1)$ and variance $\lambda(\lambda+1)$, which is a different one-parameter Gamma family than the above-described `Gamma`. The other possible link is `v=log(u)` in which case
$v \ -log(X$ Gamma$(1+1/\lambda, 1/\lambda))$, with mean $-(log(1/\lambda)+$digamma$(1+1/\lambda))$ and variance trigamma$(1+1/\lambda)$.

Usage

```
inverse.Gamma(link = "-1/mu")
# Gamma(link = "inverse") using stats::Gamma
```

Arguments

`link`	For `Gamma`, allowed links are `log` and `identity` (the default link from `Gamma`, `"inverse"`, cannot be used for the random effect specification). For `inverse.Gamma`, allowed links are `"-1/mu"` (default) and `log`.

Examples

```
# see help("HLfit") for fits using the inverse.Gamma distribution.
```

Loaloa *Loa loa prevalence in North Cameroon, 1991-2001*

Description

This data set describes prevalence of infection by the nematode *Loa loa* in North Cameroon, 1991-2001. This is a superset of the data discussed by Diggle and Ribeiro (2007) and Diggle et al. (2007). The study investigated the relationship between altitude, vegetation indices, and prevalence of the parasite.

Usage

```
data(Loaloa)
```

Format

The data frame includes 197 observations on the following variables:

latitude latitude, in degrees.

longitude longitude, in degrees.

ntot sample size per location

npos number of infected individuals per location

maxNDVI maximum normalised-difference vegetation index (NDVI) from repeated satellite scans

seNDVI standard error of NDVI

elev1 altitude, in m.

elev2,elev3,elev4 Additional altitude variables derived from the previous one, provided for convenience: respectively, positive values of altitude-650, positive values of altitude-1000, and positive values of altitude-1300

maxNDVI1 a copy of maxNDVI modified as `maxNDVI1[maxNDVI1>0.8] <- 0.8`

Source

The data were last retrieved on March 1, 2013 from P.J. Ribeiro's web resources at www.leg.ufpr.br/doku.php/pessoais:paulojus:mbgbook:datasets.

References

Diggle, P., and Ribeiro, P. 2007. Model-based geostatistics, Springer series in statistics, Springer, New York.

Diggle, P. J., Thomson, M. C., Christensen, O. F., Rowlingson, B., Obsomer, V., Gardon, J., Wanji, S., Takougang, I., Enyong, P., Kamgno, J., Remme, J. H., Boussinesq, M., and Molyneux, D. H. 2007. Spatial modelling and the prediction of Loa loa risk: decision making under uncertainty, Ann. Trop. Med. Parasitol. 101, 499-509.

Examples

```
data(Loaloa)

### Variations on the model fit by Diggle et al.
###    on a subset of the Loaloa data
### In each case this shows the slight differences in syntax,
###    and the difference in 'typical' computation times,
###    when fit using corrHLfit() or fitme().

if (spaMM.getOption("example_maxtime")>13) {
  corrHLfit(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
                   +Matern(1|longitude+latitude),HLmethod="HL(0,1)",
                 data=Loaloa,family=binomial(),ranFix=list(nu=0.5))
}
if (spaMM.getOption("example_maxtime")>3) {
  fitme(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
                   +Matern(1|longitude+latitude),method="HL(0,1)",
                 data=Loaloa,family=binomial(),fixed=list(nu=0.5))
}

if (spaMM.getOption("example_maxtime")>49) {
  corrHLfit(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
            +Matern(1|longitude+latitude),
              data=Loaloa,family=binomial(),ranFix=list(nu=0.5))
}
if (spaMM.getOption("example_maxtime")>25) {
  fitme(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
            +Matern(1|longitude+latitude),
              data=Loaloa,family=binomial(),fixed=list(nu=0.5),method="REML")
}

## Diggle and Ribeiro (2007) assumed (in this package notation) Nugget=2/7:
if (spaMM.getOption("example_maxtime")>45) {
  corrHLfit(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
           +Matern(1|longitude+latitude),
             data=Loaloa,family=binomial(),ranFix=list(nu=0.5,Nugget=2/7))
}
if (spaMM.getOption("example_maxtime")>17) {
  fitme(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
           +Matern(1|longitude+latitude),method="REML",
             data=Loaloa,family=binomial(),fixed=list(nu=0.5,Nugget=2/7))
}

## with nugget estimation:
if (spaMM.getOption("example_maxtime")>187) {
  corrHLfit(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
           +Matern(1|longitude+latitude),
             data=Loaloa,family=binomial(),
             init.corrHLfit=list(Nugget=0.1),ranFix=list(nu=0.5))
}
if (spaMM.getOption("example_maxtime")>43) {
```

```
  fitme(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
           +Matern(1|longitude+latitude),
             data=Loaloa,family=binomial(),method="REML",
             init=list(Nugget=0.1),fixed=list(nu=0.5))
}
```

LRT *Likelihood ratio test of fixed effects.*

Description

LRT performs a likelihood ratio (LR) test between two model fits, the “full” and the “null” model fits, currently differing only in their fixed effects. Parametric bootstrap p-values can be computed, either using the raw bootstrap distribution of the likelihood ratio, or a a bootstrap estimate of the Bartlett correction of the LR statistic. This function differ from fixedLRT in its arguments (model fits for LRT, but all arguments required to fit the models for fixedLRT), and in the format of its return value. The function will stop or return possibly incorrect results for models differing beyond their fixed effects. By conceptual drift, anova works as an alias for LRT.

Usage

```
## S3 method for class 'HLfit'
anova(object,object2,...)
LRT(object,object2,boot.repl=0)
LRT(object,object2,boot.repl=0)
```

Arguments

object,object2	Two models fits being compared (their order does not matter).
boot.repl	the number of bootstrap replicates.
...	Further arguments passed to or from other methods.

Details

A raw bootstrap p-value can also be computed from the simulated distribution as (1+sum(t >= t0))/(N+1) where t0 is the original likelihood ratio, t the vector of bootstrap replicates and N its length. See Davison & Hinkley (1997, p. 141) for discusion of the adjustments in this formula. The bootstrap can also be used to provide a Bartlett correction for the likelihood ratio test in small sample. According to this correction, the mean value m of the likelihood ratio statistic under the null hypothesis is computed (here estimated by a parametric bootstrap) and the original LR statistic is multiplied by n/m where n is the number of degrees of freedom of the test.

Value

An object of class `fixedLRT`, actually a list with as-yet unstable format, but here with typical elements (depending on the options)

- `fullfit` the HLfit object for the full model;
- `nullfit` the HLfit object for the null model;
- `basicLRT` A data frame including values of the likelihood ratio statistic, its degrees of freedom, and the p-value;

and, if a bootstrap was performed:

- `rawBootLRT` A data frame including values of the likelihood ratio statistic, its degrees of freedom, and the raw bootstrap p-value;
- `BartBootLRT` A data frame including values of the Bartlett-corrected likelihood ratio statistic, its degrees of freedom, and its p-value;
- `bootInfo` a list with the following elements:

  bootreps A table of fitted likelihoods for bootstrap replicates;

  meanbootLRT The mean likelihood ratio chi-square statistic for boostrap replicates;

References

Bartlett, M. S. (1937) Properties of sufficiency and statistical tests. Proceedings of the Royal Society (London) A 160: 268-282.

Davison A.C., Hinkley D.V. (1997) Bootstrap methods and their applications. Cambridge Univ. Press, Cambridge, UK.

See Also

See also `fixedLRT`.

Examples

```
data(wafers)
## Gamma GLMM with log link
m1 <- HLfit(y ~X1+X2+X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          resid.model = ~ X3+I(X3^2) ,data=wafers,HLmethod="ML")
m2 <- update(m1,formula.= ~ . -I(X2^2))
anova(m1,m2)
```

`make_scaled_dist` *Scaled Euclidean distances between unique locations*

Description

This fonction computes scaled Euclidean distances from whichever relevant argument it can use (see Details). The result can directly by used as input for computation of the Matérn correlation matrix. It is usually called internally by HLCor, so that users may ignore it, except if they wish to control the parametrization of the scaling through `control.dist$rho.mapping`.

Usage

```
make_scaled_dist(uniqueGeo, uniqueGeo2=NULL, distMatrix, rho,
                 rho.mapping=seq_len(length(rho)),
                 dist.method="Euclidean",
                 return_matrix=FALSE)
```

Arguments

- `uniqueGeo` A matrix of geographical coordinates (e.g. 2 columns for latitude and longitude), without replicates of the same location.
- `uniqueGeo2` NULL, or a second matrix of geographical coordinates, without replicates of the same location. If NULL, scaled distances among `uniqueGeo` locations are computed. Otherwise, scaled distances between locations in the two input matrices are computed.
- `distMatrix` A distance matrix.
- `rho` A scalar or vector of positive values. Scaled distance is computed as `<distances in each coordinate>` unless a non-trivial `rho.mapping` is used.
- `rho.mapping` A set of indices controlling which elements of the `rho` scale vector scales which dimension(s) of the space in which (spatial) correlation matrices of random effects are computed. Scaled distance is generally computed as `<distances in each coordinate> * rho` As shown inthe Example, if one wishes to combine isotropic geographical distance and some environmental distance, the coordinates being latitude, longitude and one environmental variable, the scaled distance may be computed as (say) `(lat,long,env) *rho[c(1,1,2)]` so that the same scaling `rho[1]` applies for both geographical coordinates. In this case, `rho` should have length 2 and `rho.mapping` should be `c(1,1,2)`.
- `dist.method` `method` argument of `proxy::dist` function (by default, `"Euclidean"`, but e.g. `"Geodesic"` can be used to compute spherical distances.
- `return_matrix` Whether to return a `matrix` rather than a `proxy::dist` or `proxy::crossdist` object.

Details

The function uses the `distMatrix` argument if provided, in which case rho must be a scalar. Vectorial `rho` (i.e., different scaling of different dimensions) is feasible only by providing `uniqueGeo`.

Value

A matrix or dist object. If there are two input matrices, rows of the return value correspond to rows of the first matrix.

Examples

```
## a biologically not very meaningful, but syntactically correct example of rho.mapping
data(blackcap)
corrHLfit(migStatus ~ 1 + Matern(1|latitude+longitude+means),data=blackcap,
          objective="p_v",HLmethod="ML",ranFix=list(nu=0.5,phi=1e-6),
          init.corrHLfit=list(rho=c(1,1)),
          control.dist=list(rho.mapping=c(1,1,2)))
```

---

mapMM — *Colorful plots of predicted responses in two-dimensional space.*

Description

These functions provide either a map of predicted response in analyzed locations, or a predicted surface. `mapMM` is a straightforward representation of the analysis of the data, while `filled.mapMM` copes with the fact that all predictor variables may not be known in all locations on a fine spatial grid, but may involve questionable choices as a result (see `map.formula` argument). Both functions takes an `HLfit` object as input. `mapMM` calls `spaMMplot2D`, which is similar but takes a more conventional (x,y,z) input.

Usage

```
spaMMplot2D(x, y, z,xrange=range(x, finite = TRUE),
      yrange=range(y, finite = TRUE),
      margin=1/20,add.map= FALSE, nlevels = 20,
      color.palette = spaMM.colors,map.asp=NULL,
      col = color.palette(length(levels) - 1),
      plot.title=NULL, plot.axes=NULL, decorations=NULL,
      key.title=NULL, key.axes=NULL, xaxs = "i",
      yaxs = "i", las = 1, axes = TRUE, frame.plot = axes, ...)

mapMM(fitobject,Ztransf=NULL,coordinates,
      add.points,decorations=NULL,plot.title=NULL,plot.axes=NULL,envir=-3, ...)

filled.mapMM(fitobject, Ztransf = NULL, coordinates, xrange = NULL,
             yrange = NULL, margin = 1/20, map.formula, phi =
             1e-05, gridSteps = 41, decorations =
             quote(points(pred[, coordinates], cex = 1, lwd = 2)),
             add.map = FALSE, axes = TRUE, plot.title = NULL,
             plot.axes = NULL, map.asp = NULL, variance = NULL,
             var.contour.args = list(), smoothObject = NULL, ...)
```

Arguments

`fitobject` The return object of a corrHLfit call.

`x,y,z` Three vectors of coordinates, with `z` being expectedly the response.

`Ztransf` A transformation of the predicted response, given as a function whose only required argument can be a one-column matrix. The name of this argument must be `Z` (not `x`), as is appropriate for use in `do.call(Ztransf,list(Z=Zvalues))`.

`coordinates` The geographical coordinates. By default they are deduced from the model formula. For example if this formula is `resp ~ 1 + Matern(1| x + y )` the default coordinates are c("x","y"). If this formula is `resp ~ 1 + Matern(1| x + y + z )`, the user must choose two of the three coordinates.

`xrange` The `x` range of the plot (a vector of length 2); by default defined to cover all analyzed points.

`yrange` The `y` range of the plot (a vector of length 2); by default defined to cover all analyzed points.

`margin` This controls how far (in relative terms) the plot extends beyond the `x` and `y` ranges of the analyzed points, and is overriden by explicit `xrange` and `yrange` arguments.

`map.formula` Plotting a filled contour generally requires prediction in non-oberved locations, where predictor variables used in the original data analysis may be missing. In that case, the original model formula cannot be used and an alternative `map.formula` must be used to interpolate (not smooth) the predicted values in observed locations (these predictions still resulting from the original analysis based on predictor variables). As a result (1) `filled.mapMM` will be slower than a mere plotting function, since it involves the analysis of spatial data; (2) the results may have little useful meaning if the effects of the original predictor variables is not correctly represented by this interpolation step. For example, it may involve biases analogous to predicting temperature in non-oberved locations while ignoring effect of variation in altitude in such locations.

`phi` This controls the phi value assumed in the interpolation step. Ideally `phi` would be zero, but problems with numerically singular matrices may arise when `phi` is too small.

`gridSteps` The number of levels of the grid of x and y values

`variance` Either NULL, or the name of a component of prediction variance to be plotted. Must name one of the components that can be returned by `predict.HLfit`. `variance="predVar"` is suitable for uncertainty in point prediction.

`var.contour.args` A list of control parameters for rendering of prediction variances. See `contour` for possible arguments (except `x`, `y`, `z` and `add`).

`add.map` Either a boolean or an explicit expression, enclosed in `quote` (see Examples). If `TRUE`, the `map` function from the `maps` package (which much therefore the loaded) is used to add a map from its default `world` database. `xrange` and `yrange` are used to select the area, so it is most convenient if the `coordinates` are longitude and latitude (in this order and in standard units). An explicit expression can also be used for further control.

levels	a set of levels which are used to partition the range of z. Must be strictly increasing (and finite). Areas with z values between consecutive levels are painted with the same color.
nlevels	if levels is not specified, the range of z, values is divided into *approximately* this many levels (a call to pretty determines the actual number of levels).
color.palette	a color palette function to be used to assign colors in the plot.
map.asp	the y/x aspect ratio of the 2D plot area (not of the full figure including the scale). By default, the scales for x and y are identical unless the x and y ranges are too different. Namely, the scales are identical if (plotted y range)/(plotted x range) is 1/4 < . < 4, and map.asp is 1 otherwise.
col	an explicit set of colors to be used in the plot. This argument overrides any palette function specification. There should be one less color than levels
plot.title	statements which add titles to the main plot. See Details for differences between functions.
plot.axes	statements which draw axes (and a box) on the main plot. See Details for differences between functions.
decorations	Either NULL or Additional graphic statements (points, polygon, etc.), enclosed in quote (the default value illustrates the latter syntax). .
add.points	Obsolete, use decorations instead.
envir	Controls the environment in which plot.title, plot.axes, and decorations are evaluated. mapMM calls spaMM2Dplot from where these graphic arguments are evaluated, and the default value -3 means that they are evaluated within the environment from where mapMM was called.
key.title	statements which add titles for the plot key.
key.axes	statements which draw axes on the plot key.
xaxs	the x axis style. The default is to use internal labeling.
yaxs	the y axis style. The default is to use internal labeling.
las	the style of labeling to be used. The default is to use horizontal labeling.
axes, frame.plot	logicals indicating if axes and a box should be drawn, as in plot.default.
smoothObject	Either NULL, or an object inheriting from class HLfit (hence, an object on which predict.HLfit can be called), predicting the response surface in any coordinates. See Details for typical usages.
...	further arguments passed to or from other methods. For mapMM, all such arguments are passed to spaMMplot2D; for spaMMplot2D, currently only additional graphical parameters passed to title() (see Details). For filled.mapMM, these parameters are those that can be passed to spaMM.filled.contour.

Details

The smoothObject argument may be used to redraw a figure faster by recycling the predictor of the response surface returned invisibly by a previous call to filled.mapMM.

For smoothObject=NULL (the default), filled.mapMM interpolates the predicted response, with sometimes unpleasant effects. For example, if one interpolates probabilities, the result may not be within [0,1], and then (say) a logarithmic `Ztransf` may generate NaN values that would otherwise not occur. The `smoothObject` argument may be used to overcome the default behaviour, by providing an alternative predictor.

If you have values for all predictor variables in all locations of a fine spatial grid, `filled.mapMM` may not be a good choice, since it will ignore that information (see `map.formula` argument). Rather, one should use `predict(<fitobject>,newdata= <all predictor variables >)` to generate all predictions, and then either `spaMM.filled.contour` or some other raster functions.

The different functions are (currently) inconsistent among themselves in the way they handle the `plot.title` and `plot.axes` argument:

spaMM.filled.contour behaves like `graphics::filled.contour`, which (1) handles arguments which are calls such as `title(.)` or `{axis(1);axis(2)}`; (2) ignores `...` arguments if `plot.title` is missing; and (3) draws axes by default when `plot.axes` is missing, given `axes = TRUE`.

By contrast, **filled.mapMM** handles arguments which are language expressions such as produced by `quote(.)` or `substitute(.)` (see Examples).

mapMM can handles language expressions, but also accepts at least some calls.

Value

`filled.mapMM` returns invisibly a predictor of the response surface. `mapMM` has no return value. Plots are produced as side-effects.

See Also

`raster` for alternative plot procedures.

Examples

```
data(blackcap)
bfit <- corrHLfit(migStatus ~ means+ Matern(1|longitude+latitude),data=blackcap,
                  HLmethod="ML",
                  ranFix=list(lambda=0.5537,phi=1.376e-05,rho=0.0544740,nu=0.6286311))
if (require(maps)) { ## required for add.map=TRUE
  mapMM(bfit,color.palette = function(n){spaMM.colors(n,redshift=1/2)},add.map=TRUE)
}

if (spaMM.getOption("example_maxtime")>6) {
  ## filled.mapMM takes a bit longer
  # showing 'add.map', 'nlevels', and contour lines for 'variances'
  if (require(maps)) { ## required for add.map=TRUE
    filled.mapMM(bfit,nlevels=30,add.map=TRUE,plot.axes=quote({axis(1);axis(2)}),
             variance="respVar",
             plot.title=title(main="Inferred migration propensity of blackcaps",
                              xlab="longitude",ylab="latitude"))
  }
}

if (spaMM.getOption("example_maxtime")>25) {
 data(Loaloa)
 lfit <- corrHLfit(cbind(npos,ntot-npos)~elev1+elev2+elev3+elev4+maxNDVI1+seNDVI
```

```
                  +Matern(1|longitude+latitude),HLmethod="HL(0,1)",data=Loaloa,
                  family=binomial(),ranFix=list(nu=0.5,rho=2.255197,lambda=1.075))

 ## longer computation requiring interpolation of 197 points
 if (require(maps)) { ## required for add.map=TRUE
  filled.mapMM(lfit,add.map=TRUE,plot.axes=quote({axis(1);axis(2)}),
             decorations=quote(points(pred[,coordinates],pch=15,cex=0.3)),
             plot.title=title(main="Inferred prevalence, North Cameroon",
                                xlab="longitude",ylab="latitude"))
  }
}
```

MaternCorr *Matern correlation function and Matern formula term.*

Description

The Matérn correlation function describes realizations of Gaussian spatial processes with different smoothnesses (i.e. either smooth or rugged surfaces). It also includes a scaling and a 'nugget' parameter. It can be invoked in two ways. First, the `MaternCorr` function evaluates these correlations, using distances as input. Second, a term of the form `Matern(1|<...>)` in a `formula` specifies a random effect with Matérn correlation function, using coordinates found in a data frame as input. In the latter case, the correlations between realizations of the random effect for any two observations in the data will be the value of the Matérn function at the scaled Euclidean distance between coordinates specified in <...>, using "+" as separator (e.g., `Matern(1|latitude + longitude)`).

Usage

```
## Default S3 method:
MaternCorr(d, rho = 1, smoothness, nu = smoothness, Nugget = 0L)
# Matern(1|...)
```

Arguments

`d`	A distance, typically an Euclidean distance
`rho`	A scaling factor for distance. The 'range' considered in some formulations is the reciprocal of this scaling factor
`smoothness`	The smoothness parameter, >0. $\nu = 0.5$ corresponds to the exponential correlation function, and the limit function when μ goes to ∞ is the squared exponential function (as in a Gaussian).
`nu`	Same as smoothness
`Nugget`	(Following the jargon of Kriging) a parameter describing a discontinuous decrease in correlation at zero distance. Correlation will always be 1 at $d = 0$, and from which it immediately drops to (1-Nugget)
`...`	Names of coordinates, using "+" as separator (e.g., `Matern(1|latitude + longitude)`

Details

The correlation at distance $d > 0$ is

$$(1 - \text{Nugget})\frac{(\rho d)^{\nu} K_{\nu}(\rho d)}{2^{(\nu-1)}\Gamma(\nu)}$$

where K_{ν} is the besselK function of order ν.

Value

Scalar/vector/matrix depending on input.

References

Stein, M.L. (1999) Statistical Interpolation of Spatial Data: Some Theory for Kriging. Springer, New York.

See Also

See corMatern for an implementation of this correlation function as a corSpatial object for use with lme or glmmPQL.

By default the Nugget is set to 0. See one of the examples on data set Loaloa for a fit including the estimation of the Nugget.

Examples

```
## See examples in help("spaMM"), help("HLCor"), help("Loaloa"), etc.
## The Matérn function can be used in Euclidean spaces of any dimension:
set.seed(123)
randpts <- matrix(rnorm(20),nrow=5)
distMatrix <- as.matrix(proxy::dist(randpts))
MaternCorr(distMatrix,nu=2)
```

---

member_extractors — *Member functions to extract various components of a fit*

Description

These extractors are defined as member functions of a fitted object. This allows their returned value to be stored in the fitted object after the first call to the extractor. Therefore, further calls do not need to recompute the result. get_ZALMatrix extracts the design matrix for the random effects v. More member extractors are available (get_w_h_coeffs, get_beta_cov, get_beta_w_cov, get_invColdoldList, get_logdispObject, and get_info_crits) but are undocumented.

Usage

```
# object$get_ZALMatrix(object)
```

Arguments

object	The return object of a call to HLfit or similar fitting function.

Value

A matrix

Examples

```
## Not run:
 data(blackcap)
 thefit <- fitme(migStatus ~ means+ Matern(1|latitude+longitude),data=blackcap,
                   method="ML",fixed=list(nu=0.63, rho=0.054))
 thefit$get_ZALMatrix(thefit)

## End(Not run)
```

---

multinomial — *Analyzing multinomial data*

Description

These functions facilite the conversion and analysis of multinomial data as as series of nested binomial data. The main function is `multi`, to be used in the `family` argument of the fitting functions. It calls `binomialize`, which can be called directly to check how the data are converted to nested binomial data. The `fitted.HLfitlist` method of the `fitted` generic function returns a matrix of fitted multinomial probabilities. The `logLik.HLfitlist` method of the `logLik` generic function returns a log-likelihood for the joint fits.

Usage

```
multi(binResponse=c("npos","nneg"),binfamily=binomial(),input="types",...)
binomialize(data,responses,sortedTypes=NULL,binResponse=c("npos","nneg"),
              depth=Inf,input="types")
## S3 method for class 'HLfitlist'
fitted(object,...)
## S3 method for class 'HLfitlist'
logLik(object,which,...)
```

Arguments

data	The data frame to be analyzed.
object	A list of binomial fits returned by a multinomial analysis
responses	column names of the data, such that <data>[,<responses>] contain the multinomial response data, as levels of factor variables.

sortedTypes	Names of multinomial types, i.e. levels of the multinomial response factors. Their order determines which types are taken first to define the nested binomial samples. By default, the most common types are considered first.
binResponse	The names to be given to the number of “success” and “failures” in the binomial response.
depth	The maximum number of nested binomial responses to be generated from the multinomial data.
binfamily	The family applied to each binomial response.
input	If `input="types"`, then the `responses` columns must contain factor levels of the binomial response. If `input="counts"`, then the `responses` columns must contain counts of different factor levels, and the column names are the types.
which	Which element of the `APHLs` list to return. The default depends on the fitting method.In particular, if it was REML or one of its variants, the function returns the log restricted likelihood (exact or approximated).
...	Other arguments passed from or to other functions.

Details

A multinomial response, say counts 17, 13, 25, 8, 3, 1 for types `type1` to `type6` can be represented as a series of nested binomials e.g. `type1` against others (17 vs 50) then among these 50 others, `type2` versus others (13 vs 37), etc. The `binomialize` function generates such a representation. By default the representation considers types in decreasing order of the number of positives, i.e. first `type3` against others (25 vs 42), then `type1` against others within these 42, etc. It stops if it has reached `depth` nested binomial responses. This can be modified by the `sortedTypes` argument, e.g. `sortedTypes=c("type6","type4","type2")`. `binomialize` returns a list of data frames which can be directly provided as a `data` argument for the fitting functions, with binomial response.

Alternatively, one can provide the multinomial response data frame, which will be internally converted to nested binomial data if the `family` argument is a call to `multinomial` (see examples).

For mixed models, the multinomial data can be fitted to a model with the same correlation parameters, and either the same or different variances of random effects, for all binomial responses. Which analysis is performed depends on the `init.corrHLfit` argument (see `corrHLfit` and the Examples).

Value

`binomialize` returns a list of data frames appropriate for analysis as binomial response. Each data frame contains the original one plus Two columns named according to `binResponse`. `multi` returns a list.

Examples

```
## An example considering pseudo-data at one diploid locus for 50 individuals
set.seed(123)
genecopy1 <- sample(4,size=50,prob=c(1/2,1/4,1/8,1/8),replace=TRUE)
genecopy2 <- sample(4,size=50,prob=c(1/2,1/4,1/8,1/8),replace=TRUE)
alleles <- c("122","124","126","128")
genotypes <- data.frame(type1=alleles[genecopy1],type2=alleles[genecopy2])
```

```
## Columns "type1","type2" each contains an allele type => input is "types" (the default)
datalist <- binomialize(genotypes,responses=c("type1","type2"))

## two equivalent fits:
f1 <- HLfit(cbind(npos,nneg)~1,data=datalist, family=binomial())
f2 <- HLfit(cbind(npos,nneg)~1,data=genotypes, family=multi(responses=c("type1","type2")))
fitted(f2)

## distinct fits for spatial data
## Not run:
genoInSpace <- data.frame(type1=alleles[genecopy1],type2=alleles[genecopy2],x=runif(50),y=runif(50))
## Fitting distinct variances of random effects for each binomial response
corrHLfit(cbind(npos,nneg)~1+Matern(1|x+y),data=genoInSpace,
          family=multi(responses=c("type1","type2")),
          ranFix=list(rho=1,nu=0.5))
## Fitting the same variance for all binomial responses
corrHLfit(cbind(npos,nneg)~1+Matern(1|x+y),data=genoInSpace,
          family=multi(responses=c("type1","type2")),
          ranFix=list(rho=1,nu=0.5),init.corrHLfit=list(lambda=1))

## End(Not run)
```

---

negbin *Family function for negative binomial GLMs and mixed models.*

Description

Specifies the information required to fit a negative binomial generalized linear model, with known or unknown underlying Gamma shape parameter.

Usage

```
negbin(shape = stop("negbin's 'shape' must be specified"), link = "log")
```

Arguments

shape	Shape parameter of the underlying Gamma distribution, given that the negbin family can be represented as a Poisson-Gamma mixture, where the conditional Poisson mean is μ times a Gamma random variable with mean 1 and shape shape (as produced by rgamma(., shape=shape,scale=1/shape)).
link	log, sqrt or identity link, specified by the sevaral available ways for GLM links (name, character string, one-element character vector, or object of class link-glm as returned by make.link).

Details

shape is the k parameter of McCullagh and Nelder (1989, p.373) and the theta parameter of Venables and Ripley (2002, section 7.4). The negbin family is sometimes called the NegBin1 model in the literature on negative binomial models.

Value

A family object.

References

McCullagh, P. and Nelder, J.A. (1989) Generalized Linear Models, 2nd edition. London: Chapman & Hall.

Venables, W. N. and Ripley, B. D. (2002) Modern Applied Statistics with S-PLUS. Fourth Edition. Springer.

Examples

```
## Fitting negative binomial model with estimated scale parameter:
data(scotlip)
fitme(cases~I(prop.ag/10)+offset(log(scotlip$expec)),family=negbin(), data=scotlip)
```

---

options *spaMM options settings*

Description

Allow the user to set and examine a variety of *options* which affect operations of the spaMM package.

Usage

```
spaMM.options(...)

spaMM.getOption(x)
```

Arguments

`x`	a character string holding an option name.
`...`	A named value or a list of named values. The following values, with their defaults, are used in `spaMM`:

- `ff_threshold`: Minimal number of matrix elements for some matrices to be stored on disk (using the `ff` package) rather than in memory
- `COMP_maxn`: Number of terms for truncation of infinite sums that are evaluated in the fitting of `COMPoisson` models.
- `TRACE.UNLINK=FALSE`: Whether to delete preexisting files named as `HLCor.args.*.RData` when the trace option is used in corrHLfit or HLCor.
- `MESSAGES.FULL.STACK=TRUE`: Whether to give information on the stack of calls in some warning messages.

`QRmethod`: To control the methods used in intensive matrix computations. Computations may be faster in simple block random effect models when this is set to `"Matrix::qr"`.

`USEEIGEN=TRUE`: Whether to use the Eigen C++ library for some matrix computations.

`wRegularization=FALSE`: Whether to to warn about the use of regularization in some operations on nearly singular matrices.

`maxLambda=1e10`: The maximum value of lambda: higher fitted lambda values in HLfit are reduced to this.

`example_maxtime=2.5`: Used in the documentation to control whether the longer examples should be run. The approximate running time of given examples on one author's laptop is compared to this value.

and possibly other undocumented values for development purposes.

Details

Invoking `spaMM.options()` with no arguments returns a list with the current values of the options. Invoking `spaMM.getOption(<option name>)` returns the value of the option rather than a list.

spaMM.options() provides an interface for changing maximal values of parameters of the Matérn correlation function. However, it is not recommended to change these values unless a spaMM message specifically suggests so. Errors may occur if too low values are chosen as these may conflit with default initial values for the parameters.

Value

For `spaMM.getOption`, the current value set for option `x`, or `NULL` if the option is unset.

For `spaMM.options()`, a list of all set options sorted by category. For `spaMM.options(name)`, a list of length one containing the set value, or `NULL` if it is unset. For uses setting one or more options, a list with the previous values of the options changed (returned invisibly).

Examples

```
  spaMM.options()
  spaMM.getOption("example_maxtime")
  ## Not run:
  spaMM.options(maxLambda=1e06)

## End(Not run)
```

`phiHGLM` *Fitting random effects in the residual dispersion model*

Description

ϕ parameters are estimated by fitting a Gamma HGLM to response values computed by the parent fitting function (e.g., by `HLfit` in the Examples). The `fitme` function is used to perform this fit. The `resid.model` of the parent call is used to control the arguments of this `fitme` call.

Usage

```
# 'resid.model' argument of main fitting functions
```

Arguments

`resid.model` is **either** a formula (without left-hand side) for the dispersion parameter `phi` of the residual error (a log link is assumed);
or a list, with possible elements:

- `formula` — model formula as in formula-only case, without left-hand side
- `family` — The family is always Gamma. The default link is log. The identity link can be tried but may fail because only the log link ensures that the fitted ϕ is positive.
- `fixed` — fixed values of parameters. Same usage as documented in `fitme`
- `control.dist` — A list of arguments that control the computation of the distance argument of the correlation functions. Same usage as documented in `HLCor`
- `rand.family` — A `family` object or a `list` of family objects describing the distribution of the random effect(s). Same usage as documented for `HLfit` `resid.model` with random effects is still experimental and complex combinations of arguments could give unexpected results. In particular, the functionality of `init.HLfit, lower, upper, control` has not been tested. The list `should not` contain the following elements:
- `init` — Currently ignored;
- `method` — which is constrained to be identical to the method from the parent call;
- `control.HLfit, control.glm` — constrained to be identical to the same-named controls from the parent call;
- `resid.model` — (constrained: no `resid.model` for a `resid.model`);
- `REMLformula` — (constrained to NULL);
- `data` — identical to data from the parent call, which must therefore include all the variables required for the `resid.model`;
- `prior.weights` — constrained: no prior weights;
- `verbose` — constrained: will display a progress line summarizing the results of the `resid.model` fit at each iteration of main loop of the parent `HLfit` call.

References

Lee, Y., Nelder, J. A. and Pawitan, Y. (2006) Generalized linear models with random effects: unified analysis via h-likelihood. Chapman & Hall: London.

Examples

```
if (spaMM.getOption("example_maxtime")>7) {
 data("crack") # crack data, Lee et al. 2006 chapter 11 etc
 hlfit <- HLfit(y~crack0+(1|specimen),family=Gamma(log),
                data=crack, HLmethod="REML",
                rand.family=inverse.Gamma(log),
                resid.model=list(formula=~cycle+(1|specimen)) )
}
```

`plot.HLfit` *Model checking plots for mixed models*

Description

This function provides diagnostic plots for residual errors from the mean model and for random effects. Plots for the mean models are similar to those for GLMs, as described in Lee et al. 2006. Plots for residual errors consider the *standardized* deviance residuals (Lee et al. 2006, p.52), and plots for random effects likewise consider standardized values, i.e. each random deviate divided by $\sqrt(1-q)$ where q is the corresponding leverage for λ.

Usage

```
## S3 method for class 'HLfit'
plot(x, which = c("mean", "ranef"),
      titles = list(
          meanmodel=list(outer="Mean model",devres="Deviance residuals",
                    absdevres="|Deviance residuals|", resq="Residual quantiles",
                         devreshist="Deviance residuals"),
        ranef=list(outer="Random effects and leverages",qq="Random effects Q-Q plot",
                     levphi=expression(paste("Leverages for ",phi)),
          levlambda=expression(paste("Leverages for ",lambda)))
        ),
      control= list(), ...)
```

Arguments

`x`	The return object of an HLCor / HLfit / corrHLfit call.
`which`	A vector of keywords for different types of plots. By default, two types of plots are presented on different devices: diagnostic plots for mean values, and diagnostic plots for random effects. Either one can be selected using this argument. Use keyword `"predict"` for a plot of predicted response against actual response.
`titles`	A list of the `main` (inner and outer) titles of the plots. See the default value for the format.
`control`	A list of default options for the plots. Defaults are `pch="+"` and `pcol="blue"` for points, and `lcol="red"` for curves.
`...`	Options passed from `plot.HLfit` to `par`.

Details

The standardized deviance residuals are defined as the deviance residuals divided by $\phi\sqrt(1-q)$, where q is the corresponding leverage for ϕ, and the deviance residuals are defined as for a GLM. The leverages are zero for ML methods. Otherwise, they depend on the fitting method used, as defined in the Details of `HLfit`. The `PQL` and `EQL-` method use leverages obtained as diagonal elements of the “hat” matrix; more elaborate methods will introduce corrections for non-Gaussian response and for non-Gaussian random effects; and "(.,1)" methods will add another correction taking into account the variation of the GLM weights in the logdet Hessian term of restricted likelihood.

In principle the deviance residuals for the mean model should have a nearly Gaussian distribution hence form a nearly straight line on a Q-Q plot. However this is (trivially) not so for well-specified (nearly-)binary response data nor even for well-specified Poisson response data with moderate expectations. Hence this plot is not so useful.

Value

Returns the input object invisibly.

References

Lee, Y., Nelder, J. A. and Pawitan, Y. (2006). Generalized linear models with random effects: unified analysis via h-likelihood. Chapman & Hall: London.

Examples

```
## see example for data(scotlip)
```

---

predict *Prediction from a model fit.*

Description

Prediction of the response variable by its expected value obtained as (the inverse link transformation of) the linear predictor (η) and more generally for terms of the form **X**_nβ+**Z**_n**Lv**, for new design matrices **X**_n and **Z**_n. Various components of prediction variances and predictions intervals can also be computed using `predict`. The `get_`... functions are convenient extractors for such components.

Usage

```
## S3 method for class 'HLfit'
predict(object, newdata = newX, newX = NULL, re.form = NULL,
                           variances=list(), binding = FALSE, intervals = NULL,
                           level = 0.95, ...)
get_fixefVar(...)
get_predVar(...)
get_residVar(...)
get_respVar(...)
get_intervals(...)
```

Arguments

`object` The return object of fitting functions `HLfit,corrHLfit,HLCor`... returning an object inheriting from `HLfit` class.

`newdata` **Either** NULL, a matrix or data frame, or a numeric vector.

If `NULL`, the original data are reused. Otherwise, all variables required to evaluate model formulas must be included. Which variables are required may depend on other arguments: see "prediction with given phi's" example, also illustrating the syntax when formulas include an offset.

or a numeric vector, which names (if any) are ignored. This makes it easier to use `predict` as an objective function for an optimization procedure such as `optim`, which calls the objective function on unnamed vectors. However, one must make sure that the order of elements in the vector is the order of first occurrence of the variables in the model formula. This order can be checked in the error message returned when calling `predict` on a `newX` vector of clearly wrong size, e.g. `predict(<object>,newdata=numeric(0))`.

`newX` equivalent to newdata, available for back-compatibility

`re.form` formula for random effects to include. By default, it is NULL, in which case all random effects are included. If it is NA, no random effect is included. If it is a formula, only the random effects it contains are retained. The other variance components are removed from both point prediction and `variances` calculations. If you want to retain only the spatial effects in the point prediction, but all variances, either use re.form and add missing variances (on linear predictor scale) manually, or ignore this argument and see Details and Examples for different ways of controlling variances.

`variances` A list which elements control the computation of different estimated variances. In particular, `list(linPred=TRUE,disp=TRUE)` is suitable for uncertainty in point prediction.

`predict` can return four components of prediction variance: `fixefVar`, `predVar`, `residVar` and `respVar`, detailed below. They are all returned as attributes of the point predictions. By default, each component is a vector of variances. However, if `variances$cov=TRUE`, a covariance matrix is returned when applicable (i.e. not for `"residVar"`).

`fixefVar` is the (co)variance of fixed effects ($\mathbf{X}\beta$) due to uncertainty in β. It is called by `variances$fixefVar=TRUE`.

`predVar` is the (co)variance of the linear predictor η. It is the most common measure of uncertainty in point prediction. It accounts for uncertainty in fixed effects ($\mathbf{X}\beta$) and random effects ($\mathbf{ZLv}$) for given dispersion parameters (see Details), but it can also accounts for uncertainty in dispersion parameters (λ and phi) estimates if `variances$disp=TRUE`, for models in which the effect of uncertainty in dispersion parameters can be computed. Currently, this effect can be computed for a scalar residual variance (ϕ) and a single random effect with a scalar variance (λ). `variances$predVar=TRUE` will return the sum of the two components, if available; otherwise it returns only the (co)variance for given λ and phi. The latter component can be requested by `variances$linPred=TRUE`.

`residVar` provides the residual variances (for Gaussian or Gamma responses). It is called by `variances$residVar=TRUE`.

	`respVar` is the variance of the response, i.e. the sum of `predVar` and `residVar`. It is called by `variances$respVar=TRUE`. Calling for one (co)variance implies that some of its components may be also returned.
`intervals`	NULL or character string or vector of strings. Provides prediction intervals with nominal level `level`, deduced from the given prediction variance term, e.g. `intervals="predVar"`. Currently only intervals from `fixefVar` and `predVar` (and for LMMs `respVar` including the residual variance) may have a probabilistic meaning. Intervals returned in other cases are (currently) meaningless.
`level`	Coverage of the intervals.
`binding`	If `binding` is a character string, the predicted values are bound with the `newdata` and the result is returned as a data frame. The predicted values column name is the given `binding`, or a name based on it, if the `newdata` already include a variable with this name). If `binding` is `FALSE`, The predicted values are returned as a matrix and the data frame used for prediction is returned as an attribute (unless it was `NULL`).
`...`	further arguments passed to or from other methods. For the `get_`... functions, they are passed to `predict`.

Details

If `newdata` is NULL, `predict` returns the fitted responses, including random effects, from the object. Otherwise it computes new predictions including random effects as far as possible. For spatial random effects it constructs a correlation matrix **C** between new locations and locations in the original fit. Then it infers the random effects in the new locations as **C** $(\mathbf{L}')^{-1}$ **v** (see `spaMM` for notation). For non-spatial random effects, it checks whether any group (i.e., level of a random effect) in the new data was represented in the original data, and it adds the inferred random effect for this group to the prediction for individuals in this group.

`fixefVar` is the (co)variance of $\mathbf{X}\beta$ (or **X**_nβ), deduced from the asymptotic covariance matrix of β estimates.

`predVar` is the prediction (co)variance of η=$\mathbf{X}\beta$+**Zv** (see `HLfit` Details for notation), or more generally of **X**_nβ+**Z**_n**Lv**, by default computed for given dispersion parameters.

For levels of the random effects present in the original data, `predVar` computation

takes into account the joint uncertainty in estimation of β and prediction of **v**.

For new levels of the random effects, `predVar` computation additionally takes into account uncertainty in prediction of **v** for these new levels. For **prediction covariance** with a new **Z**_n, it matters whether a single or multiple new levels are used: see Examples.

If `variances$disp` is `TRUE`, prediction variance may also include a term accounting for uncertainty in ϕ and λ, computed following Booth and Hobert (1998, eq. 19). This computation is currently implemented for models with a single random effect, and ignore uncertainties in spatial correlation parameters.

For models with non-Gaussian response, the prediction covariance of the response is approximated by the prediction covariance of the linear predictor, pre- and post-multiplied by $\partial\mu/\partial\eta$.

These variance calculations are approximate except for LMMs, and cannot be garanteed to give accurate results.

In the **point prediction** of the linear predictor, the unconditional expected value of u is assigned to the realizations of u for unobserved levels of non-spatial random effects (it is zero in GLMMs but not for non-gaussian random effects), and the inferred value of u is assigned in all other cases. Corresponding values of v are then deduced. This computation yields the classical "BLUP" or empirical Bayes predictor in LMMs, but otherwise it may yield less well characterized predictors, where "unconditional" v may not be its expected value when the `rand.family` link is not identity.

Intervals computations use the relevant variance estimates plugged in a Gaussian approximation, except for the simple linear model where it uses Student's *t* distribution.

Value

For `predict`, a matrix or data frame (according to the `binding` argument), with optional attributes `frame`, `intervals`, `predVar`, `fixefVar`, `residVar`, and/or `respVar`, the last four holding one or more variance vector or covariance matrices. The further attribute `fittedName` contains the binding name, if any.

The `get_`... extractor functions call `predict` and extract from its result the attribute implied by the name of the extractor. By default, `get_intervals` will return prediction intervals using `predVar`.

References

Booth, J.G., Hobert, J.P. (1998) Standard errors of prediction in generalized linear mixed models. J. Am. Stat. Assoc. 93: 262-272.

Examples

```
data(blackcap)
fitobject <- corrHLfit(migStatus ~ 1 + Matern(1|latitude+longitude),data=blackcap,
                       ranFix=list(nu=4,rho=0.4,phi=0.05))
predict(fitobject)
getDistMat(fitobject)

#### multiple controls of prediction variances
## (1) fit with an additional random effect
grouped <- cbind(blackcap,grp=c(rep(1,7),rep(2,7)))
fitobject <- corrHLfit(migStatus ~ 1 +  (1|grp) +Matern(1|latitude+longitude),
                       data=grouped,  ranFix=list(nu=4,rho=0.4,phi=0.05))

## (2) re.form usage to remove a random effect from point prediction and variances:
predict(fitobject,re.form= ~ 1 +  Matern(1|latitude+longitude))

## (3) comparison of covariance matrices for two types of new data
moregroups <- grouped[1:5,]
rownames(moregroups) <- paste("newloc",1:5,sep="")
moregroups$grp <- rep(3,5) ## all new data belong to an unobserved third group
cov1 <- get_predVar(fitobject,newdata=moregroups,
                     variances=list(linPred=TRUE,cov=TRUE))
moregroups$grp <- 3:7 ## all new data belong to distinct unobserved groups
cov2 <- get_predVar(fitobject,newdata=moregroups,
                     variances=list(linPred=TRUE,cov=TRUE))
cov1-cov2 ## the expected off-diagonal covariance due to the common group in the first fit.
```

```
## Not run:
## prediction with distinct given phi's in different locations:
varphi <- cbind(blackcap,logphi=runif(14))
vphifit <- corrHLfit(migStatus ~ 1 + Matern(1|latitude+longitude),
                     resid.model = list(formula=~0+offset(logphi)),
                     data=varphi,  ranFix=list(nu=4,rho=0.4))
# for respVar computation, one needs the resid.model formula to specify phi:
get_respVar(vphifit,newdata=data.frame(latitude=1,longitude=1,logphi=1))
# for predVar computation, phi is not needed
#     (and could have been specified through ranFix):
get_predVar(vphifit,newdata=data.frame(latitude=1,longitude=1))

## Effects of numerically singular correlation matrix C:
fitobject <- corrHLfit(migStatus ~ 1 + Matern(1|latitude+longitude),data=blackcap,
                       ranFix=list(nu=10,rho=0.001)) ## numerically singular C
predict(fitobject) ## predicted mu computed as X beta + L v
predict(fitobject,newdata=blackcap) ## predicted mu computed as X beta + C

## point predictions and variances with new X and Z
if(require("rsae", quietly = TRUE)) {
  data(landsat)
  fitobject <- HLfit(HACorn ~ PixelsCorn + PixelsSoybeans + (1|CountyName),
                     data=landsat[-33,],HLmethod="ML")
  newXandZ <- unique(data.frame(PixelsCorn=landsat$MeanPixelsCorn,
                                PixelsSoybeans=landsat$MeanPixelsSoybeans,
                                CountyName=landsat$CountyName))
  predict(fitobject,newdata=newXandZ,variances = list(predVar=TRUE))
  get_predVar(fitobject,newdata=newXandZ,variances = list(predVar=TRUE))
}


## End(Not run)
```

Predictor *Interface for model formulas*

Description

`Predictor(...)` performs some minimal syntax checking, and returns a formula with attributes. It serves as a unified interface for the set of descriptors for a linear predictor, including design matrices for random effects.

In the current version this function could be ignored by users.

Usage

```
Predictor(formula, offset=NULL, LMatrix = NULL, AMatrix = NULL, ZALMatrix = NULL)
```

Arguments

`formula`	a `formula`, which can include fixed effects, random effects, and offsets.
`offset`	a `offset` can be provided in this way, as a numeric vector. However, it **may be better** to provide the offset as an offset formula term (see `scotlip` example), in particular for later use with `predict` where the formula can be reevaluated on new data.
`LMatrix`	The "square root" of the correlation matrix between unique locations, see Details.
`AMatrix`	A matrix that relates observed (unique) locations to unobserved locations, see Details.
`ZALMatrix`	The design matrix for random effects, see Details.

Details

In a spatial model a vector of correlated random effects **Lv** can be constructed from uncorrelated ones, **v**, for some matrix **L** (this may be meaningful only for Gaussian random effects). Typically **L** is the Cholesky "square root" of a correlation matrix determined by the random effect specification (e.g., `Matern(...)`), or given as the `corrMatrix` argument of `HLCor`.

If there is one realized random effect per response value, the linear predictor contains **Lv**, where **L** is a square matrix which dimension is the number of observations.

Several observations may be taken in the same location, and a matrix **Z** (usually automatically constructed) tells which element of **Lv** affects each observation. The linear predictor then contains **ZLv**, where `dim(Z)` is (number of observations,number of locations).

Finally, in some applications the realized random effects in response locations may be viewed as linear combinations **ALv** of random effects **Lv** in distinct locations. In that case the dimension of **L** is the number of such distinct locations, **A** maps them to the observed locations, and **Z** again maps them to possibly repeated observations in observed locations.

Thus, in general the random term in the linear predictor is written **Mv**, where **M=ZAL** is reconstructed from the element matrices (usually automatically constructed if needed), unless **ZAL** is given as argument.

Value

A formula with attributes. This return object has classes `formula` and `predictor`.

Examples

```
# In the current version this function can be ignored by users,
# so examples are not required.
# (Use of AMatrix could perhaps be shown)
```

raster	*Raster plots of predicted responses*

Description

This documentation provides an example of producing a map out of spatial predictions, using rasters and geographical projections. No attempt has been made to provide a function automating the whole process, but the example can be used as a template.

Examples

```
if (spaMM.getOption("example_maxtime")>10) {
## using the Bonne projection for the fit *and* the plots

if(require("rgdal", quietly = TRUE) && require("rasterVis", quietly = TRUE)) {
  data(blackcap)

  ## fit:
  projection <- "+proj=bonne +lat_1=30n +lon_0=10e +units=km"
  bonne <- as.data.frame(project(as.matrix(blackcap[,c("longitude","latitude")]), projection))
  colnames(bonne) <- c("bonnex","bonney")
  bc <- cbind(blackcap,bonne)
  bfit <- corrHLfit(migStatus ~ 1 + Matern(1|bonnex+bonney), data=bc,
                    HLmethod="ML")

  ## plots:

  xrange <- range(bc$bonnex)
  yrange <- range(bc$bonney)
  gridSteps <- 101
  border <- 20
  xGrid <- seq(xrange[1]-border, xrange[2]+border, length.out = gridSteps)
  yGrid <- seq(yrange[1]-border, yrange[2]+border, length.out = gridSteps)
  newdata <- expand.grid(bonnex=xGrid, bonney=yGrid)
  gridpred <- predict(bfit, newdata = newdata)

  palette <- spaMM.colors()

  CreateRaster <- function(
    long,
    lat,
    values,
    proj="+proj=longlat +datum=WGS84",
    save.spatial.files=FALSE,
    filename="data_raster",
    overwrite.spatial.files=TRUE
  ) {
    # Args:
    #   long: a vector of the longitudes of the raster cells
    #   lat: a vector of the latitudes of the raster cells
```

```
  #   values: a vector of the values of the raster cells
  #   proj: the projection system for the raster
  #   save.spatial.files: logical indicating if
  #          an hard copy of the raster should be saved (as ascii)
  #   filename: name of the file for the hard copy
  #   overwrite.spatial.files: logical indicating if
  #          an existing hard copy should be overwritten or not
  #
  # Returns:
  #   The raster.
  #
  data <- data.frame(long=long, lat=lat, values=values)  # a dataframe
  #                 with longitudes, lattitudes, and values is being created
  coordinates(data) <- ~long+lat  # coordinates are being set for the raster
  proj4string(data) <- CRS(proj)  # projection is being set for the raster
  gridded(data) <- TRUE  # a gridded structure is being set for the raster
  data.raster <- raster(data)  # the raster is being created
  if(save.spatial.files) writeRaster(
    data.raster,
    filename=paste(filename, ".asc", sep=""),
    overwrite=overwrite.spatial.files
  )  # if save=TRUE the raster is exported as an ascii file
  return(data.raster) # the raster is being returned
}

raster.predict <- CreateRaster(
  long=newdata$bonnex,
  lat=newdata$bonney,
  values=gridpred,
  proj=projection)

data(worldcountries)
data(oceanmask)
# All shape files can be found here: http://www.naturalearthdata.com/downloads/
# and loaded into R by e.g.
# oceanmask <- readOGR("ne_110m_ocean.shp", layer="ne_110m_ocean")

## crop the ocean mask before projecting it, otherwise artefacts will appear
# (1) project world
worldcountries <- spTransform(worldcountries, projection)
# (2) define extent of region to be cropped in projected world
extent.crop <- extent(projectExtent(raster.predict, oceanmask@proj4string))+20
# (3) crop oceanmask according to this extent
oceanmask <- crop(oceanmask, extent.crop)
oceanmask <- spTransform(oceanmask, projection)
plot(oceanmask)


p <- levelplot( ## filled contour with legend bar
  raster.predict, ## levelplot from rasterVis handles any raster
  maxpixels=4e6,
  margin=FALSE,
  cuts=length(palette)-1,
```

```
  col.regions=palette
)


country.layer <- layer(
  sp.polygons(worldcountries, fill=fill),
  data=list(sp.polygons=sp.polygons, worldcountries=worldcountries,
            fill="transparent")
)

p+country.layer

CreateSpatialPoints <-
  function(
    long,
    lat,
    values=-9999,
    proj="+proj=longlat +datum=WGS84",
    save.spatial.files=FALSE,
    filename="my_points",
    overwrite.spatial.files=TRUE
  ) {
    data.sp <- data.frame(long=long, lat=lat, values=values)
    coordinates(data.sp) <- ~long+lat
    proj4string(data.sp) <- CRS(proj)
    if(save.spatial.files)   writeOGR(
      data.sp,
      dsn="./",
      layer=filename,
      morphToESRI=TRUE,
      driver="ESRI Shapefile",
      overwrite_layer=overwrite.spatial.files
    )
    return(data.sp)
  }

my.points <- CreateSpatialPoints( ## userdefined
  long=bc$bonnex,
  lat=bc$bonney,
  proj=projection
)

points.layer <- layer(
  sp.points(my.points, col=col, cex=cex, pch=pch, lwd=lwd),
  data=list(sp.points=sp.points, my.points=my.points,
            col="white", cex=2, pch=15, lwd=2)
)

p+points.layer

ocean.layer <- layer(
  sp.polygons(oceanmask, fill=fill, col=col),
  data=list(sp.polygons=sp.polygons, oceanmask=oceanmask,
```

```
                fill="black", col="transparent")
    )
    p+ocean.layer+country.layer+points.layer

  }

  }
```

salamander *Salamander mating data*

Description

Data from a salamander mating experiment discussed by McCullagh and Nelder (1989, Ch. 14). Twenty males and twenty females from two populations (Rough Butt and Whiteside) were each paired with 6 individuals from their own or from the other population. The experiments were later published by Arnold et al. (1996).

Usage

```
data(salamander)
```

Format

The data frame includes 360 observations on the following variables:

Female Index of the female;

Male Index of the male;

Mate Whether the pair successfully mated or not;

TypeF Population of origin of female;

TypeM Population of origin of male;

Cross Interaction term between `TypeF` and `TypeM`;

Season A factor with levels `Summer` and `Fall`;

Experiment Index of experiment

Source

The data frame was borrowed from the `HGLMMM` package (Molas and Lesaffre, 2011), version 0.1.2.

References

Arnold, S.J., Verrell, P.A., and Tilley S.G. (1996) The evolution of asymmetry in sexual isolation: a model and a test case. Evolution 50, 1024-1033.

McCullagh, P. and Nelder, J.A. (1989). Generalized Linear Models, 2nd edition. London: Chapman & Hall.

Molas, M., Lesaffre, E. (2011) Hierarchical Generalized Linear Models: The R Package HGLMMM. Journal of Statistical Software 39, 1-20.

Examples

```
data(salamander)

if (spaMM.getOption("example_maxtime")>3) {
 HLfit(cbind(Mate,1-Mate)~TypeF+TypeM+TypeF*TypeM+(1|Female)+(1|Male),
    family=binomial(),data=salamander,HLmethod="ML")
}
# In this example fitme() is slower
if (spaMM.getOption("example_maxtime")>7) {
 fitme(cbind(Mate,1-Mate)~TypeF+TypeM+TypeF*TypeM+(1|Female)+(1|Male),
    family=binomial(),data=salamander)
}
```

---

scotlip *Lip cancer in Scotland 1975 - 1980*

Description

This data set provides counts of lip cancer diagnoses made in Scottish districts from 1975 to 1980, and additional information relative to these data from Clayton and Kaldor (1987) and Breslow and Clayton (1993). The data set contains (for each district) counts of disease events and estimates of the fraction of the population involved in outdoor industry (agriculture, fishing, and forestry) which exposes it to sunlight.

`data("scotlip")` actually loads a data frame, `scotlip`, and an adjacency matrix, `Nmatrix`, between 56 Scottish districts, as given by Clayton and Kaldor (1987, Table 1).

Usage

```
data(scotlip)
```

Format

The data frame includes 56 observations on the following 7 variables:

gridcode alternative district identifier.

id numeric district identifier (1 to 56).

district district name.

cases number of lip cancer cases diagnosed 1975 - 1980.

population total person years at risk 1975 - 1980.

prop.ag percent of the population engaged in outdoor industry.

expec offsets considered by Breslow and Clayton (1993, Table 6, 'Exp' variable)

The rows are ordered according to `gridcode`, so that they match the rows of `Nmatrix`.

References

Clayton D, Kaldor J (1987). Empirical Bayes estimates of age-standardized relative risks for use in disease mapping. Biometrics, 43: 671 - 681.

Breslow, NE, Clayton, DG. (1993). Approximate Inference in Generalized Linear Mixed Models. Journal of the American Statistical Association: 88 9-25.

Examples

```
data(scotlip)
scfit <- HLCor(cases~I(prop.ag/10) +adjacency(1|gridcode)+offset(log(scotlip$expec)),
      ranPars=list(rho=0.174),adjMatrix=Nmatrix,family=poisson(),data=scotlip)
summary(scfit)
plot(scfit)
```

`seaMask` *Masks of seas or lands*

Description

These convenient masks can be added to maps of (parts of) the world to mask map information for these areas.

Usage

```
data(seaMask)
data(landMask)
data(worldcountries)
data(oceanmask)
```

Format

`seaMask` and `landMask` are data frames with two variables, `x` and `y` for longitude and latitude. Its contents are suitable for use with `polypath`: they define different polygones, each separated by a row of `NA`s.

`worldcountries` and `oceanmask` are `SpatialPolygonsDataFrame` objects.

Details

A land mask can be produced out of `worldcountries` by filling the countries (i.e. `fill="black"` in the code for `country.layer` in the Examples of `raster`).

`worldcountries` and `oceanmask` were created from public domain shapefiles downloaded from `www.naturalearth.com` on 2015/10/21. These are suitable for plots involving geographical projections not available through `map`, and more generally for raster plots. Only the lowest-resolution data are included in spaMM, to minimize the size of the package archive, but higher-resolution files are available on `www.naturalearth.com`, from where they can be loaded as shown in the examples. `worldcountries` had to be edited for non-ASCII characters before inclusion in spaMM:

worldcountries@data$formal_fr was removed and the “Côte d’Ivoire” level of some factor variables was renamed.

seaMask and landMask were created from the world map in the maps package. polypath requires polygons, while map(interior=FALSE,plot=FALSE) returns small segments. landMask is the result of reconnecting the segments into full coastlines of all land blocks.

See Also

raster for uses of worldcountries and oceanmask

Examples

```
if (spaMM.getOption("example_maxtime")>2.5) {

data(seaMask)
## plot of predictions of behaviour for a land bird:
if (require(maps)){
 data(blackcap)
 bfit <- corrHLfit(migStatus ~ means+ Matern(1|longitude+latitude),data=blackcap,
                  HLmethod="ML",
                  ranFix=list(lambda=0.5537,phi=1.376e-05,rho=0.0544740,nu=0.6286311))
 ## We add small masks to the points on small islands to see the predictions there
 ll <- blackcap[,c("longitude","latitude")]
 pointmask <- function(xy,r=1,npts=12) {
  theta <- 2*pi/npts *seq(npts)
  hexas <- lapply(seq(nrow(xy)),function(li){
    p <- as.numeric(xy[li,])
    hexa <- cbind(x=p[1]+r*cos(theta),y=p[2]+r*sin(theta))
    rbind(rep(NA,2),hexa) ## inital NA before each polygon
  })
  do.call(rbind,hexas)
 }
 pmasks <- pointmask(ll[c(2,4,5,6,7),],r=0.8) ## small islands only
 filled.mapMM(bfit,add.map=TRUE,
             plot.title=title(main="Inferred migration propensity of blackcaps",
                              xlab="longitude",ylab="latitude"),
             decorations=quote(points(pred[,coordinates],cex=1,pch="+")),
             plot.axes=quote({axis(1);axis(2);
                        polypath(rbind(seaMask,pmasks),border=FALSE,
                                 col="grey", rule="evenodd")
             }))
}
}

## Not run:
# All shape files can be found here: http://www.naturalearthdata.com/downloads/
# Once downloaded, they can be loaded into R by
if (require("rgdal", quietly = TRUE)) {
  worldcountries <- readOGR("ne_110m_admin_0_countries_lakes.shp",
                           layer="ne_110m_admin_0_countries_lakes")
}
```

```
## End(Not run)
```

---

seeds *Seed germination data*

Description

A classic toy data set, "from research conducted by microbiologist Dr P. Whitney of Surrey University. A batch of tiny seeds is brushed onto a plate covered with a certain extract at a given dilution. The numbers of germinated and ungerminated seeds are subsequently counted" (Crowder, 1978). Two seed types and two extracts are here considered in a 2x2 factorial design.

Usage

```
data(seeds)
```

Format

The data frame includes 21 observations on the following variables:

plate Factor for replication;

seed Seed type, a factor with two levels O73 and O75;

extract Root extract, a factor with two levels Bean and Cucumber;

r Number of seeds that germinated;

n Total number of seeds tested

Source

Crowder (1978), Table 3.

References

Crowder, M.J., 1978. Beta-binomial anova for proportions. Appl. Statist., 27, 34-37.

Y. Lee and J. A. Nelder. 1996. Hierarchical generalized linear models (with discussion). J. R. Statist. Soc. B, 58: 619-678.

Examples

```
data(seeds)
## An extended quasi-likelihood (EQL) fit as considered by Lee and Nelder (1996):
HLfit(cbind(r,n-r)~seed*extract+(1|plate),family=binomial(),
  rand.family=Beta(),
  HLmethod="HL(0,0)",
  data=seeds)
```

simulate.HLfit *Simulate realizations of a fitted mixed model.*

Description

From an HLfit object, `simulate.HLfit` function generates new samples given the estimated fixed effects and dispersion parameters.

This does not yet work for new locations in a mixed model involving both spatial and non-spatial random effects.

Usage

```
## S3 method for class 'HLfit'
simulate(object, nsim = 1, seed = NULL, newdata=NULL,
                         conditional=FALSE, verbose=TRUE,
                         sizes=object$weights,...)
## S3 method for class 'HLfitlist'
simulate(object, nsim = 1, seed = NULL,
                       newdata=object[[1]]$data, sizes=object[[1]]$weights,...)
```

Arguments

`object`	The return object of HLfit or similar function.
`nsim`	number of response vectors to simulate. Defaults to '1'.
`seed`	A seed for `set.seed`. If such a value is provided, the initial state of the random number generator at a global level is restored on exit from simulate.
`newdata`	A data frame closely matching the original data, except that response values are not needed. May provide new values of fixed predictor variables, new spatial locations, or new individuals within a block.
`conditional`	Boolean, for development purposes. Currently only conditional simulation is handled.
`verbose`	Boolean; whether to print some information or not.
`sizes`	A vector of sample sizes to simulate in the case of a binomial fit. Defaults to the sizes in the original data.
`...`	further arguments passed to or from other methods.

Value

For the `HLfitlist` method (i.e., the result of a multinomial fit), a list of simulated responses. Otherwise, a vector (if nsim=1) or a matrix with nsim columns, each containing a simulated response.

Examples

```
data(Loaloa)
HLC <- HLCor(cbind(npos,ntot-npos)~Matern(1|longitude+latitude),
           data=Loaloa,family=binomial(),
           ranPars=list(lambda=1,nu=0.5,rho=1/0.7))
simulate(HLC,nsim=2)
```

---

spaMM *Inference in mixed models, in particular spatial GLMMs*

Description

Fits a range of mixed models, including those with spatially correlated random effects. The random effects are either Gaussian (which defines GLMMs), or other distributions (which defines the wider class of hierarchical GLMs), or simply absent (which makes a GLM).

Details

The standard (G)LMM response families `gaussian`, `binomial`, `poisson`, and `Gamma` are handled, as well as `multinomial` response (with some constraints), negative binomial (see `negbin`) and Conway-Maxwell-Poisson response (see `COMPoisson`). GLMMs and HGLMs are fit via Laplace approximations for (1) the marginal likelihood with respect to random effects and (2) the restricted likelihood (as in REML), i.e the likelihood of random effect parameters given the fixed effect estimates.

The package fits models including several nested or crossed random effects, only one of which can be a spatial random effect. The spaMM code has not been optimized for speed in non-spatial models. It fits random effects following either a Matérn correlation model (see `Matern`) or an adjacency matrix model (see `adjacency`).

The variance(s) of random effects (u) is (are) denoted λ (`lambda` in input and output). The variance parameter of residual error is denoted ϕ (`phi`): this is the residual variance for gaussian response, but for Gamma-distributed response, the residual variance is $\phi\mu^2$ where μ is expected response. A fixed-effects linear predictor for ϕ, modeling heteroscedasticity, can be considered (see Examples). Fixed effects are described in the standard form $\mathbf{X}\beta$ where $\mathbf{X}$ is the design matrix of fixed effects and β (`beta`) is a vector of fixed effect parameters.

The structure of the random effects can generally be described by the following steps. First, independent and identically distributed (iid) random effects u are drawn from one of the following distributions: gaussian, Beta-distributed, Gamma and inverse-Gamma distributed random effects, implemented as detailed in the `HLfit` documentation. Second, a transformation $\mathbf{v}= f(\mathbf{u})$ is applied ($\mathbf{v}$ elements are still iid). Third, correlated random effects are obtained as $\mathbf{Mv}$, where the matrix $\mathbf{M}$ can describe spatial correlation between observed locations, block effects (or repeated observations in given locations), and correlations involving unobserved locations. See Details in `Predictor` for the general form of $\mathbf{M}$. In most cases $\mathbf{M}$ is determined from the model formula, but it can also be input directly (e.g., to describe genetic correlations).

The package has been extensively tested mainly for analysis of spatial GLMMs (Rousset and Ferdy 2014 and ongoing work), where the random effects are Gaussian. Other models have been checked against literature results and a few simulations.

Author(s)

François Rousset and Jean-Baptiste Ferdy.

The syntax of formula terms mostly matches the one in the lme4 package, so bits of code for parsing formulas has been recycled in spaMM from various versions of lme4.

References

Lee, Y., Nelder, J. A. and Pawitan, Y. (2006). Generalized linear models with random effects: unified analysis via h-likelihood. Chapman & Hall: London.

Rousset F., Ferdy, J.-B. (2014) Testing environmental and genetic effects in the presence of spatial autocorrelation. Ecography, 37: 781-790. http://dx.doi.org/10.1111/ecog.00566

See Also

spaMM is designed to be used through the high-level functions corrHLfit, HLCor, HLfit, and fixedLRT

Examples

```
## Fit a Poisson GLMM with adjacency (CAR) correlation model
# see ?adjacency for how to fit efficiently such model models
data(scotlip) ## loads 'scotlip' data frame, but also 'Nmatrix'
HLCor(cases~I(prop.ag/10) +adjacency(1|gridcode)+offset(log(scotlip$expec)),
          adjMatrix=Nmatrix,family=poisson(),data=scotlip)


if (spaMM.getOption("example_maxtime")>2) {
 ## Adding a Gamma random effect to fit a negative-binomial response:
 HLCor(cases~I(prop.ag/10) +(1|gridcode)+adjacency(1|gridcode)
                +offset(log(scotlip$expec)),
          data=scotlip,family=poisson(),rand.family=list(Gamma(log),gaussian()),
          adjMatrix=Nmatrix)
}

if (spaMM.getOption("example_maxtime")>4) {
 ## fit non-spatial crossed random effects with distinct families
 data(salamander)
 HLfit(cbind(Mate,1-Mate)~1+(1|Female)+(1|Male),family=binomial(),
        rand.family=list(gaussian(),Beta(logit)),data=salamander,HLmethod="ML")
}

 ## Nested effects

if (spaMM.getOption("example_maxtime")>28) {
 # lmer syntax allowing several degrees of nesting
 HLfit(cbind(Mate,1-Mate)~1+(1|Female/Male),
       family=binomial(),rand.family=Beta(logit),data=salamander,HLmethod="ML")
 # [ also allowed is cbind(Mate,1-Mate)~1+(1|Female)+(1|Male %in% Female) ]
}

## fit a non-spatial, Gamma GLMM:
```

```
data(wafers)
HLfit(y ~X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          data=wafers)

## Same with fixed-effects predictor for residual variance
## ( = structured-dispersion model):
HLfit(y ~X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          resid.model = ~ X3+I(X3^2) ,data=wafers)

## Random-slope model (mind the output!)
if (spaMM.getOption("example_maxtime")>2) {
  HLfit(y~X1+(X2|batch),data=wafers)
}

## fit a GLM (not mixed) with structured dispersion:
HLfit( y ~X1+X2+X1*X3+X2*X3+I(X2^2),family=Gamma(log),
      resid.model = ~ X3+I(X3^2) ,data=wafers)

## Fit of binary data using PQL/L. See ?arabidopsis
## Not run:
data(arabidopsis)
HLCor(cbind(pos1046738,1-pos1046738)~seasonal+Matern(1|LAT+LONG),
                   ranPars=list(rho=0.129,lambda=4.28,nu=0.291),
                   family=binomial(),HLmethod="PQL/L",data=arabidopsis)

## End(Not run)
```

---

spaMM-conventions *Difference between spaMM and common R practices*

Description

spaMM differs in the following ways from practices one may be used to:

input arguments are generally similar to those of glm and (g)lmer, in particular for the spaMM::fitme function, with the exception of the prior.weights argument, which is simply weights in the other packages. prior.weights is in a sense more consistent, since e.g. glm returns its input weights a output prior.weights, while its output weights are the distinct GLM weights.

The default **likelihood target** for dispersion parameters is restricted likelihood (REML estimation) for corrHLfit and (marginal) likelihood (ML estimation) for fitme.

Most basic fitting functions in R will seek variables in the global environment if they are not in the data. This easily leads to errors (see example in the discussion of update.HLfit) and spaMM tries to restrain this.

Computation times depend on control.HLfit$conv.threshold, which is by default 1e-5. Increasing it to 1e-4 had no notable effect (08/2016) on the tests in the tests directory of the package, except reducing computation time. spaMM is therefore slower but safer than it could be.

spaMM.colors *A flashy color palette.*

Description

spaMM.colors is the default color palette for some color plots in spaMM.

Usage

```
spaMM.colors(n = 64, redshift = 1, adjustcolor_args=NULL)
```

Arguments

n	Number of color levels returned by the function. A calling graphic function with argument nlevels will typically take the first (i.e., bluest) nlevels color levels. If n<nlevels, the color levels are recycled
redshift	The higher it is, the more the palette blushes....
adjustcolor_args	Either NULL or a list of arguments for adjustcolor, in which case adjustcolor is called to modify spaMM.colors's default vector of colors. See the documentation of the latter function for further information. All arguments except col are possible.

Details

If you don't like this color palette, have a look at the various ones provided by the fields package.

Value

A vector giving the colors in a hexadecimal format.

Examples

```
## see mapMM examples
```

spaMM.filled.contour *Level (Contour) Plots with better aspect ratio control (for geographical maps, at least)*

Description

This function is derived from `filled.contour` in the `graphics` package, and this documentation is likewise heavily based on that of `filled.contour`.

This function likewise produces a contour plot with the areas between the contours filled in solid color, and a key showing how the colors map to z values is likewise shown to the right of the plot. The only difference is the way the aspect ratio is determined and can be controlled (using the `map.asp` parameter instead of `asp`), They thus easily provide nice-looking maps with meaningful latitude/longitude ratio (see Examples). However, this does not work well with rstudio.

Usage

```
spaMM.filled.contour(x = seq(0, 1, length.out = nrow(z)),
               y = seq(0, 1, length.out = ncol(z)),
               z,
               xrange = range(x, finite = TRUE),
               yrange = range(y, finite = TRUE),
               zrange = range(z, finite = TRUE),
               margin=1/20,
               levels = pretty(zrange, nlevels), nlevels = 20,
               color.palette = spaMM.colors,
               col = color.palette(length(levels) - 1),
               plot.title, plot.axes, key.title=NULL, key.axes=NULL,
               map.asp = NULL, xaxs = "i", yaxs = "i", las = 1,
               axes = TRUE, frame.plot = axes, ...)
```

Arguments

`x, y`	locations of grid lines at which the values in `z` are measured. These must be in ascending order. (The rest of this description does not apply to `.filled.contour`.) By default, equally spaced values from 0 to 1 are used. If `x` is a `list`, its components `x$x` and `x$y` are used for `x` and `y`, respectively. If the list has component `z` this is used for `z`.
`z`	a numeric matrix containing the values to be plotted.. Note that `x` can be used instead of `z` for convenience.
`xrange`	x range of the plot.
`yrange`	y range of the plot.
`zrange`	z range of the plot.
`margin`	This controls how far (in relative terms) the plot extends beyond the `x` and `y` ranges of the analyzed points, and is overriden by explicit `xrange` and `yrange` arguments.
`levels`	a set of levels which are used to partition the range of `z`. Must be **strictly** increasing (and finite). Areas with `z` values between consecutive levels are painted with the same color.
`nlevels`	if `levels` is not specified, the range of `z`, values is divided into approximately this many levels.

color.palette	a color palette function to be used to assign colors in the plot.
col	an explicit set of colors to be used in the plot. This argument overrides any palette function specification. There should be one less color than levels
plot.title	statements which add titles to the main plot.
plot.axes	statements which draw axes (and a box) on the main plot. This overrides the default axes.
key.title	statements which add titles for the plot key.
key.axes	statements which draw axes on the plot key. This overrides the default axis.
map.asp	the y/x aspect ratio of the 2D plot area (not of the full figure including the scale). Default is (plotted y range)/(plotted x range) (i.e., scales for x are identical).
xaxs	the x axis style. The default is to use internal labeling.
yaxs	the y axis style. The default is to use internal labeling.
las	the style of labeling to be used. The default is to use horizontal labeling.
axes, frame.plot	logicals indicating if axes and a box should be drawn, as in plot.default.
...	additional graphical parameters, currently only passed to title().

Details

The values to be plotted can contain NAs. Rectangles with two or more corner values are NA are omitted entirely: where there is a single NA value the triangle opposite the NA is omitted.

Values to be plotted can be infinite: the effect is similar to that described for NA values.

Note

Builds heavily on filled.contour by Ross Ihaka and R-core. spaMM.filled.contour uses the layout function and so is restricted to a full page display.

The output produced by spaMM.filled.contour is actually a combination of two plots; one is the filled contour and one is the legend. Two separate coordinate systems are set up for these two plots, but they are only used internally – once the function has returned these coordinate systems are lost. If you want to annotate the main contour plot, for example to add points, you can specify graphics commands in the plot.axes argument. See the Examples.

References

Cleveland, W. S. (1993) *Visualizing Data.* Summit, New Jersey: Hobart.

See Also

contour, image, palette; contourplot and levelplot from package lattice.

Examples

```
spaMM.filled.contour(volcano, color = spaMM.colors) # simple

## Comparing the layout with that of filled.contour:
#  (except that it does not always achieve the intended effect
#  in RStudio Plots pane).

x <- 10*1:nrow(volcano)
y <- 10*1:ncol(volcano)
spaMM.filled.contour(x, y, volcano, color = terrain.colors,
    plot.title = title(main = "The Topography of Maunga Whau",
    xlab = "Meters North", ylab = "Meters West"),
    plot.axes = { axis(1, seq(100, 800, by = 100))
                  axis(2, seq(100, 600, by = 100)) },
    key.title = title(main = "Height\n(meters)"),
    key.axes = axis(4, seq(90, 190, by = 10)))  # maybe also asp = 1
mtext(paste("spaMM.filled.contour(.) from", R.version.string),
      side = 1, line = 4, adj = 1, cex = .66)

## compare with

filled.contour(x, y, volcano, color = terrain.colors,
    plot.title = title(main = "The Topography of Maunga Whau",
    xlab = "Meters North", ylab = "Meters West"),
    plot.axes = { axis(1, seq(100, 800, by = 100))
                  axis(2, seq(100, 600, by = 100)) },
    key.title = title(main = "Height\n(meters)"),
    key.axes = axis(4, seq(90, 190, by = 10)))  # maybe also asp = 1
mtext(paste("filled.contour(.) from", R.version.string),
      side = 1, line = 4, adj = 1, cex = .66)
```

---

spaMM_glm.fit — *Fitting generalized linear models without initial-value or divergence headaches*

Description

spaMM_glm.fit is a stand-in replacement for glm.fit, which can be called through glm by using glm(<>, method="spaMM_glm.fit"). Input and output structure are exactly as for glm.fit. It uses a Levenberg-Marquardt to prevent divergence of estimates. If the rcdd package is installed, the function can automatically find valid starting values or else indicate that no parameter value is feasible. spaMM_glm is a convenient wrapper, calling glm with default method glm.fit, then calling method spaMM_glm.fit if glm.fit failed.

Usage

```
spaMM_glm.fit(x, y, weights = rep(1, nobs), start = NULL, etastart = NULL,
        mustart = NULL, offset = rep(0, nobs), family = gaussian(),
```

```
        control = list(maxit=200), intercept = TRUE)
spaMM_glm(formula, family = gaussian, data, weights, subset,
                na.action, start = NULL, etastart, mustart, offset,
                control = list(...), model = TRUE, method = "glm.fit",
                x = FALSE, y = TRUE, contrasts = NULL, strict=FALSE, ...)
```

Arguments

All arguments except strict are common to these functions and their stats package equivalents, glm and glm.fit. Most arguments operate as for the latter functions, whose documentation is repeated below. The control argument may operate differently.

formula: an object of class "formula" (or one that can be coerced to that class): a symbolic description of the model to be fitted. The details of model specification are given in the 'Details' section of glm.

family: a description of the error distribution and link function to be used in the model. For spaMM_glm this can be a character string naming a family function, a family function or the result of a call to a family function. For spaMM_glm.fit only the third option is supported. (See family for details of family functions.)

data: an optional data frame, list or environment (or object coercible by as.data.frame to a data frame) containing the variables in the model. If not found in data, the variables are taken from environment(formula), typically the environment from which glm is called.

weights: an optional vector of 'prior weights' to be used in the fitting process. Should be NULL or a numeric vector.

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

na.action: a function which indicates what should happen when the data contain NAs. The default is set by the na.action setting of options, and is na.fail if that is unset. The 'factory-fresh' default is na.omit. Another possible value is NULL, no action. Value na.exclude can be useful.

start: starting values for the parameters in the linear predictor.

etastart: starting values for the linear predictor.

mustart: starting values for the vector of means.

offset: this can be used to specify an *a priori* known component to be included in the linear predictor during fitting. This should be NULL or a numeric vector of length equal to the number of cases. One or more offset terms can be included in the formula instead or as well, and if more than one is specified their sum is used. See model.offset.

control: a list of parameters for controlling the fitting process. This is passed to glm.control, as for glm.fit. Because one can assume that spaMM_glm.fit will converge in many cases where glm.fit does not, spaMM_glm.fit allows more iterations (200) by default. However, if spaMM_glm.fit is called through glm(. . ., method="spaMM_glm.fit") then the number of iterations is controlled by the glm.control call within glm, so that it is 25 by default, overriding the spaMM_glm.fit default.

model — a logical value indicating whether *model frame* should be included as a component of the returned value.

method — the method to be used by spaMM_glm in the firt attempt at fitting the model (spaMM_glm.fit being always used in a second attempt if the first failed). The default method "glm.fit" uses iteratively reweighted least squares (IWLS): the alternative "model.frame" returns the model frame and does no fitting.

User-supplied fitting functions can be supplied either as a function or a character string naming a function, with a function which takes the same arguments as glm.fit. If specified as a character string it is looked up from within the **stats** namespace.

x, y — For spaMM_glm: x is a design matrix of dimension n * p, and y is a vector of observations of length n.

For spaMM_glm.fit: x is a design matrix of dimension n * p, and y is a vector of observations of length n.

contrasts — an optional list. See the contrasts.arg of model.matrix.default.

intercept — logical. Should an intercept be included in the *null* model?

strict — logical. Whether to perform a fit by spaMM_glm.fit if glm.fit returned the warning "glm.fit: algorithm did not converge".

... — arguments to be used to form the default control argument if it is not supplied directly.

Value

An object inheriting from class glm. See glm for details.

Note

The source and documentation is derived in large part from those of glm.fit.

Examples

```
# glm() failures checked in R-devel version (2016-03-21 r70361)
x <- c(8.752,20.27,24.71,32.88,27.27,19.09)
y <- c(5254,35.92,84.14,641.8,1.21,47.2)
## Not run:  # fails:
  glm(y~ x,data=data.frame(x,y),family=Gamma(log))

## End(Not run)
spaMM_glm(y~ x,data=data.frame(x,y),family=Gamma(log))

## Gamma(inverse) examples
x <- c(43.6,46.5,21.7,18.6,17.3,16.7)
y <- c(2420,708,39.6,16.7,46.7,10.8)
## Not run:  # fails (can't find starting value)
  glm(y~ x,data=data.frame(x,y),family=Gamma())

## End(Not run)
if (require("rcdd",quietly=TRUE)) {
  spaMM_glm(y~ x,data=data.frame(x,y),family=Gamma())
```

```
}
```

summary.HLfit *Summary and print methods for fit and test results.*

Description

Summary and print methods for results from HLfit or related functions.

Usage

```
## S3 method for class 'HLfit'
summary(object,...)
## S3 method for class 'HLfitlist'
summary(object,...)
## S3 method for class 'fixedLRT'
summary(object,verbose=TRUE,...)
## S3 method for class 'HLfit'
print(x,...)
## S3 method for class 'HLfitlist'
print(x,...)
## S3 method for class 'fixedLRT'
print(x,...)
```

Arguments

object	The return object of HLfit or related functions.
x	The return object of HLfit or related functions.
verbose	for `summary.fixedLRT`, whether to print the model fits or not.
...	further arguments passed to or from other methods.

Value

These methods return the object invisibly. They print details of the (lower level) HLfit results in a convenient form.

Examples

```
## see examples of corrHLfit usage
```

`update.HLfit` *Updates an HLCor or HLfit fit*

Description

`update` will update and (by default) re-fit a model. It does this mostly by extracting the call stored in the object, updating the call and evaluating that call. (however, currently the `predictor` argument is processed differently). Using `update` is a risky programming style (see Note).

Usage

```
## S3 method for class 'HLfit'
update(object, formula., ..., evaluate = TRUE)
```

Arguments

`object`	A return object from an HLfit call.
`formula.`	Changes to the formula. Beware of the syntax: see *update.formula* for details.
`...`	Additional arguments to the call, or arguments with changed values. Use name = NULL to remove the argument name.
`evaluate`	If TRUE, evaluate the new call else return the call.

Value

An HLCor or HLfit fit of the same type as the input object.

Note

`update`, as a general rule, is tricky. For example with `m1 <- glm(y~x + offset(<...>),<...>); m2 <- update(m1,.~.-x` `m2` will lose the intercept term, which would be retained in the absence of an offset. `update.HLfit` has the same behaviour. Further, `update` is easily affected in a non-transparent way by changes in variables used in the original call. For example `foo <- rep(1,10) m <- lm(rnorm(10)~1, weights=foo) rm(foo) update(m, .~.) # Error` To avoid such problems, `spaMM` tries to avoid references to variables in the global environment, by enforcing that the data are explicitly provided by the `data` argument, and that prior weights, if not constant, are in the data.

See Also

See also `HLCor`, `HLfit`.

Examples

```
data(wafers)
## First the fit to be updated:
wFit <- HLfit(y ~X1*X3+X2*X3+I(X2^2)+(1|batch),family=Gamma(log),
          resid.model = ~ X3+I(X3^2) ,data=wafers)

# For estimates given by Lee et al., Appl. Stochastic Models Bus. Ind. (2011) 27:  315-328:
# Refit with given beta or/and phi values:

betavals <- c(5.55,0.08,-0.14,-0.21,-0.08,-0.09,-0.09)
# reconstruct fitted phi value from predictor for log(phi)
Xphi <- with(wafers,cbind(1,X3,X3^2)) ## design matrix
phifit <- exp(Xphi %*% c(-2.90,0.1,0.95))
update(wFit,formula.= . ~ offset(wFit$`X.pv` %*% betavals)+(1|batch),
       ranFix=list(lambda=exp(-3.67),phi=phifit))

## There are subtlety in performing REML fits of constrained models,
##   illustrated by thefact thatthe following fir does not recover
##   the original likelihood values, because dispersion parameters are
##   estimatedbut the REML correction changes with the formula:
update(wFit,formula.= . ~ offset(wFit$`X.pv` %*% fixef(wFit))+(1|batch))
## To maintain the original REML correction, Consider instead
update(wFit,formula.= . ~ offset(wFit$`X.pv` %*% fixef(wFit))+(1|batch),
       REMLformula=wFit$predictor)  ## recover orginal p_v and p_bv
## Alternatively, show original wFit as differences from betavals:
update(wFit,formula.= . ~ . +offset(wFit$`X.pv` %*% betavals))
```

---

wafers *Data from a resistivity experiment for semiconductor materials.*

Description

This data set was reported and analyzed by Robinson et al. (2006) and reanalyzed by Lee et al. (2011). The data "deal with wafers in a single etching process in semiconductor manufacturing. Wafers vary through time since there are some variables that are not perfectly controllable in the etching process. For this reason, wafers produced on any given day (batch) may be different from those produced on another day (batch). To measure variation over batch, wafers are tested by choosing several days at random. In this data, resistivity is the response of interest. There are three variables, gas flow rate (x1), temperature (x2), and pressure (x3) and one random effect (batch or day)." (Lee et al 2011).

Usage

```
data(wafers)
```

Format

The data frame includes 198 observations on the following variables:

y resistivity.

batch batch, indeed.

X1 gas flow rate.

X2 temperature.

X3 pressure.

Source

This data set was manually pasted from Table 3 of Lee et al. (2011). Transcription errors may have occurred.

References

Robinson TJ, Wulff SS, Montgomery DC, Khuri AI. 2006. Robust parameter design using generalized linear mixed models. Journal of Quality Technology 38: 38–65.

Lee, Y., Nelder, J.A., and Park, H. 2011. HGLMs for quality improvement. Applied Stochastic Models in Business and Industry 27, 315-328.

Examples

```
## see examples in the main Documentation page for the package.
```

`welding` *Welding data set*

Description

The data give the results of an unreplicated experiment for factors affecting welding quality conducted by the National Railway Corporation of Japan (Taguchi and Wu, 1980, cited in Smyth et al., 2001). It is a toy example for heterocedastic models and is also suitable for illustrating fit of overparameterized models.

Usage

```
data("welding")
```

Format

The data frame includes 16 observations on 10 variables:

Strengh response variable;

... nine two-level factors.

Source

The data were downloaded from http://www.statsci.org/data/general/welding.txt on 2014/08/19 and are consistent with those shown in table 5 of Bergman and Hynén (1997).

References

Bergman B, Hynén A (1997) Dispersion effects from unreplicated designs in the 2^{k-p} series. Technometrics, 39, 191–98.

Smyth GK, Huele AF, Verbyla AP (2001). Exact and approximate REML for heteroscedastic regression. Statistical Modelling 1, 161-175.

Taguchi G, Wu Y (1980) Introduction to off-line quality control. Nagoya, Japan: Central Japan Quality Control Association.

Examples

```
data(welding)
## toy example from Smyth et al.
HLfit(Strength ~ Drying + Material,resid.model = ~ Material+Preheating ,data=welding)
## toy example of overparameterized model
HLfit(Strength ~ Rods+Thickness*Angle+(1|Rods),resid.model = ~ Rods+Thickness*Angle ,data=welding)
```

Index