Package 'glmmTMB'

July 2, 2020

Title Generalized Linear Mixed Models using Template Model Builder

Version 1.0.2.1

Description Fit linear and generalized linear mixed models with various extensions, including zero-inflation. The models are fitted using maximum likelihood estimation via 'TMB' (Template Model Builder). Random effects are assumed to be Gaussian on the scale of the linear predictor and are integrated out using the Laplace approximation. Gradients are calculated using automatic differentiation.

License AGPL-3

Depends R (>= 3.2.0)

Imports methods, TMB (>= 1.7.14), lme4 (>= 1.1-18.9000), Matrix, nlme

LinkingTo TMB, RcppEigen

Suggests knitr, rmarkdown, testthat, MASS, lattice, ggplot2 (>= 2.2.1), mlmRev, bbmle (>= 1.0.19), pscl, coda, reshape2, car (>= 3.0.6), emmeans (>= 1.4), estimability, DHARMa, multcomp, MuMIn, effects (>= 4.0-1), dotwhisker, broom, broom.mixed, plyr, png, boot, texreg, xtable, huxtable, numDeriv

SystemRequirements GNU make

VignetteBuilder knitr

URL https://github.com/glmmTMB/glmmTMB

LazyData TRUE

BugReports https://github.com/glmmTMB/glmmTMB/issues

RoxygenNote 7.1.0

NeedsCompilation yes

Encoding UTF-8

Author Arni Magnusson [aut] (<https://orcid.org/0000-0003-2769-6741>), Hans Skaug [aut], Anders Nielsen [aut] (<https://orcid.org/0000-0001-9683-9262>), Casper Berg [aut] (<https://orcid.org/0000-0002-3812-5269>), Kasper Kristensen [aut],

Martin Maechler [aut] (<https://orcid.org/0000-0002-8685-9910>),
Koen van Bentham [aut],
Ben Bolker [aut] (<https://orcid.org/0000-0002-2127-0443>),
Nafis Sadat [ctb] (<https://orcid.org/0000-0001-5715-616X>),
Daniel Lüdecke [ctb] (<https://orcid.org/0000-0002-8895-3206>),
Russ Lenth [ctb],
Joseph O'Brien [ctb] (<https://orcid.org/0000-0001-9851-5077>),
Mollie Brooks [aut, cre] (<https://orcid.org/0000-0001-6963-8326>)

Maintainer Mollie Brooks <mollieebrooks@gmail.com>

Repository CRAN

Date/Publication 2020-07-02 11:30:17 UTC

R topics documented:

`Anova.glmmTMB` *Downstream methods*

Description

Methods have been written that allow `glmmTMB` objects to be used with several downstream packages that enable different forms of inference. For some methods (`Anova` and `emmeans`, but *not* `effects` at present), set the `component` argument to "cond" (conditional, the default), "zi" (zero-inflation) or "disp" (dispersion) in order to produce results for the corresponding part of a `glmmTMB` model.

In particular,

- `car::Anova` constructs type-II and type-III Anova tables for the fixed effect parameters of any component
- the `emmeans` package computes estimated marginal means (previously known as least-squares means) for the fixed effects of any component
- the `effects` package computes graphical tabular effect displays (only for the fixed effects of the conditional component)

Usage

```
Anova.glmmTMB(
  mod,
  type = c("II", "III", 2, 3),
  test.statistic = c("Chisq", "F"),
  component = "cond",
  vcov. = vcov(mod)[[component]],
  singular.ok,
  ...
)

Effect.glmmTMB(focal.predictors, mod, ...)
```

Arguments

`mod`	a glmmTMB model
`type`	type of test, `"II"`, `"III"`, `2`, or `3`. Roman numerals are equivalent to the corresponding Arabic numerals. See `Anova` for details.
`test.statistic`	unused: only valid choice is "Chisq" (i.e., Wald chi-squared test)
`component`	which component of the model to test/analyze ("cond", "zi", or "disp")
`vcov.`	variance-covariance matrix (usually extracted automatically)
`singular.ok`	OK to do ANOVA with singular models (unused) ?
`...`	Additional parameters that may be supported by the method.
`focal.predictors`	a character vector of one or more predictors in the model in any order.

Details

While the examples below are disabled for earlier versions of R, they may still work; it may be necessary to refer to private versions of methods, e.g. `glmmTMB:::Anova.glmmTMB(model,...)`.

Examples

```
warp.lm <- glmmTMB(breaks ~ wool * tension, data = warpbreaks)
salamander1 <- readRDS(system.file("example_files","salamander1.rds",package="glmmTMB"))
if (require(emmeans)) {
    emmeans(warp.lm, poly ~ tension | wool)
    emmeans(salamander1, ~ mined, type="response")
    emmeans(salamander1, ~ mined, component="zi", type="response")
}
if (getRversion() >= "3.6.0") {
   if (require(car)) {
       Anova(warp.lm,type="III")
       Anova(salamander1)
       Anova(salamander1, component="zi")
   }
   if (require(effects)) {
       plot(allEffects(warp.lm))
       plot(allEffects(salamander1))
   }
}
```

`confint.glmmTMB` *Calculate confidence intervals*

Description

Calculate confidence intervals

Usage

```
## S3 method for class 'glmmTMB'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = c("wald", "Wald", "profile", "uniroot"),
  component = c("all", "cond", "zi", "other"),
  estimate = TRUE,
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("profile.ncpus", 1L),
  cl = NULL,
  full = FALSE,
  ...
)
```

Arguments

`object`	`glmmTMB` fitted object.
`parm`	which parameters to profile, specified • by index (position) [*after* component selection for `confint`, if any] • by name (matching the row/column names of `vcov(object,full=TRUE)`) • as `"theta_"` (random-effects variance-covariance parameters), `"beta_"` (conditional and zero-inflation parameters), or `"disp_"` or `"sigma"` (dispersion parameters) Parameter indexing by number may give unusual results when some parameters have been fixed using the `map` argument: please report surprises to the package maintainers.
`level`	Confidence level.
`method`	'wald', 'profile', or 'uniroot': see Details function)
`component`	Which of the three components 'cond', 'zi' or 'other' to select. Default is to select 'all'.
`estimate`	(logical) add a third column with estimate ?
`parallel`	method (if any) for parallel computation
`ncpus`	number of CPUs/cores to use for parallel computation
`cl`	cluster to use for parallel computation
`full`	CIs for all parameters (including dispersion) ?
`...`	arguments may be passed to `profile.merMod` or `tmbroot`

Details

Available methods are

"wald" These intervals are based on the standard errors calculated for parameters on the scale of their internal parameterization depending on the family. Derived quantities such as standard deviation parameters and dispersion parameters are back-transformed. It follows that confidence intervals for these derived quantities are typically asymmetric.

"profile" This method computes a likelihood profile for the specified parameter(s) using `profile.glmmTMB`; fits a spline function to each half of the profile; and inverts the function to find the specified confidence interval.

"uniroot" This method uses the `uniroot` function to find critical values of one-dimensional profile functions for each specified parameter.

At present, "wald" returns confidence intervals for variance parameters on the standard deviation/correlation scale, while "profile" and "uniroot" report them on the underlying ("theta") scale: for each random effect, the first set of parameter values are standard deviations on the log scale, while remaining parameters represent correlations on the scaled Cholesky scale (see the

Examples

```
data(sleepstudy, package="lme4")
model <- glmmTMB(Reaction ~ Days + (1|Subject), sleepstudy)
model2 <- glmmTMB(Reaction ~ Days + (1|Subject), sleepstudy,
    dispformula= ~I(Days>8))
confint(model)  ## Wald/delta-method CIs
confint(model,parm="theta_")  ## Wald/delta-method CIs
confint(model,parm=1,method="profile")
```

`epil2` *Seizure Counts for Epileptics - Extended*

Description

Extended version of the `epil` dataset of the **MASS** package. The three transformed variables `Visit`, `Base`, and `Age` used by Booth et al. (2003) have been added to `epil`.

Usage

```
epil2
```

Format

A data frame with 236 observations on the following 12 variables:

`y` an integer vector.

`trt` a factor with levels `"placebo"` and `"progabide"`.

`base` an integer vector.

`age` an integer vector.

`V4` an integer vector.

`subject` an integer vector.

`period` an integer vector.

`lbase` a numeric vector.

`lage` a numeric vector.

Visit `(rep(1:4,59) -2.5) / 5`.

Base `log(base/4)`.

Age `log(age)`.

References

Booth, J.G., G. Casella, H. Friedl, and J.P. Hobert. (2003) Negative binomial loglinear mixed models. *Statistical Modelling* **3**, 179–191.

Examples

```
epil2$subject <- factor(epil2$subject)
op <- options(digits=3)
(fm <- glmmTMB(y ~ Base*trt + Age + Visit + (Visit|subject),
               data=epil2, family=nbinom2))
meths <- methods(class = class(fm))
if((Rv <- getRversion()) > "3.1.3") {
    (funs <- attr(meths, "info")[, "generic"])
    for(F in funs[is.na(match(funs, "getME"))]) {
        cat(sprintf("%s:\n-----\n", F))
        r <- tryCatch( get(F)(fm), error=identity)
        if (inherits(r, "error")) cat("** Error:", r$message,"\n")
        else tryCatch( print(r) )
        cat(sprintf("---end{%s}--------------\n\n", F))
    }
}
options(op)
```

`findReTrmClasses` *list of specials – taken from enum.R*

Description

list of specials – taken from enum.R

Usage

```
findReTrmClasses()
```

`fitTMB` *Optimize a TMB model and package results*

Description

This function (called internally by `glmmTMB`) runs the actual model optimization, after all of the appropriate structures have been set up. It can be useful to run `glmmTMB` with `doFit=TRUE`, adjust the components as required, and then finish the fitting process with `fitTMB` (however, it is the user's responsibility to make sure that any modifications create an internally consistent final fitted object).

Usage

```
fitTMB(TMBStruc)
```

Arguments

TMBStruc	a list contain

Examples

```
m0 <- glmmTMB(count ~ mined + (1|site),
              family=poisson, data=Salamanders, doFit=FALSE)
names(m0)
fitTMB(m0)
```

---

fixef *Extract fixed-effects estimates*

Description

Extract Fixed Effects

Usage

```
## S3 method for class 'glmmTMB'
fixef(object, ...)
```

Arguments

object	any fitted model object from which fixed effects estimates can be extracted.
...	optional additional arguments. Currently none are used in any methods.

Details

Extract fixed effects from a fitted glmmTMB model.

The print method for fixef.glmmTMB object *only displays non-trivial components*: in particular, the dispersion parameter estimate is not printed for models with a single (intercept) dispersion parameter (see examples)

Value

an object of class fixef.glmmTMB comprising a list of components (cond, zi, disp), each containing a (possibly zero-length) numeric vector of coefficients

Examples

```
data(sleepstudy, package = "lme4")
fm1 <- glmmTMB(Reaction ~ Days, sleepstudy)
(f1 <- fixef(fm1))
f1$cond
## show full coefficients, including dispersion parameter
unlist(f1)
print.default(f1)
```

formatVC *Format the 'VarCorr' Matrix of Random Effects*

Description

"format()" the 'VarCorr' matrix of the random effects – for print()ing and show()ing

Usage

```
formatVC(
  varcor,
  digits = max(3, getOption("digits") - 2),
  comp = "Std.Dev.",
  formatter = format,
  useScale = attr(varcor, "useSc"),
  ...
)
```

Arguments

varcor	a VarCorr (-like) matrix with attributes.
digits	the number of significant digits.
comp	character vector of length one or two indicating which columns out of "Variance" and "Std.Dev." should be shown in the formatted output.
formatter	the function to be used for formatting the standard deviations and or variances (but *not* the correlations which (currently) are always formatted as "0.nnn"
useScale	whether to report a scale parameter (e.g. residual standard deviation)
...	optional arguments for formatter(*) in addition to the first (numeric vector) and digits.

Value

a character matrix of formatted VarCorr entries from varc.

formula.glmmTMB *Extract the formula of a glmmTMB object*

Description

Extract the formula of a glmmTMB object

Usage

```
## S3 method for class 'glmmTMB'
formula(x, fixed.only = FALSE, component = c("cond", "zi", "disp"), ...)
```

Arguments

x	a glmmTMB object
fixed.only	(logical) drop random effects, returning only the fixed-effect component of the formula?
component	formula for which component of the model to return (conditional, zero-inflation, or dispersion)
...	unused, for generic consistency

---

getCapabilities *List model options that glmmTMB knows about*

Description

List model options that glmmTMB knows about

Usage

```
getCapabilities(what = "all", check = FALSE)
```

Arguments

what	(character) which type of model structure to report on ("all","family","link","covstruct")
check	(logical) do brute-force checking to test whether families are really implemented (only available for what="family")

Value

if check==FALSE, returns a vector of the names (or a list of name vectors) of allowable entries; if check==TRUE, returns a logical vector of working families

Note

these are all the options that are *defined* internally; they have not necessarily all been *implemented* (FIXME!)

`getME.glmmTMB` *Extract or Get Generalize Components from a Fitted Mixed Effects Model*

Description

Extract or Get Generalize Components from a Fitted Mixed Effects Model

Usage

```
## S3 method for class 'glmmTMB'
getME(object, name = c("X", "Xzi", "Z", "Zzi", "Xd", "theta", "beta"), ...)
```

Arguments

`object`	a fitted `glmmTMB` object
`name`	of the component to be retrieved
`...`	ignored, for method compatibility

See Also

`getME` Get generic and re-export:

`getReStruc` *Calculate random effect structure Calculates number of random effects, number of parameters, block size and number of blocks. Mostly for internal use.*

Description

Calculate random effect structure Calculates number of random effects, number of parameters, block size and number of blocks. Mostly for internal use.

Usage

```
getReStruc(reTrms, ss = NULL)
```

Arguments

`reTrms`	random-effects terms list
`ss`	a character string indicating a valid covariance structure. Must be one of `names(glmmTMB:::.valid_covs` default is to use an unstructured variance-covariance matrix (`"us"`) for all blocks).

Value

a list

blockNumTheta	number of variance covariance parameters per term
blockSize	size (dimension) of one block
blockReps	number of times the blocks are repeated (levels)
covCode	structure code

Examples

```
data(sleepstudy, package="lme4")
rt <- lme4::lFormula(Reaction~Days+(1|Subject)+(0+Days|Subject),
                     sleepstudy)$reTrms
rt2 <- lme4::lFormula(Reaction~Days+(Days|Subject),
                     sleepstudy)$reTrms
getReStruc(rt)
```

---

getXReTrms *Create X and random effect terms from formula*

Description

Create X and random effect terms from formula

Usage

```
getXReTrms(formula, mf, fr, ranOK = TRUE, type = "", contrasts, sparse = FALSE)
```

Arguments

formula	current formula, containing both fixed & random effects
mf	matched call
fr	full model frame
ranOK	random effects allowed here?
type	label for model type
contrasts	a list of contrasts (see ?glmmTMB)
sparse	(logical) return sparse model matrix?

Value

a list composed of

X	design matrix for fixed effects
Z	design matrix for random effects
reTrms	output from mkReTrms from **lme4**
offset	offset vector, or vector of zeros if offset not specified

get_cor *translate vector of correlation parameters to correlation values*

Description

translate vector of correlation parameters to correlation values

Usage

```
get_cor(theta)
```

Arguments

theta	vector of internal correlation parameters

Details

This function follows the definition at http://kaskr.github.io/adcomp/classUNSTRUCTURED_CORR__t.html: if L is the lower-triangular matrix with 1 on the diagonal and the correlation parameters in the lower triangle, then the correlation matrix is defined as $\Sigma = D^{-1/2}LL^\top D^{-1/2}$, where $D = \mathrm{diag}(LL^\top)$. For a single correlation parameter θ_0, this works out to $\rho = \theta_0/\sqrt{1+\theta_0^2}$. The function returns the elements of the lower triangle of the correlation matrix, in column-major order.

Value

a vector of correlation values

Examples

```
th0 <- 0.5
stopifnot(all.equal(get_cor(th0),th0/sqrt(1+th0^2)))
get_cor(c(0.5,0.2,0.5))
```

glmmTMB *Fit Models with TMB*

Description

Fit a generalized linear mixed model (GLMM) using Template Model Builder (TMB).

Usage

```
glmmTMB(
  formula,
  data = NULL,
  family = gaussian(),
  ziformula = ~0,
  dispformula = ~1,
  weights = NULL,
  offset = NULL,
  contrasts = NULL,
  na.action = na.fail,
  se = TRUE,
  verbose = FALSE,
  doFit = TRUE,
  control = glmmTMBControl(),
  REML = FALSE,
  start = NULL,
  map = NULL,
  sparseX = NULL
)
```

Arguments

formula	combined fixed and random effects formula, following lme4 syntax.
data	optional data frame containing model variables.
family	a family function, a character string naming a family function, or the result of a call to a family function (variance/link function) information. See family for a generic discussion of families or family_glmmTMB for details of glmmTMB-specific families.
ziformula	a *one-sided* (i.e., no response variable) formula for zero-inflation combining fixed and random effects: the default ~0 specifies no zero-inflation. Specifying ~. sets the zero-inflation formula identical to the right-hand side of formula (i.e., the conditional effects formula); terms can also be added or subtracted. **When using ~. as the zero-inflation formula in models where the conditional effects formula contains an offset term, the offset term will automatically be dropped**. The zero-inflation model uses a logit link.
dispformula	a *one-sided* formula for dispersion containing only fixed effects: the default ~1 specifies the standard dispersion given any family. The argument is ignored for families that do not have a dispersion parameter. For an explanation of the dispersion parameter for each family, see sigma. The dispersion model uses a log link. In Gaussian mixed models, dispformula=~0 fixes the residual variance to be 0 (actually a small non-zero value: at present it is set to sqrt(.Machine$double.eps)), forcing variance into the random effects.
weights	weights, as in glm. Not automatically scaled to have sum 1.
offset	offset for conditional model (only).
contrasts	an optional list, e.g., list(fac1="contr.sum"). See the contrasts.arg of model.matrix.default.

na.action	how to handle missing values, see na.action and model.frame. From lm: "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."
se	whether to return standard errors.
verbose	whether progress indication should be printed to the console.
doFit	whether to fit the full model, or (if FALSE) return the preprocessed data and parameter objects, without fitting the model.
control	control parameters, see glmmTMBControl.
REML	whether to use REML estimation rather than maximum likelihood.
start	starting values, expressed as a list with possible components beta, betazi, betad (fixed-effect parameters for conditional, zero-inflation, dispersion models); b, bzi (conditional modes for conditional and zero-inflation models); theta, thetazi (random-effect parameters, on the standard deviation/Cholesky scale, for conditional and z-i models); thetaf (extra family parameters, e.g., shape for Tweedie models).
map	a list specifying which parameter values should be fixed to a constant value rather than estimated. map should be a named list containing factors corresponding to a subset of the internal parameter names (see start parameter). Distinct factor values are fitted as separate parameter values, NA values are held fixed: e.g., map=list(beta=factor(c(1,2,3,NA))) would fit the first three fixed-effect parameters of the conditional model and fix the fourth parameter to its starting value. In general, users will probably want to use start to specify non-default starting values for fixed parameters. See MakeADFun for more details.
sparseX	a named logical vector containing (possibly) elements named "cond", "zi", "disp" to indicate whether fixed-effect model matrices for particular model components should be generated as sparse matrices, e.g. c(cond=TRUE). Default is all FALSE

Details

Binomial models with more than one trial (i.e., not binary/Bernoulli) can either be specified in the form prob ~ ...,weights = N, or in the more typical two-column matrix cbind(successes,failures)~... form.

Behavior of REML=TRUE for Gaussian responses matches lme4::lmer. It may also be useful in some cases with non-Gaussian responses (Millar 2011). Simulations should be done first to verify.

Because the df.residual method for glmmTMB currently counts the dispersion parameter, one would need to multiply by sqrt(nobs(fit) / (1+df.residual(fit))) when comparing with lm.

By default, vector-valued random effects are fitted with unstructured (general positive definite) variance-covariance matrices. Structured variance-covariance matrices can be specified in the form struc(terms|group), where struc is one of

- diag (diagonal, heterogeneous variance)
- ar1 (autoregressive order-1, homogeneous variance)
- cs (compound symmetric, heterogeneous variance)
- ou (* Ornstein-Uhlenbeck, homogeneous variance)
- exp (* exponential autocorrelation)

- gau (* Gaussian autocorrelation)
- mat (* Matérn process correlation)
- toep (* Toeplitz)

Structures marked with * are experimental/untested.

For backward compatibility, the family argument can also be specified as a list comprising the name of the distribution and the link function (e.g. list(family="binomial",link="logit")). However, **this alternative is now deprecated**; it produces a warning and will be removed at some point in the future. Furthermore, certain capabilities such as Pearson residuals or predictions on the data scale will only be possible if components such as variance and linkfun are present, see family.

Note

For more information about the **glmmTMB** package, see Brooks et al. (2017) and the vignette(package="glmmTMB") collection. For the underlying **TMB** package that performs the model estimation, see Kristensen et al. (2016).

References

Brooks, M. E., Kristensen, K., van Benthem, K. J., Magnusson, A., Berg, C. W., Nielsen, A., Skaug, H. J., Mächler, M. and Bolker, B. M. (2017). glmmTMB balances speed and flexibility among packages for zero-inflated generalized linear mixed modeling. *The R Journal*, **9**(2), 378–400.

Kristensen, K., Nielsen, A., Berg, C. W., Skaug, H. and Bell, B. (2016). TMB: Automatic differentiation and Laplace approximation. *Journal of Statistical Software*, **70**, 1–21.

Millar, R. B. (2011). *Maximum Likelihood Estimation and Inference: With Examples in R, SAS and ADMB*. Wiley, New York.

Examples

```
(m1 <- glmmTMB(count ~ mined + (1|site),
  zi=~mined,
  family=poisson, data=Salamanders))
summary(m1)

## Zero-inflated negative binomial model
(m2 <- glmmTMB(count ~ spp + mined + (1|site),
  zi=~spp + mined,
  family=nbinom2, data=Salamanders))

## Hurdle Poisson model
(m3 <- glmmTMB(count ~ spp + mined + (1|site),
  zi=~spp + mined,
  family=truncated_poisson, data=Salamanders))

## Binomial model
data(cbpp, package="lme4")
(bovine <- glmmTMB(cbind(incidence, size-incidence) ~ period + (1|herd),
  family=binomial, data=cbpp))
```

```
## Dispersion model
sim1 <- function(nfac=40, nt=100, facsd=0.1, tsd=0.15, mu=0, residsd=1)
{
  dat <- expand.grid(fac=factor(letters[1:nfac]), t=1:nt)
  n <- nrow(dat)
  dat$REfac <- rnorm(nfac, sd=facsd)[dat$fac]
  dat$REt <- rnorm(nt, sd=tsd)[dat$t]
  dat$x <- rnorm(n, mean=mu, sd=residsd) + dat$REfac + dat$REt
  dat
}
set.seed(101)
d1 <- sim1(mu=100, residsd=10)
d2 <- sim1(mu=200, residsd=5)
d1$sd <- "ten"
d2$sd <- "five"
dat <- rbind(d1, d2)
m0 <- glmmTMB(x ~ sd + (1|t), dispformula=~sd, data=dat)
fixef(m0)$disp
c(log(5^2), log(10^2)-log(5^2)) # expected dispersion model coefficients


## Using 'map' to fix random-effects SD to 10
m1_map <- update(m1, map=list(theta=factor(NA)),
                 start=list(theta=log(10)))
VarCorr(m1_map)
```

`glmmTMBControl` *Control parameters for glmmTMB optimization*

Description

Control parameters for glmmTMB optimization

Usage

```
glmmTMBControl(
  optCtrl = NULL,
  optArgs = list(),
  optimizer = nlminb,
  profile = FALSE,
  collect = FALSE,
  parallel = NULL,
  eigval_check = TRUE
)
```

Arguments

`optCtrl`	Passed as argument `control` to optimizer. Default value (if default `nlminb` optimizer is used): `list(iter.max=300,eval.max=400)`

optArgs	additional arguments to be passed to optimizer function (e.g.: list(method="BFGS") when optimizer=optim)
optimizer	Function to use in model fitting. See Details for required properties of this function.
profile	Logical; Experimental option to improve speed and robustness when a model has many fixed effects
collect	Logical; Experimental option to improve speed by recognizing duplicated observations.
parallel	Numeric; Setting number of OpenMP threads to evaluate the negative log-likelihood in parallel
eigval_check	Check eigenvalues of variance-covariance matrix? (This test may be very slow for models with large numbers of fixed-effect parameters.)

Details

The general non-linear optimizer nlminb is used by glmmTMB for parameter estimation. It may sometimes be necessary to tweak some tolerances in order to make a model converge. For instance, the warning 'iteration limit reached without convergence' may be fixed by increasing the number of iterations using something like

glmmTMBControl(optCtrl=list(iter.max=1e3,eval.max=1e3)).

The argument profile allows glmmTMB to use some special properties of the optimization problem in order to speed up estimation in cases with many fixed effects. Enable this option using

glmmTMBControl(profile=TRUE).

Control parameters may depend on the model specification, because each control component is evaluated inside TMBStruc, the output of mkTMBStruc. To specify that profile should be enabled for more than 5 fixed effects one can use

glmmTMBControl(profile=quote(length(parameters$beta)>=5)).

The optimizer argument can be any optimization (minimizing) function, provided that:

- the first three arguments, in order, are the starting values, objective function, and gradient function;
- it also takes a control argument;
- it returns a list with elements (at least) par, objective, convergence (0 if convergence is successful) and message (the code internally handles output from optim(), by renaming the value component to objective)

Examples

```
## fit with default (nlminb) and alternative (optim/BFGS) optimizer
m1 <- glmmTMB(count~ mined, family=poisson, data=Salamanders)
m1B <- update(m1, control=glmmTMBControl(optimizer=optim,
               optArgs=list(method="BFGS")))
## estimates are *nearly* identical:
all.equal(fixef(m1), fixef(m1B))
```

`isLMM.glmmTMB` *support methods for parametric bootstrapping*

Description

see `refit` and `isLMM` for details

Usage

```
## S3 method for class 'glmmTMB'
isLMM(object)

## S3 method for class 'glmmTMB'
refit(object, newresp, ...)
```

Arguments

`object`	a fitted glmmTMB object
`newresp`	a new response vector
`...`	additional arguments (for generic consistency; ignored)

Details

These methods are still somewhat experimental (check your results carefully!), but they should allow parametric bootstrapping. They work by copying and replacing the original response column in the data frame passed to `glmmTMB`, so they will only work properly if (1) the data frame is still available in the environment and (2) the response variable is specified as a single symbol (e.g. `proportion` or a two-column matrix constructed on the fly with `cbind()`. Untested with binomial models where the response is specified as a factor.

Examples

```
if (requireNamespace("lme4")) {
## Not run:
   fm1 <- glmmTMB(count~mined+(1|spp),
                  ziformula=~mined,
                  data=Salamanders,
                  family=nbinom1)
   ## single parametric bootstrap step: refit with data simulated from original model
   fm1R <- refit(fm1, simulate(fm1)[[1]])
   ## the bootMer function from lme4 provides a wrapper for doing multiple refits
   ##   with a specified summary function
   b1 <- lme4::bootMer(fm1, FUN=function(x) fixef(x)$zi, nsim=20, .progress="txt")
   if (requireNamespace("boot")) {
      boot.ci(b1,type="perc")
    }

## End(Not run)
}
```

nbinom2 *Family functions for glmmTMB*

Description

Family functions for glmmTMB

Usage

```
nbinom2(link = "log")

nbinom1(link = "log")

compois(link = "log")

truncated_compois(link = "log")

genpois(link = "log")

truncated_genpois(link = "log")

truncated_poisson(link = "log")

truncated_nbinom2(link = "log")

truncated_nbinom1(link = "log")

beta_family(link = "logit")

betabinomial(link = "logit")

tweedie(link = "log")

ziGamma(link = "inverse")
```

Arguments

link	(character) link function for the conditional mean ("log", "logit", "probit", "inverse", "cloglog", "identity", or "sqrt")

Details

If specified, the dispersion model uses a log link. Denoting the variance as V, the dispersion parameter as $\phi = \exp(\eta)$ (where η is the linear predictor from the dispersion model), and the predicted mean as μ:

gaussian (from base R): constant $V = \phi$

Gamma (from base R) phi is the shape parameter. $V = \mu\phi$

ziGamma a modified version of `Gamma` that skips checks for zero values, allowing it to be used to fit hurdle-Gamma models

nbinom2 Negative binomial distribution: quadratic parameterization (Hardin & Hilbe 2007). $V = \mu(1 + \mu/\phi) = \mu + \mu^2/\phi$.

nbinom1 Negative binomial distribution: linear parameterization (Hardin & Hilbe 2007). $V = \mu(1 + \phi)$

compois Conway-Maxwell Poisson distribution: parameterized with the exact mean (Huang 2017), which differs from the parameterization used in the **COMPoissonReg** package (Sellers & Shmueli 2010, Sellers & Lotze 2015). $V = \mu\phi$.

genpois Generalized Poisson distribution (Consul & Famoye 1992). $V = \mu \exp(\eta)$. (Note that Consul & Famoye (1992) define ϕ differently.)

beta Beta distribution: parameterization of Ferrari and Cribari-Neto (2004) and the **betareg** package (Cribari-Neto and Zeileis 2010); $V = \mu(1 - \mu)/(\phi + 1)$

betabinomial Beta-binomial distribution: parameterized according to Morris (1997). $V = \mu(1 - \mu)(n(\phi + n)/(\phi + 1))$

tweedie Tweedie distribution: $V = \phi\mu^p$. The power parameter is restricted to the interval $1 < p < 2$

Value

returns a list with (at least) components

`family`	length-1 character vector giving the family name
`link`	length-1 character vector specifying the link function
`variance`	a function of either 1 (mean) or 2 (mean and dispersion parameter) arguments giving a value proportional to the predicted variance (scaled by `sigma(.)`)

References

- Consul PC & Famoye F (1992). "Generalized Poisson regression model." Communications in Statistics: Theory and Methods 21:89–109.
- Ferrari SLP, Cribari-Neto F (2004). "Beta Regression for Modelling Rates and Proportions." *J. Appl. Stat.* 31(7), 799-815.
- Hardin JW & Hilbe JM (2007). "Generalized linear models and extensions." Stata Press.
- Huang A (2017). "Mean-parametrized Conway–Maxwell–Poisson regression models for dispersed counts." *Statistical Modelling* 17(6), 1-22.
- Morris W (1997). "Disentangling Effects of Induced Plant Defenses and Food Quantity on Herbivores by Fitting Nonlinear Models." *American Naturalist* 150:299-327.
- Sellers K & Lotze T (2015). "COMPoissonReg: Conway-Maxwell Poisson (COM-Poisson) Regression". R package version 0.3.5. https://CRAN.R-project.org/package=COMPoissonReg
- Sellers K & Shmueli G (2010) "A Flexible Regression Model for Count Data." *Annals of Applied Statistics* 4(2), 943–61. https://doi.org/10.1214/09-AOAS306.

numFactor *Factor with numeric interpretable levels.*

Description

Create a factor with numeric interpretable factor levels.

Usage

```
numFactor(x, ...)

parseNumLevels(levels)
```

Arguments

x	Vector, matrix or data.frame that constitute the coordinates.
...	Additional vectors, matrices or data.frames that constitute the coordinates.
levels	Character vector to parse into numeric values.

Details

Some glmmTMB covariance structures require extra information, such as temporal or spatial coordinates. numFactor allows to associate such extra information as part of a factor via the factor levels. The original numeric coordinates are recoverable without loss of precision using the function parseNumLevels. Factor levels are sorted coordinate wise from left to right: first coordinate is fastest running.

Value

Factor with specialized coding of levels.

Examples

```
## 1D example
numFactor(sample(1:5,20,TRUE))
## 2D example
coords <- cbind( sample(1:5,20,TRUE), sample(1:5,20,TRUE) )
(f <- numFactor(coords))
parseNumLevels(levels(f)) ## Sorted
## Used as part of a model.matrix
model.matrix( ~f )
## parseNumLevels( colnames(model.matrix( ~f )) )
## Error: 'Failed to parse numeric levels: (Intercept)'
parseNumLevels( colnames(model.matrix( ~ f-1 )) )
```

`Owls` *Begging by Owl Nestlings*

Description

Begging by owl nestlings

Usage

```
data(Owls)
```

Format

The `Owls` data set is a data frame with 599 observations on the following variables:

`Nest` a factor describing individual nest locations

`FoodTreatment` (factor) food treatment: `Deprived` or `Satiated`

`SexParent` (factor) sex of provisioning parent: `Female` or `Male`

`ArrivalTime` a numeric vector

`SiblingNegotiation` a numeric vector

`BroodSize` brood size

`NegPerChick` number of negotations per chick

Note

Access to data kindly provided by Alain Zuur

Source

Roulin, A. and L. Bersier (2007) Nestling barn owls beg more intensely in the presence of their mother than in the presence of their father. *Animal Behaviour* **74** 1099–1106. https://doi.org/10.1016/j.anbehav.2007.01.027; http://www.highstat.com/Books/Book2/ZuurDataMixedModelling.zip

References

Zuur, A. F., E. N. Ieno, N. J. Walker, A. A. Saveliev, and G. M. Smith (2009) *Mixed Effects Models and Extensions in Ecology with R*; Springer.

Examples

```
data(Owls, package = "glmmTMB")
require("lattice")
bwplot(reorder(Nest,NegPerChick) ~ NegPerChick | FoodTreatment:SexParent,
       data=Owls)
dotplot(reorder(Nest,NegPerChick) ~ NegPerChick| FoodTreatment:SexParent,
        data=Owls)
```

```
## Not run:
## Fit negative binomial model with "constant" Zero Inflation :
owls_nb1 <- glmmTMB(SiblingNegotiation ~ FoodTreatment*SexParent +
                                    (1|Nest)+offset(log(BroodSize)),
              family = nbinom1(), zi = ~1, data=Owls)
owls_nb1_bs <- update(owls_nb1,
                      . ~ . - offset(log(BroodSize)) + log(BroodSize))
fixef(owls_nb1_bs)

## End(Not run)
```

`predict.glmmTMB` *prediction*

Description

prediction

Usage

```
## S3 method for class 'glmmTMB'
predict(
  object,
  newdata = NULL,
  newparams = NULL,
  se.fit = FALSE,
  re.form = NULL,
  allow.new.levels = FALSE,
  type = c("link", "response", "conditional", "zprob", "zlink", "disp"),
  zitype = NULL,
  na.action = na.pass,
  debug = FALSE,
  ...
)
```

Arguments

`object`	a `glmmTMB` object
`newdata`	new data for prediction
`newparams`	new parameters for prediction
`se.fit`	return the standard errors of the predicted values?
`re.form`	`NULL` to specify individual-level predictions; `~0` or `NA` to specify population-level predictions (i.e., setting all random effects to zero)
`allow.new.levels`	allow previously unobserved levels in random-effects variables? see details.
`type`	Denoting mu as the mean of the conditional distribution and `p` as the zero-inflation probability, the possible choices are:

"link" conditional mean on the scale of the link function, or equivalently the linear predictor of the conditional model

"response" expected value; this is $mu * (1 - p)$ for zero-inflated models and `mu` otherwise

"conditional" mean of the conditional response; `mu` for all models (i.e., synonymous with `"response"` in the absence of zero-inflation

"zprob" the probability of a structural zero (gives an error for non-zero-inflated models)

"zlink" predicted zero-inflation probability on the scale of the logit link function

"disp" dispersion parameter however it is defined for that particular family as described in `sigma.glmmTMB`

`zitype` deprecated: formerly used to specify type of zero-inflation probability. Now synonymous with `type`

`na.action` how to handle missing values in `newdata` (see `na.action`); the default (`na.pass`) is to predict `NA`

`debug` (logical) return the `TMBStruc` object that will be used internally for debugging?

`...` unused - for method compatibility

Details

- To compute population-level predictions for a given grouping variable (i.e., setting all random effects for that grouping variable to zero), set the grouping variable values to `NA`. Finer-scale control of conditioning (e.g. allowing variation among groups in intercepts but not slopes when predicting from a random-slopes model) is not currently possible.
- Prediction of new random effect levels is possible as long as the model specification (fixed effects and parameters) is kept constant. However, to ensure intentional usage, a warning is triggered if `allow.new.levels=FALSE` (the default).
- Prediction using "data-dependent bases" (variables whose scaling or transformation depends on the original data, e.g. `poly`, `ns`, or `poly`) should work properly; however, users are advised to check results extra-carefully when using such variables. Models with different versions of the same data-dependent basis type in different components (e.g. `formula= y ~ poly(x,3),dispformula= ~poly(x,2)`) will probably *not* produce correct predictions.

Examples

```
data(sleepstudy,package="lme4")
g0 <- glmmTMB(Reaction~Days+(Days|Subject),sleepstudy)
predict(g0, sleepstudy)
## Predict new Subject
nd <- sleepstudy[1,]
nd$Subject <- "new"
predict(g0, newdata=nd, allow.new.levels=TRUE)
## population-level prediction
nd_pop <- data.frame(Days=unique(sleepstudy$Days),
                     Subject=NA)
predict(g0, newdata=nd_pop)
```

print.VarCorr.glmmTMB *Printing The Variance and Correlation Parameters of a* glmmTMB

Description

Printing The Variance and Correlation Parameters of a glmmTMB

Usage

```
## S3 method for class 'VarCorr.glmmTMB'
print(
  x,
  digits = max(3, getOption("digits") - 2),
  comp = "Std.Dev.",
  formatter = format,
  ...
)
```

Arguments

x	a result of VarCorr(<glmmTMB>).
digits	number of significant digits to use.
comp	a string specifying the component to format and print.
formatter	a function.
...	optional further arguments, passed the next print method.

profile.glmmTMB *Compute likelihood profiles for a fitted model*

Description

Compute likelihood profiles for a fitted model

Usage

```
## S3 method for class 'glmmTMB'
profile(
  fitted,
  parm = NULL,
  level_max = 0.99,
  npts = 8,
  stepfac = 1/4,
  stderr = NULL,
  trace = FALSE,
```

```
  parallel = c("no", "multicore", "snow"),
  ncpus = getOption("profile.ncpus", 1L),
  cl = NULL,
  ...
)

## S3 method for class 'profile.glmmTMB'
confint(object, parm = NULL, level = 0.95, ...)
```

Arguments

`fitted`	a fitted `glmmTMB` object
`parm`	which parameters to profile, specified • by index (position) • by name (matching the row/column names of `vcov(object,full=TRUE)`) • as `"theta_"` (random-effects variance-covariance parameters) or `"beta_"` (conditional and zero-inflation parameters)
`level_max`	maximum confidence interval target for profile
`npts`	target number of points in (each half of) the profile (*approximate*)
`stepfac`	initial step factor (fraction of estimated standard deviation)
`stderr`	standard errors to use as a scaling factor when picking step sizes to compute the profile; by default (if `stderr` is `NULL`, or `NA` for a particular element), uses the estimated (Wald) standard errors of the parameters
`trace`	print tracing information? If `trace=FALSE` or 0, no tracing; if `trace=1`, print names of parameters currently being profiled; if `trace>1`, turn on tracing for the underlying `tmbprofile` function
`parallel`	method (if any) for parallel computation
`ncpus`	number of CPUs/cores to use for parallel computation
`cl`	cluster to use for parallel computation
`...`	additional arguments passed to `tmbprofile`
`object`	a fitted profile (`profile.glmmTMB`) object
`level`	confidence level

Details

Fits natural splines separately to the points from each half of the profile for each specified parameter (i.e., values above and below the MLE), then finds the inverse functions to estimate the endpoints of the confidence interval

Value

An object of class `profile.glmmTMB`, which is also a data frame, with columns `.par` (parameter being profiled), `.focal` (value of focal parameter), value (negative log-likelihood).

Examples

```
## Not run:
m1 <- glmmTMB(count~ mined + (1|site),
       zi=~mined, family=poisson, data=Salamanders)
salamander_prof1 <- profile(m1, parallel="multicore",
                            ncpus=2, trace=1)
## testing
salamander_prof1 <- profile(m1, trace=1,parm=1)
salamander_prof1M <- profile(m1, trace=1,parm=1, npts = 4)
salamander_prof2 <- profile(m1, parm="theta_")


## End(Not run)
salamander_prof1 <- readRDS(system.file("example_files","salamander_prof1.rds",package="glmmTMB"))
if (require("ggplot2")) {
    ggplot(salamander_prof1,aes(.focal,sqrt(value))) +
        geom_point() + geom_line()+
        facet_wrap(~.par,scale="free_x")+
    geom_hline(yintercept=1.96,linetype=2)
}
salamander_prof1 <- readRDS(system.file("example_files","salamander_prof1.rds",package="glmmTMB"))
confint(salamander_prof1)
confint(salamander_prof1,level=0.99)
```

`ranef.glmmTMB` *Extract Random Effects*

Description

Extract random effects from a fitted `glmmTMB` model, both for the conditional model and zero inflation.

Usage

```
## S3 method for class 'glmmTMB'
ranef(object, condVar = TRUE, ...)

## S3 method for class 'ranef.glmmTMB'
as.data.frame(x, ...)

## S3 method for class 'glmmTMB'
coef(object, condVar = FALSE, ...)
```

Arguments

`object`	a `glmmTMB` model.
`condVar`	whether to include conditional variances in result.

...	some methods for this generic function require additional arguments (they are unused here and will trigger an error)
x	a `ranef.glmmTMB` object (i.e., the result of running `ranef` on a fitted `glmmTMB` model)

Value

- For `ranef`, an object of class `ranef.glmmTMB` with two components:

  cond a list of data frames, containing random effects for the conditional model.

  zi a list of data frames, containing random effects for the zero inflation.

  If `condVar=TRUE`, the individual list elements within the `cond` and `zi` components (corresponding to individual random effects terms) will have associated `condVar` attributes giving the conditional variances of the random effects values. These are in the form of three-dimensional arrays: see `ranef.merMod` for details. The only difference between the packages is that the attributes are called 'postVar' in **lme4**, vs. 'condVar' in **glmmTMB**.
- For `coef.glmmTMB`: a similar list, but containing the overall coefficient value for each level, i.e., the sum of the fixed effect estimate and the random effect value for that level. *Conditional variances are not yet available as an option for* `coef.glmmTMB`.
- For `as.data.frame`: a data frame with components

  component part of the model to which the random effects apply (conditional or zero-inflation)

  grpvar grouping variable

  term random-effects term (e.g., intercept or slope)

  grp group, or level of the grouping variable

  condval value of the conditional mode

  condsd conditional standard deviation

Note

When a model has no zero inflation, the `ranef` and `coef` print methods simplify the structure shown, by default. To show the full list structure, use `print(ranef(model),simplify=FALSE)` or the analogous code for `coef`. In all cases, the full list structure is used to access the data frames, see example.

See Also

`fixef.glmmTMB`.

Examples

```
if (requireNamespace("lme4")) {
   data(sleepstudy, package="lme4")
   model <- glmmTMB(Reaction ~ Days + (1|Subject), sleepstudy)
   rr <- ranef(model)
   print(rr, simplify=FALSE)
   ## extract Subject conditional modes for conditional model
   rr$cond$Subject
   as.data.frame(rr)
}
```

residuals.glmmTMB *Compute residuals for a glmmTMB object*

Description

Compute residuals for a glmmTMB object

Usage

```
## S3 method for class 'glmmTMB'
residuals(object, type = c("response", "pearson"), ...)
```

Arguments

object	a "glmmTMB" object
type	(character) residual type
...	ignored, for method compatibility

Salamanders *Repeated counts of salamanders in streams*

Description

A data set containing counts of salamanders with site covariates and sampling covariates. Each of 23 sites was sampled 4 times. When using this data set, please cite Price et al. (2016) as well as the Dryad data package (Price et al. 2015).

Usage

```
data(Salamanders)
```

Format

A data frame with 644 observations on the following 10 variables:

site name of a location where repeated samples were taken

mined factor indicating whether the site was affected by mountain top removal coal mining

cover amount of cover objects in the stream (scaled)

sample repeated sample

DOP Days since precipitation (scaled)

Wtemp water temperature (scaled)

DOY day of year (scaled)

spp abbreviated species name, possibly also life stage

count number of salamanders observed

References

Price SJ, Muncy BL, Bonner SJ, Drayer AN, Barton CD (2016) Effects of mountaintop removal mining and valley filling on the occupancy and abundance of stream salamanders. *Journal of Applied Ecology* **53** 459–468. http://dx.doi.org/10.1111/1365-2664.12585

Price SJ, Muncy BL, Bonner SJ, Drayer AN, Barton CD (2015) Data from: Effects of mountaintop removal mining and valley filling on the occupancy and abundance of stream salamanders. *Dryad Digital Repository*. http://dx.doi.org/10.5061/dryad.5m8f6

Examples

```
require("glmmTMB")
data(Salamanders)

zipm3 = glmmTMB(count~spp * mined + (1|site), zi=~spp * mined, Salamanders, family="poisson")
```

`sigma.glmmTMB` *Extract residual standard deviation or dispersion parameter*

Description

For Gaussian models, `sigma` returns the value of the residual standard deviation; for other families, it returns the dispersion parameter, *however it is defined for that particular family*. See details for each family below.

Usage

```
## S3 method for class 'glmmTMB'
sigma(object, ...)
```

Arguments

`object`	a "glmmTMB" fitted object
`...`	(ignored; for method compatibility)

Details

The value returned varies by family:

gaussian returns the *maximum likelihood* estimate of the standard deviation (i.e., smaller than the results of `sigma(lm(...))` by a factor of (n-1)/n)

nbinom1 returns an overdispersion parameter (usually denoted α as in Hardin and Hilbe (2007)): such that the variance equals $\mu(1+\alpha)$.

nbinom2 returns an overdispersion parameter (usually denoted θ or k); in contrast to most other families, larger θ corresponds to a *lower* variance which is $\mu(1+\mu/\theta)$.

Gamma Internally, glmmTMB fits Gamma responses by fitting a mean and a shape parameter; sigma is estimated as (1/sqrt(shape)), which will typically be close (but not identical to) that estimated by `stats:::sigma.default`, which uses sqrt(deviance/df.residual)

beta returns the value of ϕ, where the conditional variance is $\mu(1-\mu)/(1+\phi)$ (i.e., increasing ϕ decreases the variance.) This parameterization follows Ferrari and Cribari-Neto (2004) (and the `betareg` package):

betabinomial This family uses the same parameterization (governing the Beta distribution that underlies the binomial probabilities) as `beta`.

genpois returns the index of dispersion ϕ^2, where the variance is $\mu\phi^2$ (Consul & Famoye 1992)

compois returns the value of $1/\nu$, When $\nu = 1$, compois is equivalent to the Poisson distribution. There is no closed form equation for the variance, but it is approximately undersidpersed when $1/\nu < 1$ and approximately oversidpersed when $1/\nu > 1$. In this implementation, μ is exactly the mean (Huang 2017), which differs from the COMPoissonReg package (Sellers & Lotze 2015).

tweedie returns the value of ϕ, where the variance is $\phi\mu^p$. The value of p can be extracted using the internal function `glmmTMB:::.tweedie_power`.

The most commonly used GLM families (`binomial`, `poisson`) have fixed dispersion parameters which are internally ignored.

References

- Consul PC, and Famoye F (1992). "Generalized Poisson regression model. Communications in Statistics: Theory and Methods" 21:89–109.
- Ferrari SLP, Cribari-Neto F (2004). "Beta Regression for Modelling Rates and Proportions." *J. Appl. Stat.* 31(7), 799-815.
- Hardin JW & Hilbe JM (2007). "Generalized linear models and extensions." Stata press.
- Huang A (2017). "Mean-parametrized Conway–Maxwell–Poisson regression models for dispersed counts. " *Statistical Modelling* 17(6), 1-22.
- Sellers K & Lotze T (2015). "COMPoissonReg: Conway-Maxwell Poisson (COM-Poisson) Regression". R package version 0.3.5. https://CRAN.R-project.org/package=COMPoissonReg

---

`simulate.glmmTMB` *Simulate from a glmmTMB fitted model*

Description

Simulate from a glmmTMB fitted model

Usage

```
## S3 method for class 'glmmTMB'
simulate(object, nsim = 1, seed = NULL, ...)
```

Arguments

object	glmmTMB fitted model
nsim	number of response lists to simulate. Defaults to 1.
seed	random number seed
...	extra arguments

Details

Random effects are also simulated from their estimated distribution. Currently, it is not possible to condition on estimated random effects.

Value

returns a list of vectors. The list has length nsim. Each simulated vector of observations is the same size as the vector of response variables in the original data set. In the binomial family case each simulation is a two-column matrix with success/failure.

---

terms.glmmTMB *Methods for extracting developer-level information from* glmmTMB *models*

Description

Methods for extracting developer-level information from glmmTMB models

Usage

```
## S3 method for class 'glmmTMB'
terms(x, component = "cond", part = "fixed", ...)

## S3 method for class 'glmmTMB'
model.matrix(object, component = "cond", part = "fixed", ...)
```

Arguments

x	a fitted glmmTMB object
component	model component ("cond", "zi", or "disp"; not all models contain all components)
part	whether to return results for the fixed or random effect part of the model (at present only part="fixed" is implemented for most methods)
...	additional arguments (ignored or passed to model.frame)
object	a fitted glmmTMB object

vcov.glmmTMB *Calculate Variance-Covariance Matrix for a Fitted glmmTMB model*

Description

Calculate Variance-Covariance Matrix for a Fitted glmmTMB model

Usage

```
## S3 method for class 'glmmTMB'
vcov(object, full = FALSE, ...)
```

Arguments

object	a "glmmTMB" fit
full	return a full variance-covariance matrix?
...	ignored, for method compatibility

Value

By default (full==FALSE), a list of separate variance-covariance matrices for each model component (conditional, zero-inflation, dispersion). If full==TRUE, a single square variance-covariance matrix for *all* top-level model parameters (conditional, dispersion, and variance-covariance parameters)

weights.glmmTMB *Extract weights from a glmmTMB object*

Description

Extract weights from a glmmTMB object

Usage

```
## S3 method for class 'glmmTMB'
weights(object, type = "prior", ...)
```

Arguments

object	a fitted glmmTMB object
type	weights type
...	additional arguments (not used; for methods compatibility)

Details

At present only explicitly specified *prior weights* (i.e., weights specified in the `weights` argument) can be extracted from a fitted model.

- Unlike other GLM-type models such as `glm` or `glmer`, `weights()` does not currently return the total number of trials when binomial responses are specified as a two-column matrix.
- Since `glmmTMB` does not fit models via iteratively weighted least squares, `working weights` (see `weights.glm`) are unavailable.

Index