Package 'drmTMB'

September 15, 2026

Title Distributional Regression Models Using Template Model Builder

Version 0.7.0

Description Fast distributional regression models for univariate and bivariate responses using Template Model Builder. The current implementation focuses on Gaussian, Student-t, and skew-normal location-scale models,
known sampling covariance, phylogenetic location effects, random-effect scale models, bivariate residual correlation, positive-continuous, 'Tweedie' semi-continuous, strict-proportion, zero-one bounded, and denominator-aware proportion families, fixed-effect Bernoulli/binomial event-probability models, and
fixed-effect Poisson, negative-binomial, zero-inflated, zero-truncated, hurdle count, and ordinal cumulative-logit models.
Additional response-family models are staged for later phases. Every fitted family also exposes a distributional-output and
adequacy layer: randomized quantile-residual worm and QQ plots that detect fixed-effect shape and atom misspecification, and
conditional-quantile, exceedance, and centile outputs with plug-in (uncalibrated) intervals.

License GPL (>= 3)

URL https://itchyshin.github.io/drmTMB/,
https://github.com/itchyshin/drmTMB

BugReports https://github.com/itchyshin/drmTMB/issues

Encoding UTF-8

Language en-GB

Depends R (>= 4.1.0)

Imports cli, lifecycle, Matrix (>= 1.6.0), methods, stats, TMB (>= 1.9.6), utils

LinkingTo RcppEigen, TMB

Suggests ape, callr, detectseparation (>= 0.4.0), emmeans, extraDistr, fmesher, glmmTMB, ggplot2, JuliaCall, knitr, lme4, MASS, metadat, metafor, mvtnorm, nlme, numDeriv, ordinal,

palmerpenguins, pkgload, rmarkdown, sf, spelling, statmod, testthat (>= 3.0.0), tweedie, withr

Config/testthat/edition 3

VignetteBuilder knitr

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.2

NeedsCompilation yes

Author Shinichi Nakagawa [aut, cre, cph] (ORCID: <https://orcid.org/0000-0002-7765-5182>)

Maintainer Shinichi Nakagawa <itchyshin@gmail.com>

Repository CRAN

Date/Publication 2026-09-15 11:00:29 UTC

Contents

`animal` *Animal-model structured-effect marker*

Description

`animal()` marks pedigree or additive-relatedness animal-model syntax. It is the biological front door for questions such as whether among-individual additive genetic variance appears in the location `mu`, residual scale `sigma`, shape or skewness, inflation, or a bivariate covariance. The fitted routes are univariate Gaussian `mu` and `sigma` random intercepts from a small pedigree data frame, precomputed additive relationship matrix `A`, or inverse relationship matrix `Ainv`, for example `animal(1 | id, pedigree = pedigree)` or `animal(1 | id, Ainv = Ainv)`. Matching univariate `mu` and `sigma` intercept terms estimate one animal-model mean-scale correlation. The first bivariate Gaussian q=2 location covariance comes from matching labelled terms in `mu1` and `mu2`, and the constant all-four q=4 location-scale block comes from matching labelled terms in `mu1`, `mu2`, `sigma1`, and `sigma2`, for example `animal(1 | p | id, pedigree = pedigree)`. The pedigree route builds a dense additive relationship matrix from `id`, `dam`, and `sire` columns. The univariate Gaussian `mu` path also supports one numeric slope, for example `animal(1 + x | id, pedigree = pedigree)`, as independent intercept and slope fields with separate SDs and no intercept-slope correlation. Large-pedigree sparse precision construction, multiple structured slopes, slope correlations, predictor-dependent `corpair()` regression, residual-scale structured slopes, and animal-model `sd*()` direct-SD grammar remain planned.

Usage

```
animal(term, pedigree = NULL, A = NULL, Ainv = NULL)
```

Arguments

`term`	Structured random-effect term, such as `1 | id` or `1 + x | id`.
`pedigree`	Pedigree data frame with columns `id`, `dam`, and `sire`. Unknown parents can be `NA`, `""`, or `"0"`. The first fitted route builds a dense additive relationship matrix for Gaussian `mu` animal effects.
`A`	Additive relatedness or covariance matrix for the first fitted univariate Gaussian `mu` path.
`Ainv`	Sparse or dense inverse additive relatedness matrix for the first fitted univariate Gaussian `mu` path.

Value

A formula marker; never evaluated by users.

Examples

```
# Fitted: additive genetic variance in body size from a precomputed Ainv.
bf(body_size ~ age + sex + animal(1 | id, Ainv = Ainv),
  sigma ~ habitat
)

# Fitted: the same route can build a small additive matrix from a pedigree.
bf(activity ~ treatment + animal(1 | id, pedigree = pedigree),
  sigma ~ treatment
)
```

---

anova.drmTMB *Likelihood comparison guard for drmTMB fits*

Description

Experimental MSPL fits do not expose likelihood-ratio comparisons. Ordinary drmTMB fits likewise have no package-level `anova()` comparison contract; compare explicitly supported likelihood quantities instead.

Usage

```
## S3 method for class 'drmTMB'
anova(object, ..., test = NULL)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Additional fitted objects.
`test`	Requested test label; currently unsupported.

Value

This method always errors.

---

associate_pairs *Associate two frozen marginal drmTMB fits*

Description

`associate_pairs()` estimates a named within-row association after fitting two marginal models. It never refits, updates, profiles, or otherwise alters either margin. The reviewed Arc 6 slices implement fixed-effect Gaussian margins paired with literal Bernoulli `binomial(link = "logit")` or ordinary `nbinom2()` margins, literal Bernoulli paired with ordinary `nbinom2()`, two literal Bernoulli margins, and two ordinary `nbinom2()` margins, on the same complete analysis rows.

Usage

```
associate_pairs(fit_1, fit_2, kernel, association)
```

Arguments

fit_1, fit_2	Two fitted drmTMB marginal models. They must use the identical complete analysis data, in the same order.
kernel	A named association kernel. Arc 6 accepts only latent_normal().
association	Association formula. Most Arc 6 pair classes accept only ~ 1. The beta Bernoulli x ordinary-NB2 route accepts an intercept-bearing fixed-effect formula, including multiple predictors, factors, interactions, and explicit transformations. Random effects, offsets, missing values, aliased columns, and . expansion are not supported.

Details

The fitted parameter eta is a Gaussian-copula latent-normal association. It is neither rho12(), an observed-scale correlation, nor corpairs(). The corpair() formula marker is a distinct interface. The stage-2 Hessian treats the margins as fixed and is not used for uncertainty. For every admitted pair route, vcov() and confint() instead use a two-stage Godambe covariance that propagates fitted-margin uncertainty when the fit-specific calculation succeeds. These alpha-scale routes are interval-feasible. The retained Bernoulli x ordinary-NB2 intercept campaign supports the stronger inference-ready-with-caveats tier. Intercept-only associations also expose bounded eta intervals through confint(object, type = "eta"); predict.drm_pair_association() supplies delta-method eta standard errors and pointwise transformed intervals. Profiles remain unavailable.

Value

An object of class drm_pair_association.

Examples

```
set.seed(20260723)
dat <- data.frame(x = rnorm(80))
z_g <- rnorm(80)
z_b <- 0.35 * z_g + sqrt(1 - 0.35^2) * rnorm(80)
dat$trait_continuous <- 0.2 + 0.5 * dat$x + z_g
dat$trait_binary <- as.integer(z_b > qnorm(0.6))

gaussian_fit <- drmTMB(
  bf(mu = trait_continuous ~ x, sigma = ~ 1),
  family = gaussian(), data = dat
)
binary_fit <- drmTMB(
  bf(mu = trait_binary ~ x), family = binomial(), data = dat
)
assoc <- associate_pairs(
  gaussian_fit, binary_fit,
  kernel = latent_normal(), association = ~ 1
)
```

```
association(assoc)
sqrt(diag(vcov(assoc)))
confint(assoc)
confint(assoc, type = "eta")
```

`association` *Extract a pair association estimate*

Description

Extract a pair association estimate

Usage

```
association(object, ...)

## S3 method for class 'drm_pair_association'
association(object, type = c("coefficient", "fitted"), ...)
```

Arguments

`object`	A `drm_pair_association` object.
`...`	Reserved for future extractor options.
`type`	For a constant association, the default `"coefficient"` returns the usual single `eta`. For a covariate-varying beta association, `"coefficient"` returns the association-link coefficients and `"fitted"` returns the frozen-row latent-normal associations.

Value

For a constant association, a one-row data frame with the latent-normal association and diagnostic status. For a beta association formula, a coefficient table or a frozen-row `eta` table according to `type`. The separate `vcov()` and `confint()` methods supply alpha-scale uncertainty for admitted association routes whose fit-specific Godambe covariance diagnostics pass; `confint(object, type = "eta")` supplies the transformed interval for a constant association.

`beta` *Beta response family*

Description

`beta()` defines a one-response distribution for continuous proportions strictly inside `(0, 1)`, with formulas for mean `mu` and scale `sigma`.

Usage

```
beta()
```

Details

The implemented contract is `logit(mu)` = `eta_mu`, `log(sigma)` = `eta_sigma`, and internal precision `phi` = `1 / sigma^2`. Larger `sigma` therefore means more variation around the mean, not more precision. Ordinary unlabelled random intercepts and independent numeric slopes such as `(1 | id)` and `(0 + x | id)` may enter the logit-`mu` predictor; `sigma` remains fixed-effect in this first slice.

This helper masks `base::beta()` when `drmTMB` is attached. Use `base::beta()` for the mathematical beta function.

Value

A `drm_family` object.

Examples

```
beta()
```

---

beta_binomial *Beta-binomial response family*

Description

`beta_binomial()` defines a one-response denominator-aware distribution for successes out of known trials. Use it with two-column count responses such as `bf(cbind(successes, failures) ~ x, sigma ~ z)`, where `trials_i` = `successes_i` + `failures_i`.

Usage

```
beta_binomial()
```

Details

The implemented contract is `logit(mu)` = `eta_mu`, with optional ordinary unlabelled `mu` random intercepts, `log(sigma)` = `eta_sigma`, and internal beta precision `phi` = `1 / sigma^2`. Conditional on a latent success probability `p_i ~ Beta(mu_i * phi_i, (1 - mu_i) * phi_i)`, the observed successes follow `Binomial(trials_i, p_i)`. Larger `sigma` means more extra-binomial variation around the mean probability.

The first mixed-model slice supports ordinary `mu` random intercepts and independent numeric slopes such as `bf(cbind(successes, failures) ~ x + (1 | id) + (0 + x | id), sigma ~ z)`. Correlated slopes, labelled covariance blocks, `sigma` random effects, `zoi`/`coi`, `meta_V(V = V)`, phylogenetic or spatial terms, bivariate beta-binomial models, and a `successes/trials` response alias are planned but not implemented.

Value

A drm_family object.

Examples

```
beta_binomial()
```

`biv_associate` *Fit two margins and construct a frozen-margin association in one call*

Description

`biv_associate()` is the convenience front end for the reviewed Arc 6 frozen-margin route. It fits two univariate margins to the supplied data and then calls `associate_pairs()` without refitting either margin. It is one R call, but it is not a jointly fitted bivariate model: stage 2 treats the fitted marginal parameters as fixed and estimates only the latent-normal association `eta`.

Usage

```
biv_associate(
  formula_1,
  formula_2,
  family,
  data,
  kernel = latent_normal(),
  association = ~1,
  control_1 = list(),
  control_2 = list()
)
```

Arguments

`formula_1, formula_2`	Univariate `drm_formula` objects for the first and second response margins.
`family`	A two-element list of marginal family objects.
`data`	A data frame containing both responses and every predictor used by either margin.
`kernel`	Association kernel. Arc 6 accepts only `latent_normal()`.
`association`	Association formula. Most Arc 6 pair classes accept only `~ 1`, which estimates one constant association parameter. The beta Bernoulli x ordinary-NB2 route accepts an intercept-bearing fixed-effect model-matrix formula, including multiple predictors, factors, interactions, and explicit transformations. Alpha-scale standard errors and Wald intervals are available for every admitted association formula when its fit-specific Godambe covariance diagnostics pass.
`control_1, control_2`	Optional control lists passed to the corresponding marginal `drmTMB()` fits.

Details

The two formulas must be univariate `bf()` or `drm_formula()` objects, and `family` must be a two-element list. The supplied `data` must already be the same complete paired analysis data for both margins. If the two marginal fits retain different rows, the constructor fails rather than silently comparing different individuals.

Value

A `drm_pair_association` object that retains frozen snapshots of both fitted margins. `association()` returns the point estimate unless the numerical diagnostic is boundary-unresolved; a near-boundary status remains flagged. Admitted routes have alpha-scale `vcov()` and `confint()` methods when the fit-specific Godambe covariance succeeds. Intercept-only routes also have bounded eta intervals; `predict.drm_pair_association()` returns eta-scale standard errors and pointwise confidence intervals. No profile is available.

Examples

```
set.seed(20260725)
dat <- data.frame(x = rnorm(80))
z_continuous <- rnorm(80)
z_binary <- 0.35 * z_continuous + sqrt(1 - 0.35^2) * rnorm(80)
dat$trait_continuous <- 0.2 + 0.5 * dat$x + z_continuous
dat$trait_binary <- as.integer(z_binary > qnorm(0.4))

assoc <- biv_associate(
  bf(mu = trait_continuous ~ x, sigma = ~ 1),
  bf(mu = trait_binary ~ x),
  family = list(gaussian(), binomial()), data = dat
)
association(assoc)
sqrt(diag(vcov(assoc)))
confint(assoc)
confint(assoc, type = "eta")
```

`biv_gaussian` *Bivariate Gaussian response family*

Description

`biv_gaussian()` defines a two-response Gaussian distribution with formulas for both locations, both residual standard deviations, and residual correlation `rho12`. The residual-correlation link is recorded as `"atanh_guarded"` because fitted response-scale correlations use `rho12 = 0.999999 * tanh(eta_rho12)`.

Usage

```
biv_gaussian()
```

Value

A drm_family object.

Examples

```
biv_gaussian()
```

---

biv_lognormal *Bivariate lognormal response family*

Description

biv_lognormal() defines an exact two-response lognormal distribution. Its rho12 is the residual correlation on the log-response scale, not a raw-response correlation and not the frozen-margin association eta.

Usage

```
biv_lognormal()
```

Value

A drm_family object.

Examples

```
biv_lognormal()
```

---

biv_student *Bivariate Student-t response family*

Description

biv_student() defines an exact two-response elliptical Student-t distribution with one shared degrees-of-freedom parameter nu. Its sigma1 and sigma2 parameters are Student-t scales, and rho12 is the residual scatter correlation. The response covariance exists because the fitted transform enforces nu > 2; its marginal standard deviations are sigma_j * sqrt(nu / (nu - 2)).

Usage

```
biv_student()
```

Details

The first implementation allows fixed effects in mu1 and mu2, with intercept-only sigma1, sigma2, shared nu, and rho12. Random effects, parameter predictors outside the locations, missing response pairs, and interval inference are deferred.

Value

A drm_family object.

Examples

```
biv_student()
```

`categorical` *Unordered categorical missing-predictor family*

Description

`categorical()` defines a baseline-category softmax model for one unordered categorical predictor used inside `mi()`. It is currently a predictor-model family for `impute_model()`, not a response family for `drmTMB()`.

Usage

```
categorical()
```

Details

The first fitted route is fixed-effect and uses the first factor level as the baseline category. Missing predictor values are integrated by exact summation over the unordered levels.

Value

A drm_impute_family object.

Examples

```
categorical()
```

`centile_chart` *Model-conditional centile chart*

Description

`centile_chart()` draws fitted response centile curves against one covariate, holding every other predictor at a reference value (numeric predictors at their fitted mean, factors at their first fitted level; see `prediction_grid()`). Each curve is `predict(object, newdata = grid, type = "quantile", prob = p)` for one `p` in `prob`.

Usage

```
centile_chart(object, covariate, prob = c(0.03, 0.15, 0.5, 0.85, 0.97), ...)

## S3 method for class 'drmTMB'
centile_chart(
  object,
  covariate,
  prob = c(0.03, 0.15, 0.5, 0.85, 0.97),
  dpar = NULL,
  n = 100L,
  ...
)
```

Arguments

object	A drmTMB fit.
covariate	Character scalar naming the predictor to vary.
prob	Numeric vector of probabilities in (0, 1) giving the centiles to draw.
...	Reserved for future options.
dpar	Distributional parameter identifying the response; see predict.drmTMB(). If NULL, the first fitted distributional parameter is used (for bivariate biv_gaussian fits, dpar selects the response – see predict.drmTMB()'s Details for the marginal-only scope).
n	Number of grid points spanning covariate's fitted range; passed to prediction_grid().

Details

centile_chart() is a MODEL-CONDITIONAL summary at predict_parameters()'s fixed-effect, population-level parameter estimates theta_hat – it is **not** a WHO-style or other population reference standard. The reference values every non-focal covariate is held at are reported in the plot subtitle so a reader is not left guessing what "conditional" means here.

Value

A ggplot object.

See Also

predict.drmTMB(), prediction_grid()

Examples

```
set.seed(20260712)
n <- 60
x <- stats::rnorm(n)
dat <- data.frame(y = 0.5 + 0.8 * x + stats::rnorm(n), x = x)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), family = gaussian(), data = dat)
if (requireNamespace("ggplot2", quietly = TRUE)) {
```

```
  centile_chart(fit, covariate = "x")
}
```

---

check_drm *Check convergence and diagnostic flags for a drmTMB fit*

Description

check_drm() runs a compact set of model-fit diagnostics. It is intended as a first-pass guardrail before interpreting distributional models, especially fits with random effects, known sampling covariance, phylogenetic location effects, or bivariate residual correlation rho12.

Usage

```
check_drm(object, ...)

## S3 method for class 'drmTMB'
check_drm(
  object,
  gradient_tolerance = 0.001,
  rho_boundary = 0.98,
  sd_boundary = 1e-04,
  ...
)
```

Arguments

object	A drmTMB fit.
...	Reserved for future diagnostic options.
gradient_tolerance	Maximum absolute fixed-parameter gradient treated as acceptable.
rho_boundary	Absolute residual or structured correlation value above which a bivariate Gaussian fit receives a warning.
sd_boundary	Random-effect standard deviation below which a fit receives a warning that the variance component is near the lower boundary.

Details

The current checks cover optimizer convergence, finite objective values, optimizer evaluation counts, fixed-parameter gradients including the largest gradient component label, whether TMB::sdreport() was computed, skipped, or failed, Hessian status from TMB::sdreport(), finite fixed-effect standard errors, standard errors that are finite but inflated relative to the others despite a positive-definite Hessian (a weakly identified, near-flat direction such as a boundary correlation), dropped rows, positive scale parameters, random-effect standard deviations near the lower boundary, bivariate residual-correlation rho12 values near the boundary, Student-t nu boundary behaviour, skew-normal nu finite-value checks, known sampling covariance summaries, dense known-covariance storage

scale, dense fixed-effect design size and density, random-effect replication, and random-slope design variation. If a univariate Gaussian fit includes one or more matched labelled mu/sigma random-intercept covariance blocks, check_drm() also reports group replication and whether either component is tiny relative to its interpretation scale for each independent block. If a bivariate Gaussian fit includes one or more matched same-response labelled mu/sigma random-intercept covariance blocks, check_drm() reports one row per block. If a bivariate Gaussian fit includes a matched labelled mu1/mu2 random-intercept covariance block, check_drm() reports group replication and whether either group-level SD is tiny relative to the matching residual scale. For a matched labelled sigma1/sigma2 block, it reports group replication and whether either log-sigma random-effect SD is tiny. If a bivariate Gaussian fit includes an ordinary all-four q=4 mu1/mu2/sigma1/sigma2 block, it reports group replication, location SDs relative to residual scales, log-sigma SDs, and whether any latent correlation is near the boundary. If a bivariate Gaussian fit includes matching mu1/mu2 phylogenetic location effects, check_drm() also reports whether the fitted phylogenetic mean-mean correlation is near the boundary, whether either phylogenetic SD is tiny relative to the matching residual scale, and whether an ordinary group-level covariance block uses the same grouping factor. Matching bivariate coordinate-spatial q=2, animal(), and relmat() q=2 location effects receive the corresponding structured replication, SD-ratio, and boundary-correlation diagnostics. If a bivariate Gaussian fit includes a phylogenetic, coordinate-spatial, animal-model, or relmat() q=4 mu1/mu2/sigma1/sigma2 block, it reports level replication, location SDs relative to residual scales, log-sigma SDs, and whether any latent structured correlation is near the boundary. If a univariate Gaussian fit includes phylo(1 | species, tree = tree) or phylo(1 + x | species, tree = tree) in mu, it reports species replication, the fitted phylogenetic SDs, and whether the smallest phylogenetic SD is tiny relative to the residual scale. If a univariate Gaussian fit includes spatial(1 | site, coords = coords) or spatial(1 + x | site, coords = coords) in mu, it reports site replication, fitted coordinate range, the spatial SDs, and whether the smallest spatial SD is tiny relative to the residual scale. If a Gaussian fit includes sd_phylo(species) ~ x_species, sd_phylo1(species) ~ x_species, or sd_phylo2(species) ~ x_species, it reports species replication and the fitted direct-SD surface range. If a univariate Gaussian fit used drm_control(aggregate_gaussian = TRUE), it reports original rows, aggregation cells, compression ratio, and largest cell size. If a fit was stored with drm_control(keep_tmb_object = FALSE), the fixed-gradient check is reported as a note because the TMB automatic-differentiation object is not available. If a fit used drm_control(se = FALSE), the sdreport_status, Hessian, and finite-standard-error checks are reported as notes. If sdreport() was requested but failed, those rows are warnings.

Use check_drm() before interpreting coefficients, fitted values, or response-scale quantities. A note records something to inspect, such as dropped rows or a singly observed random-effect level. A warning means the fitted model may still be useful but needs inspection before inference. An error means at least one basic diagnostic failed. A Hessian or sdreport() warning is therefore an inference and identifiability signal, not automatic proof that fitted point estimates are unusable. For programmatic checks, the returned object has attr(x, "ok") == TRUE only when no rows have warning or error status.

For native drmTMB fits, this is a stable reader schema: the drm_check data frame has, in order, the character columns check, status, value, and message. status is one of "ok", "note", "warning", or "error". Its "ok" attribute is TRUE exactly when no row has status "warning" or "error".

Value

A drm_check data frame. Its stable native reader columns are, in order, check, status, value, and message, all character. status uses only "ok", "note", "warning", and "error"; its "ok" attribute is TRUE exactly when no row has status "warning" or "error".

Examples

```
set.seed(1)
dat <- data.frame(y = rnorm(40), x = rnorm(40))
fit <- drmTMB(drm_formula(y ~ x, sigma ~ x), data = dat)
check_drm(fit)
```

---

confint.drmTMB *Confidence intervals for fitted model parameters*

Description

confint() returns confidence intervals for a fitted drmTMB model. Wald intervals are fast and are returned for fixed-effect coefficients and direct response-scale parameter targets by default. Direct Wald targets include constant residual-scale, random-effect standard-deviation, random-effect correlation, and constant residual-correlation rows when the fitted TMB parameter and TMB::sdreport() covariance are available. Correlation Wald intervals are computed on the fitted TMB correlation-link scale, equivalent to a guarded Fisher z/atanh transform, and then returned on the correlation scale. Bootstrap intervals simulate and refit direct targets. For positive scale and SD targets, percentile endpoints are taken on the fitted log scale before back-transforming to the response scale. Profile-likelihood intervals are slower because nuisance parameters are re-optimized; this first public profile path supports explicit fixed-effect, constant distributional-scale, random-effect standard-deviation, random-effect correlation, bivariate phylogenetic q=2 mean-mean correlation, block-diagonal bivariate phylogenetic mu1/mu2 and sigma1/sigma2 correlations, and constant residual-correlation targets. For predictor-dependent scale, residual-correlation, or currently supported corpair() formulae, supply newdata with parm = "sigma", parm = "rho12", or the fitted corpair(...) dpar to profile the fitted response-scale value for each supplied row. Row-specific rho12 profile intervals from a regression rho12 formula are computable this way, but no simulation has measured their coverage; treat them as interval-feasible, not as a calibrated level interval (tracked as issue #802).

Usage

```
## S3 method for class 'drmTMB'
confint(
  object,
  parm = NULL,
  level = 0.95,
  method = c("wald", "profile", "bootstrap"),
  newdata = NULL,
  trace = FALSE,
  profile_precision = c("default", "fast"),
```

```
  profile_maxit = NULL,
  profile_engine = c("auto", "endpoint", "tmbprofile"),
  profile_endpoint_max_eval = NULL,
  R = 199L,
  seed = NULL,
  parallel = c("none", "multicore"),
  workers = NULL,
  refit_control = NULL,
  sd_boundary = 1e-04,
  rho_boundary = 0.98,
  small_sample_df = c("location", "none", "group"),
  bias_correct = c("location", "none", "group"),
  bootstrap_re_form = NULL,
  ...
)
```

Arguments

object	A drmTMB fit.
parm	Optional character or integer vector selecting interval targets. NULL selects all direct Wald-ready targets for Wald intervals. Profile intervals require explicit target names or target-set shortcuts. Supported shortcuts are "fixed_effects", "random_effects", "variance_components", and "correlations".
level	Confidence level.
method	Interval method: "wald", "profile", or "bootstrap". If newdata is supplied and method is omitted, method = "profile" is used.
newdata	Optional data frame for response-scale profile intervals for predictor-dependent sigma, sigma1, sigma2, rho12, or fitted corpair() values. Each row is profiled separately by profiling its fixed-effect linear predictor and then transforming the interval to the response scale.
trace	Logical; passed to TMB::tmbprofile() when the tmbprofile profile engine is used.
profile_precision	Profile-control shortcut. "default" leaves TMB::tmbprofile() controls unchanged. "fast" supplies ystep = 0.5 and ytol = 2 unless the caller supplies those controls in ..., giving a quicker first-pass profile for long variance-component or correlation targets.
profile_maxit	Optional positive whole number passed to TMB::tmbprofile() as maxit when method = "profile". Use this as a per-target adaptive-step budget for long or exploratory profile runs.
profile_engine	Profile engine for direct fitted-object targets. "auto" uses a scalar endpoint solver for direct scale, SD, and correlation targets when no TMB::tmbprofile() controls are supplied, and otherwise uses TMB::tmbprofile(). "endpoint" requires the scalar endpoint solver, while "tmbprofile" preserves the previous full-profile route for comparison and debugging.

`profile_endpoint_max_eval`
: Optional positive whole number limiting constrained endpoint evaluations per endpoint side when the scalar endpoint engine is used. This is a diagnostic escape hatch for long variance-component or correlation profiles; when the budget is reached the row is returned with `conf.status = "profile_failed"` and missing endpoints.

`R`
: Number of parametric-bootstrap refits when `method = "bootstrap"`.

`seed`
: Optional seed for bootstrap simulation.

`parallel`
: Profile or bootstrap backend: `"none"` or Unix `"multicore"`. For profile intervals, targets or `newdata` rows are split across workers. For bootstrap intervals, refits are split across workers.

`workers`
: Requested profile or bootstrap workers. If `NULL` and `parallel = "multicore"`, `drmTMB` uses about half the detected CPU cores. Multicore execution is capped at the number of jobs and at 10 workers.

`refit_control`
: Optional `drm_control()` object used for bootstrap refits. The default skips `TMB::sdreport()` and drops the TMB object because bootstrap intervals use refit point estimates.

`sd_boundary`, `rho_boundary`
: Boundary thresholds used by `method = "wald"` to flag intervals where the symmetric Wald interval is unreliable: a variance-component standard deviation within `sd_boundary` of zero, or a correlation within `rho_boundary` of `+/-1`, is returned with `conf.status = "wald_at_boundary"` and a warning pointing to `method = "profile"`. Defaults match `check_drm()` (`1e-4`, `0.98`).

`small_sample_df`
: Small-sample reference distribution for the `method = "wald"` interval. `"location"` (the default) references a t-quantile with `df = g - 1` – a group-based, Satterthwaite-style between-group degrees of freedom – for each *location-axis* (`mu`, `mu1`, `mu2`) structured random-effect SD target (`phylo`, `spatial`, `animal`, `relmat`) with a resolvable group count `g`, and keeps the ordinary normal quantile `z = qnorm((1 + level) / 2)` for every other target. This widens only the location-axis structured-RE SD intervals, which under-cover at small `g` under the normal quantile; the t-quantile lifts their coverage and converges back to `z` as `g` grows. Dispersion (`sigma`, `sigma1`, `sigma2`) structured SD targets are deliberately left at the normal quantile because they already over-cover; non-structured and fixed-effect targets are never widened; and a plain labelled covariance block such as `(1 + x | p | id)` is also left at the normal quantile by default, because the correction magnitude was calibrated for structured blocks only. `"none"` is the opt-out: it uses the normal quantile for *every* target and is byte-identical to the pre-default behaviour. `"group"` widens *every* resolvable SD target – structured and labelled-covariance, location and dispersion axis alike; use it only when you deliberately want to t-reference the dispersion axis or a labelled covariance block too. Note that the t-quantile only widens the interval; it does not move its centre. The residual small-g gap is ML shrinkage bias in the variance-component point estimate (a job for REML), not a quantile problem, so the centre shift is handled by `bias_correct`.

`bias_correct`
: Small-sample point-estimate bias correction for the `method = "wald"` interval. `"location"` (the default) adds a simulation-calibrated shift `log(g / (g - 1))` to

the log-scale point estimate – before back-transforming – of each *location-axis* (`mu`, `mu1`, `mu2`) structured random-effect SD target with a resolvable group count `g` (the same targets `small_sample_df = "location"` widens), and leaves every other target unshifted. The ML estimate of a structured-RE variance-component SD is biased low on the log scale by about `log(g / (g - 1))` – a downward shrinkage that REML, or simply a larger `g`, removes asymptotically; this shift moves the interval centre up to counter that shrinkage. Used together with `small_sample_df = "location"`, the centre shift and the `t(df = g - 1)` width act independently, giving `exp((log(sigma_hat) + log(g / (g - 1))) +/- qt(p, g - 1) * se_log)` and lifting small-g coverage of location-axis structured-RE SD targets to nominal. `"none"` is the opt-out: it leaves every centre at the ML estimate and is byte-identical to the pre-default behaviour. `"group"` shifts *every* resolvable SD target – structured and labelled-covariance, location and dispersion axis alike. The default deliberately excludes the dispersion axis (`sigma` SD intervals already over-cover, so the upward shift would push them further conservative) and plain labelled covariance blocks (whose correction magnitude is not yet simulation-calibrated).

`bootstrap_re_form`

`re.form` passed through to `stats::simulate()` when `method = "bootstrap"`. The default, `NULL`, redraws random effects for every replicate (marginal simulation), matching what a parametric bootstrap of the fitted data-generating process requires. Passing `NA` instead holds random effects fixed at their conditional-mode estimate; this is a documented escape hatch for random-effect structures whose marginal draw is not yet implemented, and it makes the resulting intervals **anticonservative** because between-group variability is never resampled – it is not a neutral alternative to the default.

`...`

Additional arguments passed to `TMB::tmbprofile()` when `method = "profile"` and the `tmbprofile` profile engine is used. `drmTMB` supplies the profiled `obj`, `name`, `lincomb`, and `trace` arguments internally; set the profile target with `parm`.

Details

Target names follow the profile target namespace. For fixed effects, use names such as `"fixef:mu:x"`, `"fixef:sigma:(Intercept)"`, or `"fixef:rho12:w"`. Compact coefficient labels from `summary(fit)`, such as `"mu:x"`, are also accepted. Random-effect SD intervals are reported on the SD scale, and random-effect correlation intervals are reported on the correlation scale. For bivariate Gaussian and exact bivariate lognormal fits with constant residual correlation, `parm = "rho12"` profiles the residual correlation and reports the interval on the response correlation scale. For fits with constant `sigma`, `sigma1`, or `sigma2`, `parm = "sigma"` and friends report response-scale intervals.

The fastest routine route is `confint(fit)`, which uses Wald intervals for fixed effects and direct response-scale targets. For long phylogenetic, spatial, animal-model, or relatedness fits, profile only the needed variance-component or correlation rows with the default `profile_engine = "auto"` first; direct scalar scale, SD, and correlation targets use the endpoint engine when no full-profile controls are supplied. Use `profile_engine = "tmbprofile"` or `profile_precision = "fast"` when you want the previous full-curve `TMB::tmbprofile()` route for comparison, diagnostics, or control tuning.

Value

A data frame with columns `parm`, `level`, `lower`, `upper`, `scale`, `transformation`, `tmb_parameter`, `index`, `method`, and `profile.engine`, `conf.status`, `profile.boundary`, and `profile.message`. Successful rows currently use `conf.status = "wald"`, `"profile"`, or `"bootstrap"`. Failed numeric profile rows use `"profile_failed"` with missing endpoints; profile rows mark intervals that land near a lower SD boundary or correlation boundary. Bootstrap interval results carry a `"bootstrap.diagnostics"` attribute with one diagnostic row per refit and target, including refit convergence, target availability, draw use, and the refit message.

Default uncertainty story

Use this recipe for ordinary first-week inference; it restates measured behaviour and does **not** claim nominal coverage on every route.

- Fixed effects and other routine Wald-ready targets: start with `confint(fit)` (`method = "wald"`). It is the fastest fitted-object route when `TMB::sdreport()` succeeded.
- Random-effect SDs and other direct variance-component targets: prefer `confint(fit, parm = ..., method = "profile")` after `profile_targets(fit)` shows the row is profile-ready. Profile re-optimizes nuisance parameters and is slower than Wald.
- Always read `conf.status` and `profile.boundary` on the returned table. A usable profile interval that lands on a variance-component boundary warns with class `drmTMB_profile_boundary_warning`; treat that interval as indicative of scale, not as a calibrated `level` interval (see Boundary intervals below).
- A computable interval is not coverage certification. Check the exact cell in Capabilities and limits before reporting a route as inference-ready.

For a short applied walkthrough, see the vignette *First-week intervals: fit, profile, and boundary*.

Boundary intervals

Every interval method is unreliable when a variance component approaches zero or a correlation approaches `+/-1`, and each warns about its own case.

`method = "wald"` flags a row with `conf.status = "wald_at_boundary"` and warns (class `drmTMB_wald_boundary_warning`) because a symmetric Wald interval undercovers under boundary (chi-square-mixture) inference. That warning recommends `method = "profile"`.

`method = "profile"` sets `profile.boundary = TRUE` and warns (class `drmTMB_profile_boundary_warning`) when it returns a usable interval that reaches a boundary. **A profile interval is not a repair for a boundary.** Conditional on that flag, a seeded 10-group random-effect SD gate measured coverage at 0.1021, 0.2387, and 0.8683 against a nominal 0.95, and 0 of 89 in a fourth cell where boundary hits are rare (0.09% of fits). The mechanism is a maximum likelihood random-effect SD biased 8.3%-15.8% low at small group counts, which anchors the interval low so that it misses from above.

Unconditionally – across all fits, flagged or not – the same gate measured 0.9248 against a nominal 0.95 over 400,000 attempts. A flagged interval is the bad case, not the typical one, and the flag is returned in the table so you can tell which case you are in.

Refitting the same design with `REML = TRUE` improved this without repairing it: over 400,000 paired replicates, profile coverage moved from 0.9248 to 0.9463 against a nominal 0.95, the SD point estimate's downward bias roughly halved (pooled -10.9% under maximum likelihood to -4.6% under

REML), and the upper-to-lower miss asymmetry fell from 5.7:1 to 2.0:1. Coverage conditional on the boundary flag improved but stayed well below nominal (0.74 to 0.83), so a flagged interval remains the bad case under either estimator. The default estimator is unchanged; this measurement covers the one design below only.

This is a property of profile intervals near a variance boundary rather than of `drmTMB`: `lme4::lmer` on the same data-generating process and seeds agreed on boundary incidence for 4000/4000 replicates and matched the conditional coverage to four decimal places. Treat a flagged interval as indicative of scale, not as a calibrated `level` interval, and prefer more groups where the design allows it.

These figures come from one measured design: Gaussian, a single random intercept on the mean, 10 groups, 4 or 10 observations per group, maximum likelihood. They describe that corner, and are not a general statement about every family, provider, or group count.

`method = "bootstrap"` flags a row with `conf.status = "bootstrap_at_boundary"` and warns (class `drmTMB_bootstrap_boundary_warning`) when at least 5% of the retained resamples land on the target's bound. Resampling does not repair a boundary: a percentile interval whose draws pile up at zero is reporting the constraint rather than the sampling distribution. Measured on three Gaussian random-intercept fits at `R = 200`, a true SD of 0 put 43% of draws on the bound and a true SD of 0.25 put 5% there with a lower endpoint of exactly zero, while a true SD of 0.9 put none there. The flag needs at least 20 retained draws to fire, because a share computed from a handful of resamples is noise; `bootstrap.n` reports how many were retained.

Rows with `conf.status = "profile_failed"` or `"clamp_limited"` also carry `profile.boundary = TRUE`, but return missing endpoints and are not warned about separately; read their `conf.status` and `profile.message` instead.

`check_drm()` does **not** assess any of this. It reads the fit, so a target whose point estimate sits well clear of zero passes every fit-level check while its interval is still flagged here. `check_drm()` emits an `interval_reliability_scope` note saying so; read `conf.status` before reporting an interval.

Profiling a structured `sigma` random-effect SD

A profile of a `sigma`-axis random-effect SD under `phylo()`, `animal()`, `relmat()`, `spatial()`, or `phylo_interaction()` is computable for `zero_one_beta()`, `nbinom2()`, and `zi_nbinom2()`, but the maximum likelihood point estimate it is built around is **biased low** for this class of cell: eleven of twelve retained ML estimates fell below truth in the sibling `nbinom2` provider cells (fit-level one-sided sign test `p = 0.0032`; `p = 0.0625` at cell level, once the shared data-generating process and random-number stream are respected). Two corrections that exist elsewhere for this bias do not reach these routes: `bias_correct` shifts only the `method = "wald"` centre and never a profile endpoint, and native scale-side REML is unavailable here because `drm_validate_reml_spec()` admits only Gaussian and binomial models. The profile interval therefore inherits the bias of its centre uncorrected. These routes carry point-fit recovery evidence only, and no seeded campaign has yet measured the coverage of a profile interval built on them, so treat the endpoints as a computed profile rather than calibrated inference.

Ordinal cutpoints

For a `cumulative_logit()` fit, the public ordinal targets are the ordered latent-logistic cutpoints `"ordinal:cutpoint:<label>"`, not the internal `"ordinal:theta_ord:<label>"` coordinates. Use `confint(fit, parm = "ordinal:cutpoint:<label>", method = "profile")` to obtain a pointwise likelihood-ratio interval on the cutpoint scale. The constrained engine fixes the cumulative

cutpoint while preserving strict ordering of every threshold; it does not change the fitted likelihood. These are not simultaneous bands or category-probability intervals, and they are not yet calibrated coverage claims. Raw `theta_ord` entries remain visible in `profile_targets()` only as internal diagnostics and reject interval methods. The fixed-effect coefficients on `mu` (and on any other fitted distributional parameter) remain available through the usual `confint(fit, parm = "fixed_effects")` or `confint(fit, parm = "fixef:mu:x")` calls.

References

The small-sample corrections applied by `small_sample_df` and `bias_correct` (whether the default `"location"` scope or the broader `"group"` scope) are *motivated* by established mixed-model theory but are simulation-calibrated, not derived: the `log(g / (g - 1))` SD-scale centre shift is about **twice** the leading-order REML SD correction (`0.5 * log(g / (g - 1))`), matching the larger ML shrinkage measured on these structured/bivariate cells (their effective df is well below `g - 1`). The magnitude's authority is the per-model-class simulation in `docs/design/219-structured-re-small-sample-bias-co` not a single source. Relevant references:

Restricted maximum likelihood and the variance-component bias that motivates the centre shift (REML debiases the variance by `g/(g-1)`; the shift here is on the SD log scale and ~2x the leading-order REML SD term):

Patterson, H. D., & Thompson, R. (1971). Recovery of inter-block information when block sizes are unequal. Biometrika, 58(3), 545-554. doi:10.1093/biomet/58.3.545

Harville, D. A. (1977). Maximum likelihood approaches to variance component estimation and to related problems. Journal of the American Statistical Association, 72(358), 320-338. doi:10.1080/01621459.1977.10480998

Searle, S. R., Casella, G., & McCulloch, C. E. (1992). Variance Components. New York: Wiley. doi:10.1002/9780470316856

Analytic and penalized first-order bias reduction of maximum likelihood estimates (the general framework an additive log-scale shift instantiates):

Cox, D. R., & Snell, E. J. (1968). A general definition of residuals. Journal of the Royal Statistical Society, Series B, 30(2), 248-265. doi:10.1111/j.25176161.1968.tb00724.x

Firth, D. (1993). Bias reduction of maximum likelihood estimates. Biometrika, 80(1), 27-38. doi:10.1093/biomet/80.1.27

The t-quantile / between-group effective degrees of freedom used by `small_sample_df`:

Satterthwaite, F. E. (1946). An approximate distribution of estimates of variance components. Biometrics Bulletin, 2(6), 110-114. doi:10.2307/3002019

Kenward, M. G., & Roger, J. H. (1997). Small sample inference for fixed effects from restricted maximum likelihood. Biometrics, 53(3), 983-997. doi:10.2307/2533558

Boundary regime (a variance component at or near zero), where neither the t-width nor the centre shift restores nominal coverage:

Self, S. G., & Liang, K.-Y. (1987). Asymptotic properties of maximum likelihood estimators and likelihood ratio tests under nonstandard conditions. Journal of the American Statistical Association, 82(398), 605-610. doi:10.1080/01621459.1987.10478472

Stram, D. O., & Lee, J. W. (1994). Variance components testing in the longitudinal mixed effects model. Biometrics, 50(4), 1171-1177. doi:10.2307/2533455

Parametric-bootstrap bias correction of mixed-model variance components, and the general delicacy of bootstrap bias estimation (a single-level parametric bootstrap does not recover the centre bias for these targets at small g):

Kubokawa, T., & Nagashima, B. (2012). Parametric bootstrap methods for bias correction in linear mixed models. Journal of Multivariate Analysis, 106, 1-16. doi:10.1016/j.jmva.2012.01.011

Efron, B., & Tibshirani, R. J. (1993). An Introduction to the Bootstrap. New York: Chapman & Hall.

Small-sample variance-component (repeatability) interval coverage in ecology and evolution:

Wolak, M. E., Fairbairn, D. J., & Paulsen, Y. R. (2012). Guidelines for estimating repeatability. Methods in Ecology and Evolution, 3(1), 129-137. doi:10.1111/j.2041210X.2011.00125.x

See Also

The tier definitions and per-cell evidence behind these interval targets, including random-effect standard-deviation rows, are curated in `vignette("capability-and-limits", package = "drmTMB")`: `confint()` computes generically for any target, and the tier a given cell belongs to is a documentation-level curation, not a runtime guard.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
confint(fit)
confint(fit, parm = "variance_components")
confint(fit, parm = "sigma", method = "profile")
confint(
  fit, parm = "sigma", method = "profile",
  profile_engine = "tmbprofile", profile_precision = "fast"
)
set.seed(1)
confint(fit, parm = "sigma", method = "bootstrap", R = 99)
```

`confint.drmTMB_julia` *Inspect legacy interval output from a halted Julia bridge*

Description

The Julia bridge is halted/deferred future work and is not a current fitting or inference route. This method is retained only for inspecting existing `drmTMB_julia` objects; use native TMB fits for new analyses.

Usage

```
## S3 method for class 'drmTMB_julia'
confint(
  object,
  parm = NULL,
```

```
  level = 0.95,
  method = c("wald", "profile", "bootstrap"),
  R = 199L,
  seed = NULL,
  threads = FALSE,
  ...
)
```

Arguments

object	A drmTMB_julia fit.
parm	Optional target selection. For "wald", compact coefficient labels ("mu:x") or full names ("fixef:mu:x"); for "profile" / "bootstrap", supported SD target names such as "sd:mu:phylo(1 | species)" or, for q = 4 bivariate fits, "sd:sigma1:phylo(1 | species)".
level	Confidence level.
method	"wald" (default), "profile", or "bootstrap".
R	Bootstrap replicate count (used only when method = "bootstrap").
seed	Optional bootstrap seed.
threads	Logical; request Julia-side threaded inference for the profile / bootstrap path.
...	Unused.

Details

For a legacy engine = "julia" fit, confint() exposes two interval families:

- method = "wald" (the default) builds symmetric Wald intervals for the fixed-effect coefficients (mu, sigma, ...) on the linear-predictor (link) scale, using the fixed-effect covariance DRM.jl marshals back through the bridge (vcov(object)). This mirrors the native drmTMB Wald path, whose fixed-effect rows are also reported on the link scale.
- method = "profile" / method = "bootstrap" re-enter DRM.jl's inference primitive for supported phylogenetic SD targets, transformed back to the positive response scale. The current R bridge exposes the univariate Gaussian sd:mu:phylo(1 | species) target and the four bivariate q = 4 targets sd:mu1:*, sd:mu2:*, sd:sigma1:*, and sd:sigma2:*.

Value

A confidence-interval data frame with the shared parm, level, lower, upper, scale, transformation, tmb_parameter, index, method, and conf.status columns.

confint.drm_pair_association *Confidence intervals for a frozen-margin association*

Description

For admitted fixed-effect complete-pair association routes, `vcov()` returns the association-coefficient block of the two-stage Godambe sandwich and this method returns corresponding Wald intervals. `type = "alpha"` returns the coefficient-scale intervals. For an intercept-only association, `type = "eta"` monotonically transforms its link-scale limits to the bounded latent-association scale. A covariate-varying association has no single eta; use `predict.drm_pair_association()` with `newdata` for row-specific eta uncertainty. Every route is interval-feasible when its fit-specific covariance diagnostics pass. Coverage evidence currently promotes only the retained Bernoulli x ordinary-NB2 intercept domain to inference-ready with caveats; other routes receive an experimental-coverage warning.

Usage

```
## S3 method for class 'drm_pair_association'
confint(object, parm = NULL, level = 0.95, type = c("alpha", "eta"), ...)
```

Arguments

`object`	A fitted `drm_pair_association` object.
`parm`	Association coefficients to include. `NULL` (the default) or `"alpha"` selects all association coefficients; a numeric or character subset may also be supplied.
`level`	Confidence level in `(0, 1)`.
`type`	`"alpha"` for coefficient-scale intervals or `"eta"` for the bounded latent association of an intercept-only model.
`...`	Reserved for future options.

Value

A matrix with Wald confidence limits on the requested scale.

See Also

`vcov()`, `predict.drm_pair_association()`

Examples

```
set.seed(20260801)
dat <- data.frame(x = rnorm(100))
z_1 <- rnorm(100)
z_2 <- 0.35 * z_1 + sqrt(1 - 0.35^2) * rnorm(100)
dat$continuous <- 0.2 + 0.4 * dat$x + z_1
dat$binary <- as.integer(z_2 > qnorm(0.55))
```

```
assoc <- biv_associate(
  bf(mu = continuous ~ x, sigma = ~ 1),
  bf(mu = binary ~ x),
  family = list(gaussian(), binomial()), data = dat
)
confint(assoc, type = "eta")
```

`corpair` *Latent random-effect correlation formula marker*

Description

`corpair()` marks predictor-dependent latent random-effect correlations. It is distinct from residual `rho12` and from the `corpairs()` extractor. The first fitted paths are q=2 location-location cases for matching labelled `mu1`/`mu2` random intercepts: `corpair(id, level = "group", block = "p", from = "mu1", to = "mu2") ~ x` and `corpair(species, level = "phylogenetic", block = "p", from = "mu1", to = "mu2") ~ ecology`. Predictors must be constant within the grouping factor. Spatial `corpair()` regressions, location-scale `corpair()` regressions, scale-scale `corpair()` regressions, and q=4 `corpair()` regressions remain planned. The phylogenetic q=2 route uses a positive-definite two-field loading contract for the whole tree-coupled species block.

Usage

```
corpair(
  group,
  level = NULL,
  block = NULL,
  class = NULL,
  from = NULL,
  to = NULL
)
```

Arguments

`group`	Grouping factor for the latent covariance block.
`level`	Optional latent correlation level, such as `"group"`, `"phylogenetic"`, or `"spatial"`.
`block`	Optional covariance-block label, such as `"p"`.
`class`	Optional latent correlation class: `"location-location"`, `"location-scale"`, or `"scale-scale"`. This is an extraction-oriented shorthand and is not the first fitted q=4 correlation-regression target.
`from`, `to`	Optional endpoint-specific distributional parameters, such as `"mu1"` and `"mu2"` for the first fitted q=2 targets, or `"mu1"` and `"sigma2"` for later location-scale targets.

Value

A formula marker; never evaluated by users.

Examples

```
bf(corpair(id, level = "group", block = "p",
  from = "mu1", to = "mu2") ~ ecology)

# fitted q=2 phylogenetic sibling
bf(corpair(species, level = "phylogenetic", block = "p",
  from = "mu1", to = "mu2") ~ ecology)
```

`corpairs` *Extract fitted correlation pairs*

Description

`corpairs()` returns a long table of fitted correlation pairs from a `drmTMB` model. The current implementation reports correlations that are already fitted elsewhere: residual bivariate `rho12`, ordinary univariate group-level `mu` random-effect correlations, matched univariate and same-response bivariate `mu`/`sigma` covariance blocks, matched bivariate `mu1`/`mu2` random-intercept and slope-only covariance blocks, and matched bivariate `sigma1`/`sigma2` random-intercept and slope-only covariance blocks from `corpars`, plus fitted bivariate phylogenetic, coordinate-spatial, animal-model, and `relmat()` correlation rows. Full q4 phylogenetic, coordinate-spatial, animal-model, and `relmat()` blocks report six derived endpoint correlations; block-diagonal q4 fallback fits report the direct `mu1`/`mu2` and `sigma1`/`sigma2` block correlations.

Usage

```
corpairs(object, ...)

## S3 method for class 'drmTMB'
corpairs(
  object,
  level = NULL,
  group = NULL,
  block = NULL,
  class = NULL,
  conf.int = FALSE,
  conf.level = 0.95,
  method = "profile",
  trace = FALSE,
  ...
)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Additional arguments passed to `TMB::tmbprofile()` when `conf.int = TRUE`.
`level`	Optional character vector of correlation levels to keep, such as `"residual"`, `"group"`, `"phylogenetic"`, or `"spatial"`.

group	Optional character vector of grouping factors to keep, such as "id". Residual rows have no grouping factor and are removed by this filter.
block	Optional character vector of covariance-block labels to keep, such as "p". Residual rows have no block label and are removed by this filter.
class	Optional character vector of pair classes to keep, such as "residual" or "mean-slope". Location aliases such as "location-location" and "location-scale" are accepted as filters for the current "mean-mean" and "mean-scale" rows.
conf.int	Logical; include profile-likelihood confidence intervals where the correlation target is currently profile-ready. Unsupported derived targets receive an explicit interval status instead of silent missing bounds.
conf.level	Confidence level used when conf.int = TRUE. This is named separately from the level filter to avoid ambiguity with correlation levels such as "phylogenetic".
method	Interval method used when conf.int = TRUE. Only "profile" is currently supported for correlation-pair intervals.
trace	Logical; passed to TMB::tmbprofile() when profile intervals are requested.

Details

Use corpairs() when the question is about correlations among fitted residual, ordinary group-level, phylogenetic, coordinate-spatial, animal-model, or relmat() latent effects. Use rho12() when the only target is the residual correlation curve of a bivariate model.

The table is intentionally more explicit than rho12() or corpars because double-hierarchical, phylogenetic, spatial, animal-model, and lower-level relatedness models can contain several scientifically different correlations. Profile intervals are opt-in and can be slow; filter with level, group, block, or class before requesting conf.int = TRUE on large models. Bootstrap intervals are not a corpairs() route.

Value

A data frame with one row per fitted correlation pair or pair summary. Predictor-dependent rho12 is summarized by its mean, minimum, and maximum over the fitted rows. Rows include conf.status and interval_source so point-only and interval-aware pair tables use the same provenance vocabulary as prediction tables.

Examples

```
set.seed(1)
n <- 40
x <- rnorm(n)
z1 <- rnorm(n)
z2 <- rnorm(n)
mu1 <- 0.2 + 0.5 * x
mu2 <- -0.1 + 0.4 * x
sigma1 <- exp(-0.2 + 0.15 * z1)
sigma2 <- exp(0.1 - 0.1 * z2)
rho <- 0.35
e1 <- rnorm(n)
e2 <- rho * e1 + sqrt(1 - rho^2) * rnorm(n)
```

```
dat <- data.frame(
  y1 = mu1 + sigma1 * e1,
  y2 = mu2 + sigma2 * e2,
  x = x,
  z1 = z1,
  z2 = z2
)
fit <- drmTMB(
  bf(
    mu1 = y1 ~ x,
    mu2 = y2 ~ x,
    sigma1 = ~ z1,
    sigma2 = ~ z2,
    rho12 = ~ 1
  ),
  family = c(gaussian(), gaussian()),
  data = dat
)
pairs <- corpairs(fit)
pairs
corpairs(fit, level = "residual")
corpairs(fit, level = "residual", conf.int = TRUE)
```

`cumulative_logit` *Cumulative logit ordinal response family*

Description

`cumulative_logit()` defines a one-response ordinal model for ordered categories. The first implemented path uses a location formula `mu ~ ...` and ordered cutpoints with a fixed latent logistic scale. The location intercept is dropped internally, as in standard cumulative-link models, because a free location intercept and free cutpoints are not jointly identifiable.

Usage

```
cumulative_logit()
```

Details

The implemented contract is `Pr(y_i <= k) = logit^-1(theta_k - mu_i)`, with `mu_i = X_mu[i, ] beta_mu` and `theta[1] < theta[2] < ... < theta[K - 1]`. `fitted()` returns the expected ordered-category score, `sum_k k * Pr(y_i = k)`. Ordinal scale or discrimination formulas are planned but not exposed in this first implementation.

Value

A `drm_family` object.

Examples

```
cumulative_logit()
```

`drmTMB` *Fit a distributional regression model with TMB*

Description

`drmTMB()` is the main model-fitting entry point. The current implementation supports univariate Gaussian location-scale models, univariate Student-t and skew-normal location-scale-shape models, lognormal location-scale models, Gamma mean-CV models for positive responses, Tweedie mean-scale-power models for non-negative semicontinuous responses, beta mean-scale models for strict proportions, zero-one beta mean-scale-boundary models for continuous proportions with structural exact zeroes or ones, beta-binomial mean-overdispersion models for success counts, fixed-effect Bernoulli/binomial event-probability models, fixed-effect cumulative-logit ordinal location models, fixed-effect Poisson mean, zero-inflated Poisson, negative-binomial mean-dispersion, zero-inflated negative-binomial mean-dispersion, zero-truncated negative-binomial mean-dispersion, and hurdle negative-binomial mean-dispersion models for counts. Student-t, lognormal, Gamma, beta, ordinary Poisson, ordinary negative-binomial, beta-binomial, and zero-truncated negative-binomial `mu` formulas support ordinary unlabelled random intercepts and independent numeric slopes where documented. Every univariate `mu` formula may include a standard R `offset()` term, which enters the location linear predictor as a known constant. Read it on the family's own link scale: for log-link families (Poisson, ordinary negative-binomial, Gamma, Tweedie) `offset(log(exposure))` is the usual exposure or effort rate model; for identity-link families (Gaussian, Student-t, lognormal, skew-normal) an offset is a known additive shift of the mean; and for logit-link families (Bernoulli/binomial, beta-binomial, beta, zero-one-beta, ordinal cumulative-logit) it is a known log-odds shift, which is a calibration term rather than an exposure. A zero-one-beta offset shifts only the interior beta component, leaving `zoi` and `coi` unchanged, and an ordinal offset shifts the latent location against fixed cutpoints. Zero-truncated and hurdle negative-binomial responses, every bivariate family, and Gaussian sufficient-statistic aggregation reject offsets: the first two renormalise their observed mean over a restricted support, so an exposure term would not scale the reported mean, and the others have no per-response offset contract yet. Offsets remain restricted to `mu`; other distributional parameters reject them. drmTMB also supports Gaussian random intercepts, independent numeric random slopes, and labelled or unlabelled correlated numeric random intercept-slope blocks in the location formula, known sampling covariance through `meta_V(V = V)` with deprecated `meta_known_V(V = V)` retained as a compatibility alias, residual-scale random intercepts and independent numeric random slopes in the scale formula, labelled `mu`/`sigma` random-intercept covariance blocks, and one or more group-level random-effect scale formulae such as `sd(id) ~ x_group`, plus phylogenetic random intercepts, one numeric phylogenetic random slope, and `sd_phylo(species) ~ x_species` direct-SD models in univariate Gaussian location formulas, Gaussian `mu` animal-model and user-supplied relatedness random intercepts and one numeric random slope, matching bivariate Gaussian `mu1`/`mu2` location formulas, and matching labelled bivariate Gaussian `mu1`/`mu2`/`sigma1`/`sigma2` phylogenetic location-scale blocks, coordinate-based spatial random intercepts and one numeric coordinate-spatial slope in univariate Gaussian `mu`, fixed-effect bivariate Gaussian distributional models, and matched labelled bivariate Gaussian `mu1`/`mu2`, `sigma1`/`sigma2`, and same-response `mu`/`sigma` random-intercept covariance blocks, including the

first matching slope-only mu1/mu2 covariance block, the first all-four q=4 ordinary random-intercept covariance blocks, and predictor-dependent q=2 ordinary or phylogenetic corpair() regressions. Bivariate Gaussian location formulas may be written explicitly as mu1 = y1 ~ ..., mu2 = y2 ~ ..., or with mvbind(y1, y2) ~ ... shorthand when both responses share the same location predictors.

Usage

```
drmTMB(
  formula,
  family = stats::gaussian(),
  data,
  weights = NULL,
  control = list(),
  impute = NULL,
  missing = miss_control(),
  engine = c("tmb", "julia"),
  REML = FALSE,
  penalty = NULL,
  estimator = c("ml", "mspl"),
  ...
)
```

Arguments

formula	A drm_formula object created by drm_formula() or bf().
family	A response family, such as stats::gaussian(), student(), skew_normal(), lognormal(), stats::Gamma() with link = "log", tweedie(), beta(), zero_one_beta(), beta_binomial(), stats::binomial() with link = "logit", "probit", or "cloglog", cumulative_logit(), stats::poisson() with link = "log", nbinom2(), truncated_nbinom2(), or biv_gaussian(). Adding zi ~ predictors to a Poisson or nbinom2() model fits the corresponding zero-inflated count model. Adding hu ~ predictors to a truncated_nbinom2() model fits a hurdle count model whose nonzero counts use the zero-truncated NB2 component. The current bivariate Gaussian engine also accepts family = c(gaussian(), gaussian()) and family = list(gaussian(), gaussian()).
data	A data frame.
weights	Optional non-negative likelihood weights. These are row log-likelihood multipliers, not known sampling variances. For meta-analytic sampling variance or covariance, use meta_V() in the model formula instead.
control	Optional list passed to stats::nlminb(), or a drm_control() object when optimizer settings and fitted-object storage choices should be supplied together.
impute	Optional one-element named list of predictor models for the current missing-predictor routes. Bare formulas such as list(x = x ~ z) define Gaussian models for numeric missing predictors. Use impute_model() for explicit predictor families, such as list(treatment = impute_model(treatment ~ z, family = binomial())) for a binary predictor, list(score = impute_model(score ~ z, family = cumulative_logit())) for an ordered predictor, or list(habitat

= impute_model(habitat ~ z, family = categorical())) for an unordered predictor, or list(cover = impute_model(cover ~ z, family = beta())) for a strict proportion predictor in (0, 1), or list(cover = impute_model(cover ~ z, family = zero_one_beta())) for a boundary proportion predictor in [0, 1], or list(cover = impute_model(success ~ z, family = beta_binomial(), trials = trials)) for a denominator-aware success/trial proportion predictor, or list(abundance = impute_model(abundance ~ z, family = poisson())) or list(abundance = impute_model(abundance ~ z, family = nbinom2())) for a count predictor, including list(abundance = impute_model(abundance ~ z, family = truncated_nbinom2())) for a positive zero-truncated count predictor, or list(biomass = impute_model(biomass ~ z, family = lognormal())) for a positive continuous predictor, or list(biomass = impute_model(biomass ~ z, family = Gamma(link = "log"))) for a Gamma positive continuous predictor, or list(biomass = impute_model(biomass ~ z, family = tweedie())) for a non-negative semi-continuous predictor with exact zeros. Grouped Gaussian covariate models use syntax such as list(x = x ~ z + (1 | group)); structured Gaussian covariate models use explicit syntax such as list(x = x ~ z + relmat(1 | line, Q = Q)). Most fitted routes use a univariate Gaussian formula containing one mi(x) location term and missing = miss_control(predictor = "model"). The first non-Gaussian response route also supports family = poisson() with one fixed-effect binary mi() predictor modelled by family = binomial().

`missing` Missing-data policy created by miss_control(). The default keeps the existing complete-case behaviour. In the current fitted slices, missing = miss_control(response = "include") is implemented only for univariate Gaussian response masks and bivariate Gaussian partial-response rows without dense known covariance. missing = miss_control(predictor = "model") is implemented for one mi() missing predictor in a univariate Gaussian location model: numeric Gaussian predictors may use a fixed-effect, one random-intercept, or one intercept-only structured Gaussian impute formula; binary, ordered categorical, unordered categorical, strict beta/proportion, zero-one beta boundary-proportion, beta-binomial denominator-aware proportion, Poisson, negative-binomial, or zero-truncated negative-binomial count, lognormal positive continuous, Gamma positive continuous, and Tweedie semi-continuous predictors may use one fixed-effect family-aware impute_model(). The first Poisson-response route supports one fixed-effect binary missing predictor with a Bernoulli/logit impute_model(), complete count responses, and no zero-inflation, random, or structured response terms.

`engine` Computational engine. The default "tmb" uses the native drmTMB TMB backend. The "julia" compatibility bridge is halted and deferred for future work; it is retained only so existing objects and code can be inspected, not as a current fitting route.

`REML` Logical; use restricted maximum likelihood where the selected engine supports it. Native engine = "tmb" restricts the likelihood by marginalising the admitted fixed-effect mean coefficients. Validated Gaussian routes include ordinary random effects; mean-side phylo(), spatial(), animal(), and relmat() effects; known sampling covariance through meta_V(); non-unit likelihood weights; and selected scale-side random or structured effects. Mean-side spatial(), animal(), and relmat() REML is deliberately limited to an unlabelled in-

tercept or independent intercept-plus-one-numeric-slope term with a constant residual scale (`sigma ~ 1`, with no sigma random effect). Slope-only, labelled, multiple-slope, and matched non-phylogenetic mean-scale REML remain outside that route. The two bivariate non-phylogenetic exceptions are matching labelled fixed-covariance q2 location intercepts for `spatial(1 | p | site, coords = coords)` and `relmat(1 | p | id, K = K)` in `mu1` and `mu2`, with intercept-only `sigma1`, `sigma2`, and `rho12`, complete response pairs, unit weights, no known `meta_V()` covariance, and no additional ordinary random effect, direct-SD formula, or `corpair()` regression. Both have point-fit-recovery evidence only; bivariate `relmat(..., Q = Q)` REML remains deferred. Aggregation and ordinary direct `sd()` scale formulae also remain unsupported under REML. For `binomial()` models, the bounded native route requires exactly one ordinary unlabelled `mu` random intercept or exactly one independent numeric slope. Fixed-only and multiple-term binomial models, correlated or labelled covariance blocks, structured effects, missing-data engines, and other extensions are not admitted under REML; use `REML = FALSE` for those models. The binomial route has diagnostic parity and finite-uncertainty evidence, not calibrated interval or coverage evidence. Every other non-Gaussian family rejects `REML = TRUE`; use `REML = FALSE`. The package-private AGHQ plus Cox-Reid (O3) estimator is not a `drmTMB()` argument and is not what `REML = TRUE` runs. Public cumulative-logit random-slope fits remain maximum likelihood (`point_fit_recovery`). The halted `engine = "julia"` compatibility bridge is not a supported estimator or REML route. Use native `engine = "tmb"` for fitting and use `REML = FALSE` for likelihood-ratio tests, AIC/BIC comparisons across different fixed-effect formulas, non-binomial non-Gaussian models, and currently unsupported extensions.

`penalty` Optional penalty / prior built by `drm_phylo_penalty()`, or `NULL` (default) for plain maximum likelihood. A non-`NULL` penalty switches the fit to a penalized / maximum-a-posteriori (MAP) estimator that regularises a weakly-identified phylogenetic standard deviation; the fit is labeled `MAP` and `logLik()` returns the unpenalized data log-likelihood. Native `engine = "tmb"` only.

`estimator` Estimator for the native TMB route. The default `"ml"` preserves ordinary maximum likelihood. Experimental `"mspl"` implements the clean-room maximum softly-penalized likelihood criterion for one complete Bernoulli or grouped-binomial model with one ordinary q = 1 or correlated q = 2 grouping block. Supported links are `"logit"`, `"probit"` and `"cloglog"`. Kosmidis and Firth (2021, Theorem 1 and Section 3.1) prove the Jeffreys penalty gives finite estimates for any link whose working weight vanishes in both tails, which these three do; drmTMB's own TMB-Laplace evidence for them is recorded in `docs/dev-log/simulation-artifacts/`. Three limits are worth stating. The soft-penalty scale below is a *logit* delta-method constant, kept unchanged for all three links because using it for probit and cloglog was measured to cost about 1% of one standard error – measured at `q = 1` with two fixed-effect columns, and not at `q = 2`. Under deep separation with a random slope the standard error is frequently unavailable for **every** link, logit included (see the `NA` note below); in the most extreme cells measured this affected the large majority of converged fits. And the evidence behind the two new links used **Bernoulli** responses with two fixed-effect columns, so grouped-binomial and wider designs inherit the `n_eff` extrapolation described next without direct measurement, for any link. For grouped binomial data, the soft-penalty

scale `c_n = 2 * sqrt(p / n_eff)` uses `n_eff = sum(trials * frequency)`, the retained total number of Bernoulli trials, in place of the source paper's row count `n`; the two agree exactly when every row is a single Bernoulli trial (`trials = frequency = 1`). This is a deliberate grouped-data extension, not a claim that Sterzinger and Kosmidis's (2023) results carry over unchanged. `vcov()` and the `std_error` column of `summary()` are available: they invert the Hessian of the *unpenalized* Laplace log-likelihood evaluated at the MSPL estimate, so the penalty is used to obtain a finite estimate but not to describe sampling variability. **A standard error is reported; an interval is not claimed.** Likelihood comparisons (`logLik()`, `AIC`, `BIC`, `anova()`), profiles, and `confint()` remain deliberately unavailable. Do not form `coef ± 1.96 * se` by hand: Kosmidis and Firth show that Wald intervals in this setting fail to cover regardless of the nominal level, a failure that persists even for profile penalized-likelihood intervals. When the information matrix is not positive definite the standard errors are `NA` with a `drmTMB_mspl_wald_unavailable` warning rather than a fabricated number.

`...` Reserved for future model options.

Value

A `drmTMB` fit object.

Examples

```
set.seed(20260525)
dat <- data.frame(
  y = 0.2 + 0.6 * seq(-1, 1, length.out = 24) + rnorm(24, sd = 0.5),
  x = seq(-1, 1, length.out = 24)
)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
fit
```

`drm_control` *Control fitting and fitted-object storage*

Description

`drm_control()` collects optimizer settings and storage choices for `drmTMB()`. Use `optimizer` for settings passed to `stats::nlminb()`, or `optimizer_preset` for named `nlminb()` budgets that keep ordinary defaults fast while making complex refits easier to write. Use the storage flags when a fitted object should keep less R-side state, for example during large-data experiments where the original data frame and TMB automatic-differentiation object are expensive to retain.

Usage

```
drm_control(
  optimizer = list(),
  se = TRUE,
```

```
  se_report_covariance = TRUE,
  se_skip_delta_method = FALSE,
  se_group_sd = FALSE,
  keep_data = TRUE,
  keep_model_frame = TRUE,
  keep_tmb_object = TRUE,
  sparse_fixed = FALSE,
  aggregate_gaussian = FALSE,
  logsigma_clamp = c(-12, 12),
  logsigma_clamp_margin = 3,
  optimizer_preset = c("default", "careful", "robust"),
  multi_start = 1L,
  fallback_optimizer = NULL
)
```

Arguments

optimizer	Named list passed to the control argument of stats::nlminb().
se	Logical; compute standard errors and fixed-effect covariance with TMB::sdreport() after optimization. Set to FALSE to keep fitted coefficients, fitted values, residuals, predictions, simulations, and profile-likelihood paths while skipping Wald standard errors, stats::vcov(), and Wald confidence intervals. Profile-likelihood intervals still require keep_tmb_object = TRUE.
se_report_covariance	Logical; passed to the getReportCovariance argument of TMB::sdreport(). The default TRUE builds the full covariance matrix of every ADREPORTed quantity. Models with a direct-SD surface (sd_phylo(...) ~ .) report one standard deviation *per group*, so this matrix is n_group x n_group and its memory cost grows with the square of the number of groups; at ten thousand tips it dominates the fit. Set to FALSE to keep per-quantity standard errors while skipping their joint covariance.
se_skip_delta_method	Logical; passed to the skip.delta.method argument of TMB::sdreport(). Set to TRUE to skip standard errors for ADREPORTed quantities entirely while retaining fixed-effect standard errors and stats::vcov(). This is the cheapest route to Wald inference on the fixed effects for large structured models.
se_group_sd	Logical; report delta-method standard errors for the *per-group* direct-SD surface (sd_phylo(...) ~ .). Defaults to FALSE. The surface has one standard deviation per group, so ADREPORTing it makes the joint ADREPORT covariance n_group x n_group. Under REML = TRUE the fixed effects are integrated into the Laplace random block and stats::vcov() reads exactly that joint covariance, so at ten thousand tips a bivariate fit needs tens of gigabytes for it. The fitted per-group standard deviations themselves are always available (they are recomputed from the parameters); only their standard errors are opt-in. Set to TRUE to restore the pre-0.3.0 behaviour.
keep_data	Logical; keep the complete-case model data in the fitted object. Set to FALSE to drop fit$data and fit$model$data after fitting. Prediction, fitted values,

residuals, simulation, and basic summaries still use the stored model matrices and response vectors.

`keep_model_frame`
: Logical; keep model frames in the fitted object. Set to `FALSE` to drop `fit$model$model_frame` and random-effect scale model frames after fitting. Prediction, fitted values, residuals, simulation, `sigma()`, `rho12()`, `corpairs()`, and `check_drm()` use stored model matrices, terms, response vectors, offsets, and response-name metadata.

`keep_tmb_object`
: Logical; keep the TMB automatic-differentiation object in `fit$obj`. Set to `FALSE` to reduce fitted-object size after optimization. `check_drm()` will then report the fixed-gradient check as a note because it cannot re-evaluate the gradient without `fit$obj`, and profile-likelihood confidence intervals will be unavailable.

`sparse_fixed`
: Logical; opt-in control for sparse fixed-effect design matrices. The first fitted path is limited to univariate Gaussian `mu` fixed effects with no random effects and intercept-only `sigma`.

`aggregate_gaussian`
: Logical; opt-in control for sufficient- statistic row aggregation in univariate Gaussian fixed-effect models. The first fitted path rejects random effects, structured effects, known sampling covariance, bivariate models, non-Gaussian families, non-unit likelihood weights, and combined sparse fixed-effect matrices.

`logsigma_clamp`
: Numeric `c(lo, hi)` band, or `NULL`. Bounds the per-observation Gaussian `log(sigma)` with an identity-in-band soft-clamp, a numerical guard against scale overflow on near-degenerate per-group scale models. The default `c(-12, 12)` is identity for any standardized response; widen it for legitimately huge-variance unstandardized data, or set `NULL` to disable the guard entirely. It is a numerical guard only and does not change identifiability.

`logsigma_clamp_margin`
: Positive number; the soft-clamp saturation margin beyond `logsigma_clamp` (default `3`, saturating to `[-15, 15]` for the default band). Ignored when `logsigma_clamp = NULL`.

`optimizer_preset`
: Optimizer-budget preset. `"default"` adds no optimizer controls, `"careful"` sets `iter.max = 1000` and `eval.max = 1000`, and `"robust"` sets `iter.max = 5000` and `eval.max = 5000`. The optimizer escalates this ladder automatically when a preset does not converge.

`multi_start`
: Whole number `>= 1` (default `1`). With `multi_start > 1`, each optimizer preset is run from `multi_start` starting points – the principled start plus reproducibly perturbed starts – and the lowest- objective result is kept. Opt-in robustness for weakly identified models; `multi_start = 1` is the single-start fit and is unchanged.

`fallback_optimizer`
: `NULL` (default) or one `stats::optim()` method (`"BFGS"`, `"L-BFGS-B"`, `"Nelder-Mead"`, `"CG"`). When set, and no `nlminb()` preset converges, `drmTMB()` tries this optimizer as a final attempt; a different algorithm can succeed on a numerically awkward but identified problem. Opt-in; `NULL` keeps the `nlminb()`-only ladder.

Details

For optimizer-only settings, `control = list(eval.max = 1000)` remains valid. When using `drm_control()`, put optimizer arguments inside `optimizer = list(...)`; do not pass `eval.max` directly to `drm_control()`. Presets `"careful"` and `"robust"` expand to explicit `iter.max` and `eval.max` controls for `nlminb()`. Values in `optimizer` override values from the selected preset.

When the selected preset uses the standard `nlminb()` budget and no explicit optimizer controls, `drmTMB()` escalates the preset ladder (`"default"` -> `"careful"` -> `"robust"`) when an attempt either raises an error (such as a non-finite gradient) or does not converge cleanly (a nonzero convergence code or a non-finite objective). The first cleanly-converged attempt is returned; if no preset converges, the best (lowest-objective) attempt is returned and the fit-time convergence warning flags it. Every attempted preset is recorded in `fit$optimizer_attempts` and the selected one in `fit$optimizer_used`.

Value

A `drm_control` object.

Examples

```
dat <- data.frame(y = rnorm(20), x = rnorm(20))
fit <- drmTMB(
  bf(y ~ x, sigma ~ 1),
  data = dat,
  control = drm_control(
    optimizer_preset = "careful",
    se = FALSE,
    keep_data = FALSE,
    keep_model_frame = FALSE,
    keep_tmb_object = FALSE
  )
)
```

---

drm_formula *Build a drmTMB formula object*

Description

`drm_formula()` captures the formulae that define a `drmTMB` model. The family decides which distributional parameters are valid; `drm_formula()` only records the user's intended formulas. `bf()` is a short alias.

Usage

```
drm_formula(...)

bf(...)
```

Arguments

`...`	Formulae or named formulae. The unnamed response formula is interpreted as the location formula for a univariate model. For bivariate models, prefer explicit `mu1 = y1 ~ ...` and `mu2 = y2 ~ ...` formulas.

Value

A `drm_formula` object.

Examples

```
drm_formula(y ~ x, sigma ~ z)
drm_formula(y ~ x + (1 | id), sigma ~ z, sd(id) ~ x_group)
drm_formula(
  y ~ x + (1 | id) + (1 | site),
  sigma ~ z,
  sd(id) ~ x_group,
  sd(site) ~ site_type
)
drm_formula(
  mu1 = y1 ~ x1 + x2,
  mu2 = y2 ~ x1,
  sigma1 = ~ x1,
  sigma2 = ~ x2,
  rho12 = ~ x1 + x2
)
drm_formula(mvbind(y1, y2) ~ x1 + x2, sigma1 = ~ x1, sigma2 = ~ x2)
```

`drm_phylo_penalty` *Penalty / prior specification for a phylogenetic location-scale fit*

Description

Builds an optional penalty (a weakly-informative prior) for the phylogenetic standard deviations and, optionally, the phylogenetic cross-parameter correlation of a `drmTMB()` fit. Passing the result to the `penalty` argument of `drmTMB()` switches the estimator from plain maximum likelihood to a penalized / maximum-a-posteriori (MAP) estimator.

Usage

```
drm_phylo_penalty(sd_u = 1, sd_alpha = 0.05, cor_sd = NULL)
```

Arguments

`sd_u`, `sd_alpha`	Penalised-complexity prior scale and tail probability for each phylogenetic SD: a priori `P(sd > sd_u) = sd_alpha`. `sd_u` must be positive and `sd_alpha` must lie in `(0, 1)`.

`cor_sd` Optional standard deviation of a mean-zero normal penalty on the phylogenetic cross-parameter correlation parameter. `NULL` (the default) applies no correlation penalty. A non-`NULL` `cor_sd` requires a coupled phylogenetic model with at least two phylogenetic SDs (for example a coupled location-scale or bivariate fit); a location-only phylogenetic model has a single phylogenetic SD and no correlation parameter to penalize, so `drmTMB()` errors rather than silently ignoring `cor_sd`.

Details

The standard-deviation penalty is a penalised-complexity (PC) prior (Simpson et al. 2017): an exponential prior on the SD scale with mass at zero, which regularises a weakly-identified phylogenetic SD (for example a scale-side phylogenetic field at about one observation per tip) toward the simpler "no phylogenetic variance" model. The rate is `lambda = -log(sd_alpha) / sd_u` so that, a priori, `P(sd > sd_u) = sd_alpha`. The optional correlation penalty is a mean-zero normal on the unconstrained phylogenetic correlation parameter.

A penalized fit is a MAP point estimate, not a maximum-likelihood fit: its standard errors are credible-interval-shaped, and likelihood-ratio tests or AIC across penalized fits are not standard. `logLik()` returns the unpenalized data log-likelihood; the penalty contribution is stored separately on the fit as `fit$phylo_penalty`.

Value

An object of class `drm_phylo_penalty`.

References

Simpson, D., Rue, H., Riebler, A., Martins, T. G., & Sorbye, S. H. (2017). Penalising model component complexity: a principled, practical approach to constructing priors. Statistical Science, 32(1), 1-28.

Chung, Y., Rabe-Hesketh, S., Dorie, V., Gelman, A., & Liu, J. (2013). A nondegenerate penalized likelihood estimator for variance parameters in multilevel models. Psychometrika, 78(4), 685-709.

Examples

```
# Penalised-complexity prior: a priori P(phylogenetic SD > 1) = 0.05.
pen <- drm_phylo_penalty(sd_u = 1, sd_alpha = 0.05)
pen$rate

# Also penalize the phylogenetic correlation in a coupled location-scale or
# bivariate phylogenetic model.
pen_cor <- drm_phylo_penalty(sd_u = 1, sd_alpha = 0.05, cor_sd = 0.5)
pen_cor$cor_sd
```

`drm_phylo_penalty_sweep` *Prior-sensitivity sweep for the phylogenetic correlation penalty*

Description

Refits a penalized (MAP) phylogenetic model across a range of `cor_sd` values so you can see whether a weakly identified coupling is data-informed or prior-shaped. There is no universal `cor_sd`: a coupling that is stable across the sweep is data-informed, while one that tracks `cor_sd` is prior-shaped. This is the sweep the penalized/MAP workflow asks you to run; it is most informative for coupled location-scale or bivariate phylogenetic models that actually estimate a phylogenetic correlation.

Usage

```
drm_phylo_penalty_sweep(
  formula,
  data,
  family = gaussian(),
  cor_sd = c(0.25, 0.5, 1),
  sd_u = 1,
  sd_alpha = 0.05,
  control = drm_control(),
  ...
)
```

Arguments

`formula`, `data`, `family`, `control`	Passed to `drmTMB()`.
`cor_sd`	Numeric vector of positive correlation-penalty SDs to sweep.
`sd_u`, `sd_alpha`	Penalty SD-prior parameters; see `drm_phylo_penalty()`.
`...`	Further arguments passed to `drmTMB()`.

Value

A list with `$summary` – a data frame with one row per `cor_sd` giving `convergence`, `pdHess`, `logLik`, and any fit `error` – and `$fits` – the fitted objects, named by `cor_sd`, for extracting `corpars()`, `coef()`, and other couplings.

Examples

```
if (requireNamespace("ape", quietly = TRUE)) {
  set.seed(20260601)
  n_tip <- 10
  tree <- ape::rcoal(n_tip)
  tree$tip.label <- paste0("sp_", seq_len(n_tip))
```

```
  A <- ape::vcv(tree, corr = TRUE)
  u <- as.vector(t(chol(A)) %*% rnorm(n_tip)) * 0.6
  species <- factor(rep(tree$tip.label, each = 2), levels = tree$tip.label)
  x <- rnorm(length(species))
  dat <- data.frame(
    y = 0.3 + 0.5 * x + u[rep(seq_len(n_tip), each = 2)] +
      rnorm(length(species), sd = 0.5),
    x = x,
    species = species
  )

  out <- drm_phylo_penalty_sweep(
    bf(
      y ~ x + phylo(1 | species, tree = tree),
      sigma ~ phylo(1 | species, tree = tree)
    ),
    data = dat,
    family = gaussian(),
    cor_sd = c(0.5, 1)
  )
  out$summary
}
```

---

exceedance *Exceedance probability from a fitted model*

Description

`exceedance()` returns `Pr(Y > threshold | x)` (or, with `lower.tail = TRUE`, `Pr(Y <= threshold | x)`) at a fitted model's per-row conditional distribution. It is a thin wrapper over the shared CDF exposed by `fitted_distribution()`: `1 - fitted_distribution(object, newdata)$p(threshold)` (or the `p(threshold)` complement for `lower.tail = TRUE`).

Usage

```
exceedance(object, threshold, newdata = NULL, lower.tail = FALSE, ...)

## S3 method for class 'drmTMB'
exceedance(
  object,
  threshold,
  newdata = NULL,
  lower.tail = FALSE,
  response = NULL,
  ...
)
```

Arguments

object	A drmTMB fit.
threshold	Numeric threshold c. Either a single value (recycled across rows) or one value per row of newdata (or per fitted row when newdata is omitted).
newdata	Optional data frame for prediction. If omitted, fitted rows are used; see fitted_distribution().
lower.tail	Logical. If FALSE (default), returns Pr(Y > threshold). If TRUE, returns Pr(Y <= threshold).
...	Reserved for future options.
response	For a bivariate biv_gaussian fit, 1 or 2, selecting which response's marginal exceedance to compute; see fitted_distribution(). Must be NULL (the default) for univariate model types.

Details

threshold is evaluated with the standard CDF convention F(c) = Pr(Y <= c), so for atom-bearing families (e.g. Tweedie's point mass at y = 0) a threshold exactly at the atom includes that atom's mass in F(c): exceedance(fit, 0) (the default lower.tail = FALSE) excludes the atom at 0, matching Pr(Y > 0); exceedance(fit, 0, lower.tail = TRUE) recovers the atom mass Pr(Y <= 0) = Pr(Y == 0) when the family's support has no continuous mass below the atom.

This is a distributional (plug-in) output at predict_parameters()'s fixed-effect, population-level parameter estimates theta_hat: the result carries attr(., "calibrated") <- FALSE and does not propagate theta_hat uncertainty. See fitted_distribution() for the "spike"/"unimplemented" status gate this inherits (a "spike"-status family emits a one-time warning) and for the response argument's contract on a bivariate biv_gaussian fit (REQUIRED there: 1 or 2, selecting which response's MARGINAL exceedance to return; rho12 and any joint tail structure are ignored).

Value

A numeric vector, one value per row, with attr(., "calibrated") == FALSE.

See Also

fitted_distribution(), predict.drmTMB()

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, -0.5, 0, 0.5))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
exceedance(fit, threshold = 1)
```

`fitted.drmTMB` *Extract fitted response values*

Description

`fitted()` returns fitted response values from a `drmTMB` model. For univariate Gaussian, Student-t, Gamma, Tweedie, beta, beta-binomial, ordinary Poisson, ordinary negative-binomial, and cumulative-logit ordinal fits this is the fitted response summary. For Tweedie fits, that summary is the unconditional response mean `mu`. For beta-binomial fits, that summary is the fitted success probability `mu`. For ordinal fits, that summary is the expected ordered category score, `sum_k k * Pr(y_i = k)`. For zero-truncated negative-binomial 2 fits this is the positive-count mean `mu / (1 - Pr_NB2(0))`, where `mu` is the untruncated NB2 component mean. For hurdle negative-binomial 2 fits this is the unconditional response mean `(1 - hu) * mu / (1 - Pr_NB2(0))`. For zero-inflated Poisson and zero-inflated negative-binomial 2 fits this is the unconditional response mean `(1 - zi) * mu`, where `mu` is the conditional count mean. For bivariate Gaussian fits this is a two-column matrix with `mu1` and `mu2`. The same two-column marginal-mean contract applies to bivariate Student-t fits because their shared `nu` is greater than 2. For bivariate lognormal fits it is a two-column matrix of arithmetic marginal means. For lognormal fits this is the arithmetic response mean, `exp(mu + sigma^2 / 2)`.

Usage

```
## S3 method for class 'drmTMB'
fitted(object, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future fitted-value options.

Details

Fitted values are returned for the original fitted rows. Use `predict()` for new data or for non-location distributional parameters such as `sigma` or `rho12`.

Value

A numeric vector for univariate fits, or a two-column matrix for bivariate Gaussian, bivariate lognormal, or bivariate Student-t fits.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
fitted(fit)
```

`fitted_distribution` *Fitted distribution accessor*

Description

`fitted_distribution()` returns an object carrying, for each row of the fitted data (or `newdata`), the fitted distributional-parameter estimates from `predict_parameters()` together with density (`d`), CDF (`p`), and quantile (`q`) functions evaluated at those fitted parameters. Downstream consumers (quantile residuals, `predict(type = "quantile")`, `exceedance()`) are meant to route through this accessor rather than re-deriving the public-to-native parameter conversion.

Usage

```
fitted_distribution(object, ...)

## S3 method for class 'drmTMB'
fitted_distribution(object, newdata = NULL, response = NULL, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future options.
`newdata`	Optional data frame for prediction. If omitted, fitted rows are used.
`response`	For a bivariate `biv_gaussian` fit, `1` or `2`, selecting which response's marginal distribution to return. Required for `biv_gaussian`; must be `NULL` (the default) for univariate model types.

Details

`fitted_distribution()` only supports `model_types` with a promoted entry in `drm_family_dpq()`: as of DO-T3 batch D all 18 fitted `model_type` values are promoted (`status = "reference"`), including bivariate `"biv_gaussian"` (see the `response` argument below). `newdata` support inherits the same limitation as `predict_parameters()`: fixed-effect, population-level predictions only. For meta-analysis gaussian fits (`meta_V()`), the known sampling variance is taken from the fit for fitted rows (`newdata = NULL`); when `newdata` is supplied it must carry a `V` column giving the per-row known sampling variance, or an error is raised (rather than silently assuming 0). Ordinary (non-meta) fits need no `V` column. For binomial and beta_binomial fits, fitted rows reuse the fitted `trials` denominator; `newdata` must carry a `trials` column giving the per-row denominator, mirroring the meta_V() `V`-column contract. For cumulative_logit fits, the fitted ordinal cutpoints are attached as `CP1`..`CPk` columns (constant across rows, including `newdata` rows – the cutpoints do not depend on covariates).

`response` selects which response a bivariate `biv_gaussian` fit's returned distribution describes: `1` for `(mu1, sigma1)`, `2` for `(mu2, sigma2)`. It is **required** for `biv_gaussian` (an error names the two valid values if omitted) and is **not used** for univariate `model_types` (passing a non-`NULL` value errors, rather than being silently ignored). The returned distribution is the MARGINAL of that one response – exactly `N(mu_k, sigma_k)`, independent of `rho12` – never the joint bivariate

distribution; there is no `response = "joint"` option. `newdata` for a `biv_gaussian` fit inherits any known bivariate sampling covariance as `V_known = 0` (marginal-only scope, matching DO-T2's original `predict(type = "quantile")` documentation); fitted rows correctly use response `k`'s slice of a `meta_V()` fit's known sampling variance.

Value

An object of class `"drm_fitted_distribution"`: a list with `model_type`, `status`, `discrete`, `has_atom`, `atoms` (numeric vector of isolated atom locations, `numeric(0)` when none – see `drm_family_dpq()`'s header comment), `params` (wide data frame of per-row native dpar estimates), and `d`, `p`, `q` (one-argument functions bound to `params`).

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, -0.5, 0, 0.5))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
fd <- fitted_distribution(fit)
fd$p(dat$y)
```

---

fixef *Extract fixed-effect coefficients*

Description

`fixef()` returns the fixed-effect coefficients for one distributional parameter, or all fixed-effect coefficient blocks when `dpar = NULL`. It is a mixed-model-friendly alias for `coef()`.

Usage

```
fixef(object, ...)

## S3 method for class 'drmTMB'
fixef(object, dpar = NULL, ...)

## S3 method for class 'drmTMB'
coef(object, dpar = NULL, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.
`dpar`	Optional distributional parameter name, such as `"mu"`, `"sigma"`, `"hu"`, `"rho12"`, `"sd(id)"`, or `"sd_phylo(species)"`.

Value

A named numeric vector when `dpar` is supplied, otherwise a named list of coefficient vectors.

Examples

```
set.seed(20260525)
dat <- data.frame(
  y = 0.2 + 0.6 * seq(-1, 1, length.out = 24) + rnorm(24, sd = 0.5),
  x = seq(-1, 1, length.out = 24)
)
fit <- drmTMB(bf(y ~ x, sigma ~ x), data = dat)

# One distributional parameter at a time: a named numeric vector.
fixef(fit, "mu")
coef(fit, "sigma")

# Omit `dpar` for every block at once: a named list, one element per
# distributional parameter. `coef()` and `fixef()` return the same structure.
coef(fit)
names(coef(fit))
identical(coef(fit), fixef(fit))
```

---

imputed *Extract fitted missing-predictor summaries*

Description

`imputed()` reports the fitted values used for explicitly modelled missing predictors, for two readers: applied users inspecting a fit, and downstream packages (for example `drmSEM`) that build effect intervals directly on these values and need to know, row by row, whether `std_error` is usable.

Usage

```
imputed(object, ...)

## S3 method for class 'drmTMB'
imputed(object, variable = NULL, rows = c("missing", "all"), se = TRUE, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.
`variable`	Optional missing-predictor name. The default uses the only modelled missing predictor in the fit.
`rows`	Which rows to return. `"missing"` returns only fitted missing predictor values. `"all"` returns retained model rows, with observed predictor values labelled as observed.
`se`	Logical; include conditional standard errors when the fit contains a successful `TMB::sdreport()` result. `se = FALSE` reports `std_error = NA` and `uncertainty_status = "ok"` throughout, because nothing was requested.

Details

Gaussian missing predictor values are reported as conditional modes from the fitted TMB likelihood, a genuine random effect; their `std_error` comes from `TMB::sdreport()`'s conditional covariance. Every other fitted route already stores a normalized posterior probability vector or matrix over a finite-state or quadrature grid, and the reported value is already the posterior mean of that grid: binary missing predictor values are fitted conditional probabilities (Bernoulli/logit predictor model), ordered categorical values are fitted conditional expected scores (cumulative-logit predictor model), beta/proportion, zero-one beta boundary-proportion, and denominator-aware beta-binomial values are fitted conditional means (quadrature or exact summation over the reported proportion), count values are fitted conditional expected counts (Poisson, negative-binomial, or zero-truncated negative-binomial predictor models), and lognormal, Gamma, and Tweedie values are fitted conditional quadrature means. For all of these, `std_error` is the posterior standard deviation over that same grid, `sqrt(sum(p * (x - mean)^2))`, computed directly from the stored grid with no additional TMB computation. The one exception is unordered categorical missing predictor values, reported as fitted conditional modal category scores from the baseline-category softmax predictor model: the reported value is a mode over unordered nominal codes, and no metrically meaningful variance exists over categories that carry no order, so `std_error` stays `NA` for this route.

`uncertainty_status` tells a consumer why `std_error` is `NA` when it is, and what to do about it:

- `"ok"`: `std_error` is a usable standard error (or the row is an observed value, which never carries one). Use it.
- `"sdreport_skipped"`: the fit used `drm_control(se = FALSE)`. Refit with `se = TRUE` (the default) to get standard errors.
- `"sdreport_failed"` or `"sdreport_non_pd_hessian"`: `TMB::sdreport()` ran but did not return a usable covariance. Standard errors are unavailable for this fit; diagnose with `check_drm()` before trusting point estimates either.
- `"sdreport_unavailable"`: the fit has no `TMB::sdreport()` object at all (for example, refitted without `keep_tmb_object`). Refit to get one.
- `"route_conditional_se_unavailable"`: the fit and `sdreport()` are both fine, but this specific missing-predictor row has no well-defined posterior standard error — currently only the unordered categorical route. Do not treat the missing `std_error` as zero or impute one; the point estimate (the modal category) remains usable on its own.

This is not multiple imputation: the output does not contain posterior means, posterior intervals, credible intervals, or pooled-imputation summaries.

Value

A data frame with `variable`, `original_row`, `model_row`, `observed`, `estimate`, `std_error`, `source`, and `uncertainty_status`. `uncertainty_status` takes the values `"ok"`, `"sdreport_skipped"`, `"sdreport_failed"`, `"sdreport_non_pd_hessian"`, `"sdreport_unavailable"`, or `"route_conditional_se_unavailab`
see Details.

Examples

```
set.seed(20260532)
n <- 48
dat <- data.frame(
```

```
  moisture = seq(-1.5, 1.5, length.out = n),
  canopy = cos(seq_len(n) / 5)
)
dat$body_mass_full <- 0.2 + 0.7 * dat$moisture - 0.2 * dat$canopy +
  rnorm(n, sd = 0.08)
dat$growth <- 0.6 + 1.1 * dat$body_mass_full - 0.3 * dat$moisture +
  rnorm(n, sd = 0.20)
dat$body_mass <- dat$body_mass_full
dat$body_mass[c(7, 19, 34, 43)] <- NA_real_

fit <- drmTMB(
  bf(growth ~ moisture + mi(body_mass), sigma ~ 1),
  family = gaussian(),
  data = dat,
  impute = list(body_mass = body_mass ~ moisture + canopy),
  missing = miss_control(predictor = "model")
)
imputed(fit)
```

`impute_model` *Define a missing-predictor model*

Description

`impute_model()` wraps the model for a predictor used inside `mi()`. A bare formula in `impute`, such as `impute = list(x = x ~ z)`, is still treated as a Gaussian model for a numeric missing predictor. Use `impute_model()` when the missing predictor needs an explicit non-Gaussian predictor family. The first non-Gaussian fitted routes are fixed-effect Bernoulli/logit models for one binary predictor, fixed-effect cumulative-logit models for one ordered categorical predictor, fixed-effect baseline-category softmax models for one unordered categorical predictor, fixed-effect beta models for one strict proportion predictor in `(0, 1)`, fixed-effect zero-one beta models for one boundary proportion predictor in `[0, 1]`, fixed-effect beta-binomial models for one denominator-aware success/trial proportion predictor, fixed-effect Poisson, negative-binomial, or zero-truncated negative-binomial models for one count predictor, fixed-effect lognormal or Gamma models for one positive continuous predictor, and fixed-effect Tweedie models for one non-negative semi-continuous predictor with exact zeros. Most current non-Gaussian predictor families are fitted inside a Gaussian response location model; Poisson, binomial, negative-binomial, and beta responses are currently supported for one binary missing predictor.

Usage

```
impute_model(formula, family = stats::gaussian(), trials = NULL)
```

Arguments

`formula`	Two-sided predictor-model formula. For most families, the left-hand side must be the same variable used inside `mi()`. For `family = beta_binomial()`, the left-hand side is the success-count column, while the `mi()` variable is the success proportion used in the response model.

`family` Predictor-model family. `gaussian()` keeps the existing continuous predictor route. `binomial(link = "logit")` fits the binary missing-predictor route. `cumulative_logit()` fits the ordered categorical missing-predictor route. `categorical()` fits the unordered categorical missing-predictor route. `beta()` fits the strict beta/proportion missing-predictor route. `zero_one_beta()` fits the boundary-proportion missing-predictor route. `beta_binomial()` fits a denominator-aware success/trial proportion route and requires `trials`. `poisson(link = "log")`, `nbinom2()`, and `truncated_nbinom2()` fit count missing-predictor routes. `lognormal()` and `Gamma(link = "log")` fit positive continuous missing-predictor routes. `tweedie()` fits a semi-continuous non-negative missing-predictor route with exact zeros.

`trials` Optional trial-count column for `family = beta_binomial()`. The formula left-hand side is the success count and `trials` is the known denominator for each row.

Value

A `drm_impute_model` object for the `impute` argument of `drmTMB()`.

Examples

```
impute_model(x ~ z)
impute_model(treatment ~ z, family = binomial())
impute_model(score ~ z, family = cumulative_logit())
impute_model(habitat ~ z, family = categorical())
impute_model(cover ~ z, family = beta())
impute_model(cover ~ z, family = zero_one_beta())
impute_model(success ~ z, family = beta_binomial(), trials = trials)
impute_model(abundance ~ z, family = poisson())
impute_model(abundance ~ z, family = nbinom2())
impute_model(abundance ~ z, family = truncated_nbinom2())
impute_model(biomass ~ z, family = lognormal())
impute_model(biomass ~ z, family = Gamma(link = "log"))
impute_model(biomass ~ z, family = tweedie())
```

---

is_converged *Check whether a fit converged*

Description

`is_converged()` is a compact programmatic flag for workflows that need a yes/no answer before comparing, displaying, or post-processing a `drmTMB` fit. By default it checks only stored optimizer status: the `nlminb()` convergence code must be 0 and the stored objective and log-likelihood must be finite.

Usage

```
is_converged(object, ...)

## Default S3 method:
is_converged(object, ...)

## S3 method for class 'drmTMB'
is_converged(object, include_hessian = FALSE, ...)
```

Arguments

object	A drmTMB fit.
...	Reserved for future convergence options.
include_hessian	Logical; require a positive-definite TMB::sdreport() Hessian in addition to optimizer convergence.

Details

Set include_hessian = TRUE when the next step needs Wald-style uncertainty. In that mode, is_converged() also requires successful TMB::sdreport() output with pdHess = TRUE. A fit can therefore be optimizer-converged while still returning FALSE with include_hessian = TRUE; this marks an inference-readiness problem, not automatic proof that point estimates are unusable.

Use check_drm() when you need the full diagnostic table and messages.

Value

A single logical value.

Examples

```
set.seed(20260531)
dat <- data.frame(y = rnorm(30), x = rnorm(30))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
is_converged(fit)
is_converged(fit, include_hessian = TRUE)
```

---

latent_normal *Construct a latent-normal association kernel*

Description

latent_normal() declares the Gaussian-copula kernel used by associate_pairs(). It is not a Gaussian residual-correlation model and does not use rho12().

Usage

```
latent_normal()
```

Value

A latent-normal kernel specification.

`lognormal` *Lognormal response family*

Description

`lognormal()` defines a one-response positive continuous distribution with formulas for log-location `mu` and log-scale `sigma`.

Usage

```
lognormal()
```

Details

The model is defined on the log response scale: `log(y) ~ Normal(mu, sigma^2)`. The fitted distributional parameter `mu` is therefore the mean of `log(y)`, not the arithmetic mean of `y`. Ordinary `mu` random intercepts such as `(1 | id)` are supported in the first positive-continuous mixed-model slice; the independent numeric slope `(0 + x | id)` also fits but is a recovery-grade route only (trust the point estimate, not the interval). One complete-data ordinary unlabelled correlated intercept-slope block `(1 + x | id)` is also fitted at `point_fit_recovery` (`mc-0720`) under the design-17 map for group-level `sd0`, `sd1`, and `rho_re` (never residual `rho12`).

Value

A `drm_family` object.

Examples

```
lognormal()
```

`make_mesh` *Construct a fixed-kappa SPDE mesh*

Description

Builds a two-dimensional `fmesher` triangulation, its finite-element matrices, and the observation-to-vertex projection matrix used by the fixed-kappa mesh spatial field. This helper only creates the geometry and fixed SPDE inputs; it neither estimates kappa nor defines a spatial likelihood.

Usage

```
make_mesh(coords, kappa, mesh = NULL, crs = attr(coords, "crs"), ...)
```

Arguments

coords	A finite two-column matrix or data frame of metric planar coordinates, normally returned by spatial_coords(). Supply crs when the coordinate object does not already carry it.
kappa	One finite strictly positive fixed SPDE kappa value, expressed in inverse units of coords.
mesh	Optional pre-built fmesher two-dimensional mesh. When supplied, it is reprojected onto coords and validated rather than rebuilt.
crs	A projected CRS for plain planar coordinate inputs. Geographic CRS values are rejected.
...	Additional arguments passed to fmesher::fm_mesh_2d_inla() when a mesh is built.

Value

A drmTMBmesh object containing loc_xy, xy_cols, mesh, spde, loc_centers, A_st, alignment identifiers, the projected CRS, and the fixed kappa configuration.

Examples

```
if (requireNamespace("sf", quietly = TRUE) && requireNamespace("fmesher", quietly = TRUE)) {
  locations <- data.frame(lon = c(-123.2, -123.1, -123.15),
                          lat = c(49.2, 49.3, 49.25))
  xy <- spatial_coords(locations, lon, lat, crs_out = 32610)
  make_mesh(xy, kappa = 1 / 10000)
}
```

---

marginal_parameters *Marginal summaries of predicted distributional parameters*

Description

marginal_parameters() averages predicted distributional parameters over fitted rows or a supplied newdata grid. It is a simple plug-in summary layer built on predict_parameters(), intended for interpretation tables and plotting helpers that need averages rather than row-level predictions.

Usage

```
marginal_parameters(object, ...)

## S3 method for class 'drmTMB'
marginal_parameters(
  object,
  newdata = NULL,
  dpar = NULL,
  by = NULL,
```

```
  type = c("response", "link"),
  ...
)
```

Arguments

object	A drmTMB fit.
...	Reserved for future options.
newdata	Optional data frame for prediction. If omitted, fitted rows are used.
dpar	Optional character vector of distributional parameters to summarise, including fitted random-effect scale model names such as "sd(id)". NULL summarises all fitted distributional parameters.
by	Optional character vector of columns in newdata used to define marginal groups. NULL averages over all prediction rows for each distributional parameter.
type	Prediction scale: "response" or "link".

Details

This helper does not compute uncertainty, contrasts, or profile intervals. It reports unweighted averages of already-predicted parameter values. For population-level summaries, supply an explicit newdata grid; with newdata = NULL, the fitted-row prediction contract is the same as predict.drmTMB().

Averaging uses a moment-appropriate scale so that dispersion and correlation summaries stay interpretable. On type = "response", standard-deviation parameters (sigma, sigma1, sigma2, and random-effect sd(...) models) are averaged on the variance scale (the reported value is the root of the mean squared per-row SD), and correlation parameters (rho12) are averaged on the Fisher-z scale. Location and other parameters use the arithmetic mean. On type = "link", all parameters are already on an unconstrained scale, so the arithmetic mean of the linear predictor is reported unchanged. These remain unweighted plug-in summaries, not exact marginal moments of the mixture over rows.

The returned table carries the same interval provenance columns as predict_parameters(). In this first contract, marginal summaries are point estimates with conf.status = "not_requested" and interval_source = "not_available".

Value

A data frame with one row per distributional parameter and grouping combination. The returned columns are dpar, component, type, optional by columns, estimate, n, conf.status, and interval_source.

Examples

```
set.seed(20260523)
n <- 48
x <- seq(-1.5, 1.5, length.out = n)
habitat <- factor(rep(c("reef", "sand"), length.out = n))
eta <- 0.4 + 0.7 * x + ifelse(habitat == "reef", 0.25, -0.15)
sigma <- exp(-0.35 + 0.15 * x)
dat <- data.frame(y = eta + rnorm(n, sd = sigma), x = x, habitat = habitat)
fit <- drmTMB(bf(y ~ x + habitat, sigma ~ x), data = dat)
```

```
grid <- prediction_grid(
  fit,
  focal = "habitat",
  at = list(habitat = levels(dat$habitat)),
  margin = "empirical"
)
marginal_parameters(fit, newdata = grid, dpar = c("mu", "sigma"), by = "habitat")
```

`meta_V` *Known sampling covariance marker*

Description

`meta_V()` marks known sampling variance or covariance in a formula. It is designed for meta-analysis and other regression problems where part of the observation covariance is known in advance.

Usage

```
meta_V(V)
```

Arguments

`V`	A column name, vector, diagonal matrix, block-diagonal matrix, or full covariance matrix. Diagonal/vector `V` represents independent known sampling variances. A matrix represents the known covariance among rows.

Value

A formula marker; never evaluated by users.

Examples

```
bf(yi ~ moderator + meta_V(V = vi), sigma ~ moderator)
```

`meta_vcov_bivariate` *Build paired bivariate sampling covariance*

Description

`meta_vcov_bivariate()` builds a dense row-paired sampling covariance matrix for bivariate meta-analysis. It is a convenience helper for constructing the known `V` matrix used by `meta_V()` in complete-row bivariate Gaussian meta-analysis.

Usage

```
meta_vcov_bivariate(v1, v2, cov12 = NULL, cor12 = NULL)
```

Arguments

v1, v2	Numeric vectors of known sampling variances for response 1 and response 2.
cov12	Optional known sampling covariance between the two response estimates within each study. May be length one or the same length as v1.
cor12	Optional known sampling correlation between the two response estimates within each study. May be length one or the same length as v1. Supply at most one of cov12 and cor12.

Details

The returned matrix uses row-paired stacking: y1[1], y2[1], y1[2], y2[2], ..., y1[n], y2[n]. Each study contributes one 2 by 2 block with diagonal entries v1[i] and v2[i] and off-diagonal entries cov12[i]. If cor12 is supplied, the covariance is computed as cor12 * sqrt(v1 * v2).

In a bivariate Gaussian fit, this known sampling covariance is added to the fitted residual covariance from sigma1, sigma2, and rho12. The fitted rho12 therefore remains the residual covariance component after accounting for known within-study sampling covariance. A separate study-level random effect would be needed to label a correlation as a study-level correlation.

Value

A dense 2 * length(v1) by 2 * length(v1) covariance matrix with class "drm_meta_vcov_bivariate".

Examples

```
V <- meta_vcov_bivariate(
  v1 = c(0.04, 0.03),
  v2 = c(0.05, 0.02),
  cor12 = c(0.4, 0.2)
)
dim(V)
```

---

mi *Missing-predictor model marker*

Description

mi() marks a predictor whose missing values should be handled by the missing-data predictor model. Most fitted routes support one mi(x) term in a univariate Gaussian location formula. Numeric missing predictors can use a matching Gaussian fixed-effect, one random-intercept, or one intercept-only structured predictor model supplied through impute, for example impute = list(x = x ~ z) or impute = list(x = x ~ z + relmat(1 | line, Q = Q)). Family-aware fixed-effect predictor models supplied with impute_model() cover binary, ordered categorical, unordered categorical, strict proportion, zero-one boundary proportion, denominator-aware beta-binomial proportion, count, positive continuous, and semi-continuous predictors. The non-Gaussian response routes support one binary mi() predictor modelled by family = binomial() for family = poisson(), binomial(), nbinom2(), and beta().

Usage

```
mi(x)
```

Arguments

x	A predictor in a supported missing-predictor route.

Value

x, so standard R model-frame construction can evaluate the marker.

Examples

```
bf(y ~ z + mi(x), sigma ~ 1)
```

---

miss_control *Configure missing-data handling*

Description

`miss_control()` configures the first drmTMB missing-data slices. The default keeps the existing complete-case behaviour. In the current fitted slices, `response = "include"` is implemented for every current univariate fitted response route and for bivariate Gaussian partial-response rows without dense known covariance. It uses an observed-response mask: a missing response contributes no direct response likelihood. `predictor = "model"` is implemented mainly for one `mi()` missing predictor at a time in a univariate Gaussian location model: numeric missing predictors can use Gaussian fixed-effect, grouped, or structured predictor models. Binary, ordered categorical, unordered categorical, strict beta/proportion, zero-one beta boundary proportion, denominator-aware beta-binomial success/trial proportion, Poisson, negative-binomial, or zero-truncated negative-binomial count, positive continuous lognormal or Gamma, and exact-zero semi-continuous Tweedie missing predictors can use fixed-effect predictor models supplied by `impute_model()`. The non-Gaussian response slices support `poisson()`, `binomial()`, `nbinom2()`, and `beta()` responses, each with one fixed-effect Bernoulli/logit binary missing predictor. EM/profile engines and simulation-based imputation summaries are reserved for later slices.

Usage

```
miss_control(
  response = c("drop", "include"),
  predictor = c("fail", "model"),
  engine = "laplace"
)
```

Arguments

response	Response missingness policy. "drop" keeps existing complete-case fitting; for a bivariate Gaussian model, a row with either response missing is omitted. "include" keeps rows with supported missing responses and masks their response likelihood contribution.
predictor	Predictor missingness policy. "fail" errors on missing predictors. "model" enables the current mi() predictor-model routes when paired with a matching impute formula or impute_model() in drmTMB().
engine	Missing-data engine. Only "laplace" is implemented in this slice.

Value

A drm_missing_control object.

Examples

```
miss_control()
miss_control(response = "include")
miss_control(predictor = "model")
```

---

native_reader_contracts

Stable native reader contracts

Description

These contracts define the minimum public output used by drmTMB's native reader journeys. They do not stabilize every component retained on fitted or summary objects for backward compatibility.

Details

check_drm() returns a drm_check data frame with the character columns check, status, value, and message, in that order. Status values are "ok", "note", "warning", or "error". Its "ok" attribute is TRUE exactly when no row has status "warning" or "error".

summary() guarantees the reader tables coefficients, parameters, covariance, and derived, plus the confint key. confint is NULL when intervals were not requested and a table otherwise. Relayed sdpars, corpars, ordinal, uncertainty, and mspl components remain available for compatibility but are outside this stable reader contract.

For ranef(), terms is the stable model-scale conditional-deviation interface on ordinary distributional and named structured-effect blocks. It is not a random-effect standard-deviation extractor. values, latent, covariance_blocks, and specialised mesh fields are advanced compatibility components; covariance-block entries do not promise terms. A fit with no random effects returns an empty list when dpar = NULL; requesting a named block produces an error explaining that the fit contains no random effects.

fitted() is the response-summary interface, while predict_parameters() is the distributional-component surface. For example, for a lognormal fit, fitted() returns the arithmetic response

mean `exp(mu + sigma^2 / 2)`, not the log-scale `mu` returned by `predict_parameters(..., dpar = "mu")`.

In interval tables, `conf.status` reports the result or availability state, `interval_source` records provenance, and `profile.boundary` is only a flag that the relevant profile reached a boundary.

See Also

`check_drm()`, `summary.drmTMB()`, `ranef.drmTMB()`, `fitted.drmTMB()`, and `predict_parameters()`

`nbinom2` *Negative binomial 2 response family*

Description

`nbinom2()` defines a one-response count distribution with formulas for the mean `mu` and overdispersion scale `sigma`.

Usage

```
nbinom2()
```

Details

The implemented contract is `log(mu) = eta_mu`, `log(sigma) = eta_sigma`, and `Var(y) = mu + sigma^2 * mu^2`. Thus larger `sigma` means greater extra-Poisson variation. Internally this is equivalent to the usual NB2 size parameter `size = 1 / sigma^2`. Ordinary non-zero-inflated NB2 models also support one complete-data unlabelled `(1 + x | id)` ML-Laplace correlated intercept-slope block on the log-mean (`mc-0719`), and first-slice random intercepts on the log-`sigma` predictor, such as `bf(count ~ x, sigma ~ z + (1 | id))`. Structured `sigma` effects (`phylo`/`spatial`/`animal`/`relmat`) also fit as a point-recovery route (trust the point estimate, not the interval; not yet coverage-verified). NB2 `sigma` random slopes remain planned. Zero-inflated NB2 supports only the point-fit-only IID control `bf(count ~ fixed_effects, sigma ~ 1 + (1 | group), zi ~ 1)` under ML with complete responses; zero-inflated sigma predictors, slopes, labels, and other random-effect combinations remain unsupported.

Value

A `drm_family` object.

Examples

```
nbinom2()
```

phylo *Phylogenetic structured-effect marker*

Description

phylo() marks user-facing syntax for phylogenetic dependence. The current fitted paths support Gaussian location and residual-scale effects, response-specific direct-SD formulas for location effects, labelled bivariate Gaussian location-scale blocks, and the first ordinary Poisson q=1 and NB2 q=1 location effects. Use phylo(1 | species, tree = tree) in univariate Gaussian mu, univariate Gaussian sigma, ordinary Poisson mu, or ordinary NB2 mu, phylo(1 + x | species, tree = tree) for the unlabelled ordinary Poisson/NB2 count one-slope gate, one numeric univariate Gaussian mu slope with independent intercept/slope SDs, matching univariate Gaussian mu and sigma intercept terms for a mean-scale phylogenetic correlation, matching terms in bivariate Gaussian mu1 and mu2, matching labelled all-four intercept terms across Gaussian mu1, mu2, sigma1, and sigma2, or the first shared-label all-four one-slope point-fit/extractor cell. A single shared intercept label estimates the full q4 block; a mu1/mu2 intercept label plus a separate sigma1/sigma2 intercept label estimates the block-diagonal fallback. The all-four phylo(1 + x | p | species, tree = tree) cell is native point-fit/extractor evidence only; bridge parity, intervals, coverage, REML, AI-REML, block-diagonal all-four slope layouts, Gaussian multiple phylogenetic slopes, pure, labelled, or multiple non-Gaussian phylogenetic slopes, zero-inflated phylogenetic effects, and phylogenetic slope correlations remain planned. The public phylo() API requires an ultrametric tree with branch lengths and uses the Hadfield and Nakagawa A-inverse sparse-precision path internally.

Usage

```
phylo(term, tree)
```

Arguments

term	Structured random-effect term, currently 1 | species or 1 + x | species.
tree	Ultrametric phylogeny input with branch lengths.

Value

A formula marker; never evaluated by users.

Examples

```
bf(y ~ x + phylo(1 | species, tree = tree), sigma ~ z)
bf(count ~ x + phylo(1 | species, tree = tree))
bf(count ~ x + phylo(1 + x | species, tree = tree))
```

`phylo_interaction` *Bipartite phylogenetic interaction marker*

Description

`phylo_interaction()` marks a pair-level phylogenetic interaction between two partner clades. It is the friendly route for bipartite ecological-network effects where rows are plant-pollinator, host-symbiont, predator-prey, or other two-partner interactions. The first fitted slice is a single q=1 location random intercept whose latent precision is the sparse Kronecker product of the two phylogenetic augmented precisions. Use `relmat(1 | pair, Q = Q_pair)` as the lower-level user-supplied precision escape hatch.

Usage

```
phylo_interaction(term, tree1, tree2)
```

Arguments

`term`	Structured pair random-effect term, currently `1 | partner1:partner2`.
`tree1`	Ultrametric phylogeny for the first partner clade.
`tree2`	Ultrametric phylogeny for the second partner clade.

Details

The fitted first slice supports ordinary Gaussian, ordinary Poisson, and ordinary NB2 location models wherever the existing q=1 structured `mu` machinery is active. Binary/Bernoulli incidence models and additive models that combine partner main phylogenies and the pair-level interaction are planned after the corresponding likelihood and multi-layer structured recovery tests are in place.

Value

A formula marker; never evaluated by users.

Examples

```
bf(
  count ~ x + phylo_interaction(
    1 | plant:pollinator,
    tree1 = plant_tree,
    tree2 = pollinator_tree
  )
)
```

plot.profile.drmTMB *Plot profile-likelihood curves*

Description

plot() for "profile.drmTMB" objects draws the likelihood-ratio curve returned by profile.drmTMB(). The dotted horizontal line is the likelihood-ratio cutoff for the stored confidence level, the solid vertical line marks the fitted estimate, and dashed vertical lines mark profile interval endpoints when they were extracted successfully. When the profile object contains coarse and dense passes, colour and line type separate the passes and the caption reports elapsed time for each.

Usage

```
## S3 method for class 'profile.drmTMB'
plot(x, interval = TRUE, ...)
```

Arguments

x	A "profile.drmTMB" object returned by profile.drmTMB().
interval	Logical; draw profile interval endpoint lines when finite conf.low and conf.high columns are available.
...	Reserved for future options.

Value

A ggplot object.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
prof <- profile(fit, parm = "sigma", profile_precision = "fast")
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(prof)
}
```

plot_corpairs *Plot fitted correlation-pair summaries*

Description

plot_corpairs() is a small ggplot2 consumer for tables returned by corpairs(). It does not compute correlation pairs, fit intervals, or choose a correlation layer. Build the table first with corpairs(), then pass that table to this helper.

Usage

```
plot_corpairs(
  data,
  colour = "level",
  facet = NULL,
  label = NULL,
  interval = TRUE,
  interval_style = c("eye", "line"),
  ...
)
```

Arguments

data	A data frame returned by corpairs(), or a compatible table with columns level, class, parameter, estimate, and modelled. conf.status and interval_source are optional for point-only tables, but finite intervals are drawn only when those columns mark a supported interval.
colour	Optional character scalar naming a column to map to colour. Use NULL to suppress colour mapping.
facet	Optional character scalar naming a column to facet by. Use NULL to suppress faceting.
label	Optional character scalar naming a column to use for y-axis row labels. Use this for publication figures where the full level | class | parameter label is too long. If NULL, labels are built from level, class, and parameter.
interval	Logical; draw finite conf.low/conf.high intervals when those columns are present.
interval_style	Character scalar. "eye" draws the default Confidence Eye region plus hollow point estimate. "line" draws conventional interval segments. Ignored when interval = FALSE.
...	Reserved for future options.

Details

The helper draws one hollow point per correlation row. If the table contains finite conf.low and conf.high bounds plus interval provenance columns that describe a real interval, the default draws a pale Confidence Eye for those rows only, using a guarded Fisher-z/atanh correlation scale to shape the eye. The zero-correlation reference is dotted. Rows without supported bounds remain visible as point estimates and keep their display interval status attached to the plotted data. Set interval_style = "line" for a conventional CI-line variant.

Value

A ggplot object.

Examples

```
pairs <- data.frame(
  level = c("residual", "group", "phylo", "group"),
  class = c("residual", "mean-slope", "structured", "scale-scale"),
  parameter = c(
    "rho12",
    "cor((Intercept),x | p | id)",
    "cor(mu1,mu2 | species)",
    "cor(sigma1,sigma2 | site)"
  ),
  label = c(
    "Residual\nrho12",
    "Group\nmean-slope cor",
    "Phylogenetic\nmu1-mu2",
    "Group\nsigma block"
  ),
  estimate = c(0.25, 0.45, -0.30, 0.12),
  modelled = c(FALSE, FALSE, FALSE, FALSE),
  conf.low = c(0.05, 0.10, -0.55, -0.12),
  conf.high = c(0.43, 0.72, -0.08, 0.34),
  conf.status = rep("profile", 4),
  interval_source = rep("profile", 4)
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  pair_palette <- c(
    group = "#D55E00",
    phylo = "#009E73",
    residual = "#0072B2"
  )
  plot_corpairs(pairs, label = "label") +
    ggplot2::scale_colour_manual(values = pair_palette) +
    ggplot2::scale_fill_manual(values = pair_palette) +
    ggplot2::theme_minimal(base_size = 11) +
    ggplot2::theme(
      panel.grid.major.y = ggplot2::element_blank(),
      panel.grid.minor.y = ggplot2::element_blank()
    ) +
    ggplot2::guides(colour = "none")
}
```

---

plot_parameter_surface

Plot predicted distributional-parameter surfaces

Description

`plot_parameter_surface()` is a small ggplot2 consumer for long tables returned by `predict_parameters()`. It does not fit a model, build a grid, compute predictions, compute confidence intervals, or choose an estimand. Build an explicit grid with `prediction_grid()` or another data-frame workflow first, then pass the resulting prediction table to this helper.

Usage

```
plot_parameter_surface(
  data,
  x,
  colour = NULL,
  group = NULL,
  facet = "dpar",
  dpar = NULL,
  type = NULL,
  line = TRUE,
  point = TRUE,
  interval = TRUE,
  ...
)
```

Arguments

data	A data frame returned by predict_parameters(), or a compatible long table with columns dpar, type, estimate, conf.status, and interval_source. If conf.low and conf.high are present, both must be numeric.
x	Character scalar naming the column to draw on the x-axis.
colour	Optional character scalar naming a column to map to colour.
group	Optional character scalar naming a column to group lines. If NULL, lines are grouped by dpar, colour, and facet columns when present.
facet	Optional character scalar naming a column to facet by. Use NULL to suppress faceting. The default facets by dpar.
dpar	Optional character vector of distributional parameters to keep.
type	Optional character vector of prediction scales to keep, such as "response" or "link".
line	Logical; draw lines through the estimates.
point	Logical; draw points at the estimates.
interval	Logical; draw finite conf.low/conf.high intervals when those columns are present and conf.status plus interval_source indicate that an interval was actually computed.
...	Reserved for future options.

Details

The helper plots estimate against one supplied column. It expects the interval provenance columns created by predict_parameters(). When finite conf.low and conf.high columns are present and the provenance columns describe a real interval, it draws confidence bands for continuous x-values and interval bars for discrete x-values. Rows without finite supported bounds remain visible as point or line estimates only. When the filtered table contains a single distributional parameter, the y-axis label names that parameter and, when unique, the prediction scale.

Value

A ggplot object.

Examples

```
x <- seq(-1, 1.5, length.out = 8)
pred <- rbind(
  data.frame(
    dpar = "mu",
    type = "response",
    estimate = 1 + 0.5 * x,
    conf.low = 0.85 + 0.5 * x,
    conf.high = 1.15 + 0.5 * x,
    conf.status = "wald",
    interval_source = "wald",
    x = x
  ),
  data.frame(
    dpar = "sigma",
    type = "response",
    estimate = 0.55 + 0.08 * x,
    conf.low = 0.47 + 0.08 * x,
    conf.high = 0.63 + 0.08 * x,
    conf.status = "wald",
    interval_source = "wald",
    x = x
  )
)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot_parameter_surface(pred, x = "x", point = FALSE) +
    ggplot2::labs(
      title = "Predicted parameter surfaces",
      subtitle = "Ribbons are Wald intervals from the supplied table"
    ) +
    ggplot2::theme_minimal(base_size = 11)
}
```

---

predict.drmTMB *Predict distributional parameters*

Description

predict() returns fitted or predicted values for one distributional parameter of a drmTMB fit.

Usage

```
## S3 method for class 'drmTMB'
predict(
  object,
```

```
  newdata = NULL,
  dpar = NULL,
  type = c("response", "link", "quantile"),
  prob = c(0.025, 0.5, 0.975),
  ...
)
```

Arguments

object	A drmTMB fit.
newdata	Optional data frame for prediction. If omitted, fitted rows are used. When supplied, newdata must include the predictors used by the requested dpar; required predictor values must be complete, required numeric predictors must be finite, and factor predictors must use fitted levels. Transformed predictor terms, such as log(size), must also evaluate to finite design-matrix values.
dpar	Distributional parameter to predict. If NULL, the first fitted distributional parameter is used. For type = "quantile" on a bivariate biv_gaussian fit, dpar instead selects which response's marginal quantile to compute (see Details).
type	Prediction scale: "response", "link", or "quantile".
prob	Numeric vector of probabilities in (0, 1), used only when type = "quantile".
...	Reserved for future prediction options.

Details

By default, predictions are returned on the distributional parameter's response scale. For positive scale parameters such as sigma, this means the exponentiated value. For bivariate residual correlation rho12 or a fitted corpair() model, this means the correlation scale. Use type = "link" to return the linear predictor instead.

When newdata = NULL, predictions are for the fitted rows and include currently implemented conditional random-effect contributions for mu, including registry-backed q > 2 ordinary covariance blocks, bivariate mu1/mu2, phylogenetic mu, and residual-scale sigma including bivariate sigma1/sigma2 blocks. Fitted-row predictions also include ordinary zero-one-beta zoi and coi random-effect contributions. When newdata is supplied, predictions are fixed-effect, population-level predictions for the supplied rows.

type = "quantile" returns per-row conditional quantiles of the fitted RESPONSE distribution (not of a linear predictor): qnorm-style inverse CDF evaluation via fitted_distribution()$q() at predict_parameters()'s fixed-effect, population-level parameter estimates – see fitted_distribution() for the fixed-effect-only scope and the "spike"/"unimplemented" status gate this inherits (a "spike"-status family emits a one-time warning; an unregistered model_type raises a clear error). For bivariate biv_gaussian fits, dpar selects which response's MARGINAL quantile to return ("mu1"/"sigma1" for response 1, "mu2"/"sigma2" for response 2); the joint rho12 correlation, any joint tail structure, and any known bivariate sampling covariance are ignored – this is a marginal-only computation, not a joint bivariate quantile. The result carries attr(., "calibrated") <- FALSE: this is a distributional (plug-in) interval at the point estimate theta_hat, not a calibrated-coverage interval, and it does not propagate theta_hat uncertainty.

Value

A numeric vector for `type = "response"/"link"`. For `type = "quantile"`, a numeric matrix with one row per observation and one column per `prob` (columns named as percentages, e.g. `"2.5%"`), with `attr(., "calibrated") == FALSE`.

See Also

`fitted.drmTMB()`, `rho12()`, `stats::sigma()`, `fitted_distribution()`, `exceedance()`

Examples

```
dat <- data.frame(
  y = c(0.2, 0.5, 1.1, 1.4, 1.8, 2.2),
  x = c(-1, -0.5, 0, 0.5, 1, 1.5)
)
fit <- drmTMB(bf(y ~ x, sigma ~ x), data = dat)
predict(fit, dpar = "mu")
predict(fit, dpar = "sigma")
predict(fit, dpar = "sigma", type = "link")
predict(fit, newdata = data.frame(x = c(0, 1)), dpar = "mu")
predict(fit, type = "quantile", prob = c(0.025, 0.5, 0.975))
```

`predict.drmTMB_julia` *Predict from a legacy Julia-bridge* `drmTMB` *fit*

Description

The Julia bridge is halted/deferred future work. This compatibility method is for inspecting an existing `drmTMB_julia` object, not for a new Julia analysis. Use native TMB fits for new prediction work.

Usage

```
## S3 method for class 'drmTMB_julia'
predict(object, newdata = NULL, dpar = NULL, type = c("response", "link"), ...)

## S3 method for class 'drmTMB_julia_xfam'
predict(object, newdata = NULL, dpar = NULL, type = c("response", "link"), ...)
```

Arguments

`object`	A `drmTMB_julia` fit.
`newdata`	Optional data frame. When supplied, predictions are population-level (random effects set to zero).
`dpar`	Distributional parameter to predict. Defaults to the first (`mu`). With `newdata`, must be a location parameter (`mu` / `mu1` / `mu2`).
`type`	`"response"` (default) or `"link"`.
`...`	Reserved.

Details

With `newdata = NULL`, `predict()` returns the stored fitted values for the requested distributional parameter. With `newdata` supplied, it returns a **population-level, fixed-effect** prediction for the location parameter (`mu` / `mu1` / `mu2`): the linear predictor `X %*% beta` built from the fit's fixed-effect coefficients and a design matrix constructed from `newdata` using the training-data model terms. Group-level random effects (phylogenetic / spatial / study) are held at **zero** – a `newdata` row need not belong to any fitted group – so the result is the marginal mean at the population level, matching the native `predict.drmTMB()` contract for `newdata`. `type = "link"` returns the linear predictor; `type = "response"` applies the model's inverse link.

Predicting `sigma` / `rho12` for fresh `newdata` is not implemented; refit with `engine = "tmb"` for those.

A legacy cross-family object (`drmTMB_julia_xfam`) is narrower still: only `mu1` and `mu2` are available. Stored and new-data predictions are response means with the shared latent effect fixed at `u = 0`; they are not marginal means. Cross-family covariance, fixed-effect Wald inference, and scale-axis prediction are unavailable because that legacy bridge did not retain the required payload.

Value

A numeric vector of predictions, length `nrow(newdata)` when `newdata` is supplied.

---

predict.drm_pair_association

Predict a frozen-margin pair association

Description

With no `newdata` or `type`, this method preserves the historical `fitted()`-style output of the two frozen margins. `type = "link"` returns the association linear predictor and `type = "eta"` returns its bounded latent-normal association transform; `type = "response"` is a compatibility alias for `"eta"`. The beta Bernoulli x ordinary-NB2 route also admits new-data prediction from its fixed-effect association formula.

Usage

```
## S3 method for class 'drm_pair_association'
predict(
  object,
  newdata = NULL,
  type = NULL,
  se.fit = FALSE,
  interval = c("none", "confidence"),
  level = 0.95,
  ...
)
```

Arguments

object	A drm_pair_association object.
newdata	Optional data frame for beta Bernoulli x ordinary-NB2 association prediction. Its terms, factor levels, contrasts, and columns must match the fitted association formula.
type	Prediction scale: "link" for X_A %*% alpha or "eta" for 0.999999 * tanh(X_A %*% alpha). "response" is an alias for "eta". Omit type together with newdata to retain the historical frozen-margin fitted output.
se.fit	Logical; return pointwise standard errors on the requested scale. Eta-scale standard errors use the delta method.
interval	"none" or "confidence". Confidence limits are pointwise link-scale Wald limits transformed monotonically to eta when type = "eta".
level	Confidence level in (0, 1).
...	Must be empty.

Value

With omitted type and newdata, the frozen marginal fitted values. Without uncertainty, a numeric association prediction on the requested scale. With interval = "confidence", a matrix with fit, lwr, and upr columns. With se.fit = TRUE, a list containing fit and se.fit; fit is the three-column matrix when an interval is requested.

See Also

vcov.drm_pair_association(), confint.drm_pair_association()

Examples

```
set.seed(20260801)
n <- 30
dat <- data.frame(
  x1 = seq(-1.2, 1.2, length.out = n),
  x2 = rep(c(-0.5, 0.5), length.out = n)
)
z_binary <- rnorm(n)
eta <- 0.999999 * tanh(-0.1 + 0.4 * dat$x1 - 0.2 * dat$x2)
z_count <- eta * z_binary + sqrt(1 - eta^2) * rnorm(n)
dat$binary <- as.integer(
  z_binary > qnorm(plogis(-0.2 + 0.3 * dat$x1), lower.tail = FALSE)
)
dat$count <- qnbinom(
  pnorm(z_count), mu = exp(0.5 + 0.2 * dat$x2), size = 4
)
binary_fit <- drmTMB(bf(mu = binary ~ x1), binomial(), dat)
count_fit <- drmTMB(
  bf(mu = count ~ x2, sigma = ~ 1), nbinom2(), dat
)
assoc <- associate_pairs(
```

```
  binary_fit, count_fit,
  kernel = latent_normal(), association = ~ x1 + x2
)
new_dat <- data.frame(x1 = c(-1, 0, 1), x2 = 0)
eta_prediction <- predict(
  assoc,
  newdata = new_dat,
  type = "eta"
)
eta_prediction
```

`prediction_grid` *Build prediction grids for distributional-parameter summaries*

Description

`prediction_grid()` creates explicit `newdata` grids for `predict_parameters()` and `marginal_parameters()`. It does not fit, predict, average, or plot. The helper records which terms were varied, which terms were fixed, and which grid rule was used so later interpretation and plotting helpers do not hide those choices.

Usage

```
prediction_grid(object, ...)

## S3 method for class 'drmTMB'
prediction_grid(
  object,
  focal = NULL,
  at = list(),
  condition = list(),
  margin = c("mean_reference", "empirical"),
  n = 50L,
  weights = c("equal", "proportional"),
  ...
)
```

Arguments

`object`	A `drmTMB` fit that retained its fitted model data.
`...`	Reserved for future options.
`focal`	Optional character vector of predictor names to vary.
`at`	Optional named list of values for focal predictors. Focal numeric predictors without an `at` entry use an evenly spaced sequence over the fitted range. Focal factors without an `at` entry use all fitted levels.
`condition`	Optional named list of non-focal predictors to hold at supplied values.

margin	Grid rule. "mean_reference" returns one row for each focal combination with nuisance predictors set to reference values. "empirical" crosses focal combinations with the fitted model rows.
n	Number of points for automatically generated numeric focal grids.
weights	Metadata label for later marginalisation helpers. The current function records the choice but does not compute weighted summaries.

Details

Use margin = "mean_reference" for adjusted predictions at named covariate values. Use margin = "empirical" when the target is an average over the fitted covariate distribution after replacing one or more focal predictors.

With margin = "mean_reference", focal terms vary across the requested grid and all other predictors are set to reference values: numeric predictors use their fitted-row mean, factors use their first fitted level, character predictors use their first fitted value with fitted levels preserved, and logical predictors use their first fitted value unless supplied through condition.

With margin = "empirical", focal terms are crossed with the fitted model rows. Non-focal predictors keep their observed fitted-row values unless supplied through condition. This produces a counterfactual-style grid that can be passed to marginal_parameters(..., by = focal_terms) for simple empirical averaging.

Value

A data frame with class drm_prediction_grid. The ordinary columns are valid newdata columns for the fitted model. Attribute "prediction_grid" stores focal_terms, conditioned_terms, margin, weights, grid_source, reference_terms, predictor_terms, n_source_rows, and n_grid_rows.

Examples

```
set.seed(20260523)
n <- 48
x <- seq(-1.5, 1.5, length.out = n)
habitat <- factor(rep(c("reef", "sand"), length.out = n))
eta <- 0.4 + 0.7 * x + ifelse(habitat == "reef", 0.25, -0.15)
sigma <- exp(-0.35 + 0.15 * x)
dat <- data.frame(y = eta + rnorm(n, sd = sigma), x = x, habitat = habitat)
fit <- drmTMB(bf(y ~ x + habitat, sigma ~ x), data = dat)

grid <- prediction_grid(
  fit,
  focal = "x",
  at = list(x = c(-1, 0, 1)),
  condition = list(habitat = "reef")
)
predict_parameters(fit, newdata = grid, dpar = c("mu", "sigma"), conf.int = TRUE)

empirical_grid <- prediction_grid(
  fit,
  focal = "habitat",
  at = list(habitat = levels(dat$habitat)),
```

```
  margin = "empirical"
)
marginal_parameters(fit, newdata = empirical_grid, dpar = "mu", by = "habitat")
```

`predict_parameters` *Predict distributional parameters in long format*

Description

`predict_parameters()` returns predicted distributional parameters from a `drmTMB` fit in one long data frame. It is a compact data surface for interpretation tables, plotting helpers, and marginalisation helpers: the same grid can hold location (which can differ from the response mean), scale, shape, probability, and coscale quantities.

Usage

```
predict_parameters(object, ...)

## S3 method for class 'drmTMB'
predict_parameters(
  object,
  newdata = NULL,
  dpar = NULL,
  type = c("response", "link"),
  include_newdata = TRUE,
  conf.int = FALSE,
  conf.level = 0.95,
  ...
)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future options.
`newdata`	Optional data frame for prediction. If omitted, fitted rows are used.
`dpar`	Optional character vector of distributional parameters to predict, such as `"mu"`, `"sigma"`, `"nu"`, `"rho12"`, `"sigma1"`, or `"sigma2"`, plus fitted random-effect scale model names such as `"sd(id)"`. `NULL` predicts all fitted distributional parameters.
`type`	Prediction scale: `"response"` or `"link"`.
`include_newdata`	Logical; when `TRUE` and `newdata` is supplied, append the supplied covariate columns to the returned table.
`conf.int`	Logical; include Wald fixed-effect confidence intervals when available for the supplied prediction grid.
`conf.level`	Confidence level for Wald intervals when `conf.int = TRUE`.

Details

The helper calls `predict.drmTMB()` for each requested distributional parameter. With `newdata = NULL`, predictions use the fitted rows. With `newdata` supplied, predictions are fixed-effect, population-level predictions for those rows, matching `predict.drmTMB()`. Use this table when the reader needs distributional-parameter values on an explicit covariate grid. Use `marginal_parameters()` when the target is an average over rows rather than a row-by-row prediction table.

By default, the table includes interval provenance columns with `conf.status = "not_requested"` and `interval_source = "not_available"`. When `conf.int = TRUE` and `newdata` is supplied for ordinary fixed-effect distributional parameters, the helper adds Wald fixed-effect intervals from the fitted coefficient covariance and records the requested confidence level. These are population-level intervals for the supplied grid. Link-scale intervals are computed on the linear predictor scale; response-scale intervals use the model link and a delta method standard error. They do not include random-effect mode uncertainty, profile-likelihood uncertainty, or uncertainty for direct random-effect scale models.

For native `drmTMB` fits, the stable core columns retain this relative order: `row`, `row_label`, `dpar`, `component`, `type`, `estimate`, `conf.status`, and `interval_source`. `conf.status` describes the requested interval result or availability; `interval_source` identifies its provenance. Interval columns appear only when `conf.int = TRUE`, in the established position after `estimate` and before `conf.status`. Supplied `newdata` columns are appended after the core and optional interval columns, with a `newdata_` prefix if their names would collide with this schema and a stable suffix when two supplied names would otherwise collide after prefixing.

Value

A data frame whose stable native core columns retain this relative order: `row`, `row_label`, `dpar`, `component`, `type`, `estimate`, `conf.status`, and `interval_source`. `conf.status` is an interval result or availability state; `interval_source` is interval provenance. Only when `conf.int = TRUE`, `std.error`, `conf.low`, `conf.high`, and `conf.level` are included after `estimate` and before the status/provenance columns. When `include_newdata = TRUE`, supplied `newdata` columns are appended after the schema columns and renamed with `newdata_` plus, when needed, a stable uniqueness suffix when they collide.

Examples

```
set.seed(20260522)
n <- 36
x <- seq(-1.5, 1.5, length.out = n)
sigma <- exp(-0.35 + 0.2 * x)
dat <- data.frame(
  y = 0.4 + 0.7 * x + rnorm(n, sd = sigma),
  x = x
)
fit <- drmTMB(bf(y ~ x, sigma ~ x), data = dat)
grid <- data.frame(x = c(-1, 0, 1))
pred <- predict_parameters(
  fit,
  newdata = grid,
  dpar = c("mu", "sigma"),
  conf.int = TRUE
```

```
)
pred

predict_parameters(
  fit,
  newdata = grid,
  dpar = "sigma",
  type = "link",
  include_newdata = FALSE,
  conf.int = TRUE
)
```

`profile.drmTMB` *Compute profile-likelihood curves for fitted model targets*

Description

`profile()` computes and returns the full likelihood-profile curve for one or more direct `profile_targets()`. It is a diagnostic companion to `confint.drmTMB()`. Use `confint.drmTMB()` for interval tables, especially with the fast endpoint engine; use `profile()` followed by `plot()` when you need to see whether a target has a peaked, flat, one-sided, or boundary-like likelihood shape.

Usage

```
## S3 method for class 'drmTMB'
profile(
  fitted,
  parm,
  level = 0.95,
  trace = FALSE,
  profile_precision = c("default", "fast"),
  profile_maxit = NULL,
  compare = FALSE,
  first_pass_ystep = 0.5,
  first_pass_ytol = 2,
  ...
)
```

Arguments

`fitted`	A drmTMB fit.
`parm`	Character or integer vector selecting direct profile targets. Use `profile_targets()` to inspect available names. Unlike `confint.drmTMB()`, this helper always uses the full `TMB::tmbprofile()` curve because the curve itself is the diagnostic.
`level`	Confidence level used for the likelihood-ratio cutoff and interval endpoint annotations.
`trace`	Logical; passed to `TMB::tmbprofile()`.

profile_precision
: Profile-control shortcut. "default" leaves TMB::tmbprofile() controls unchanged. "fast" supplies ystep = 0.5 and ytol = 2 unless the caller supplies those controls in ....

profile_maxit
: Optional positive whole number passed to TMB::tmbprofile() as maxit.

compare
: Logical; if TRUE, run a coarse first-pass profile and then the requested profile controls so the returned object can compare curve shape and elapsed time.

first_pass_ystep, first_pass_ytol
: Coarse TMB::tmbprofile() controls used only when compare = TRUE. The dense pass uses profile_precision, profile_maxit, and ....

...
: Additional arguments passed to TMB::tmbprofile(). drmTMB supplies the profiled obj, name, lincomb, and trace arguments internally; set the profile target with parm.

Details

The returned x-axis values are transformed to the same scale shown by profile_targets(). For example, SD and scale targets are shown on their public positive scale, and correlation targets are shown on the correlation scale. The y-axis diagnostic is likelihood-ratio distance, 2 * (profile_nll - min(profile_nll)), so a flatter curve indicates weaker likelihood support around the fitted value.

Value

A data frame with class "profile.drmTMB". The main columns are parm, profile_value, profile_value_link, objective, delta_objective, delta_deviance, estimate, profile_pass, elapsed, profile_source, conf.low, conf.high, conf.status, and profile.message.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
prof <- profile(fit, parm = "sigma", profile_precision = "fast")
head(prof)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  plot(prof)
}
```

---

profile_targets *List confidence-interval targets for a fitted model*

Description

profile_targets() shows the names that can be supplied to confint.drmTMB(). The table also records whether each row is currently ready for direct profile-likelihood intervals. This helps users inspect the fitted object before starting an expensive profile. Full q4 unstructured and phylogenetic correlation summaries are derived targets; direct SD and scale targets can still fail on a weak, boundary-limited, or one-sided profile.

Usage

```
profile_targets(object, ready_only = FALSE)
```

Arguments

object	A drmTMB fit.
ready_only	Logical; if TRUE, return only targets whose profile_ready column is TRUE.

Details

Use ready_only = TRUE for the fastest inspection path before calling confint(..., method = "profile"). Use the target_class column to filter fixed effects, random-effect SDs, residual correlations, and other variance-component rows before a long profile run.

Value

A data frame with columns parm, target_class, dpar, term, tmb_parameter, index, estimate, link_estimate, scale, transformation, target_type, profile_ready, and profile_note. target_type is "direct" for a target that maps to a single fitted TMB parameter, "derived" for a transformed or multi-parameter quantity, or "constrained" for a target that needs a model-specific constrained profile. profile_ready = TRUE means the target is direct or constrained and the fitted object retained the TMB object needed for confint.drmTMB() with method = "profile". Common profile_note values are "ready", "tmb_object_required", "missing_tmb_parameter", "point_fit_only_zi_nbinom2_sigma_q1", "point_fit_only_zero_one_beta_phylo_q1", "point_fit_only_zero_o
"point_fit_only_zero_one_beta_phylo_coi_q1", "point_fit_only_zero_one_beta_animal_q1",
"point_fit_only_zero_one_beta_animal_zoi_q1", "point_fit_only_zero_one_beta_animal_coi_q1",
"point_fit_only_zero_one_beta_relmat_q1", "point_fit_only_zero_one_beta_relmat_zoi_q1",
"point_fit_only_zero_one_beta_relmat_coi_q1", "point_fit_only_zero_one_beta_spatial_q1",
"point_fit_only_zero_one_beta_phylo_interaction_q1", "point_fit_only_zero_one_beta_phylo_interaction
"point_fit_only_zero_one_beta_phylo_interaction_coi_q1", "point_fit_only_zero_one_beta_zoi_q1",
"point_fit_only_zero_one_beta_coi_q1", "derived_target", and "derived_unstructured_correlation".
A point_fit_only_* note records a direct point-estimate target that is deliberately unavailable to profiling: the zi-nbinom2 note covers the ordinary q1 scale route, and the zero-one-beta notes cover that bounded family's structured q1 mean route and its zero- and one-inflation routes. Derived variance-ratio summaries such as repeatability and phylogenetic signal are listed as point-estimate targets with profile_ready = FALSE.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
profile_targets(fit)
profile_targets(fit, ready_only = TRUE)
```

qq_plot *Normal QQ plot of randomized quantile residuals*

Description

qq_plot() draws a normal QQ plot of drm_quantile_residuals() against their N(0,1) order-statistic theoretical quantiles, with the y = x reference line dotted. Points on the reference line are no detectable departure from N(0,1); systematic curvature away from it flags a mis-specification of the fitted distributional form.

Usage

```
qq_plot(object, seed = NULL, nsim = 1L, response = NULL, ...)
```

Arguments

object	A drmTMB fit.
seed	Optional single integer seed, passed to drm_quantile_residuals().
nsim	Number of Dunn-Smyth realizations to overplot as an envelope; passed to drm_quantile_residuals().
response	For a bivariate biv_gaussian fit, 1 or 2, selecting which response's marginal residuals to plot; see fitted_distribution(). Must be NULL (the default) for univariate model types.
...	Reserved for future options.

Details

When nsim > 1, a pale grey envelope (with a darker outline) overplots the per-rank range across the nsim Dunn-Smyth realizations, so a single randomized draw is not over-read; the first realization's points are drawn on top.

This is fixed-effect adequacy only – see drm_quantile_residuals(). See worm_plot() for the detrended variant that makes systematic bends easier to read.

What this detects – and does not. Same scope as worm_plot() – shape/atom mis-specification a family cannot reabsorb through its own free parameters is detected with gated power >= 0.8 (heavy tails, ignored overdispersion or zero-inflation in a no-free-dispersion family, ignored truncation, a missing zero/one atom); a mis-specification a free nuisance/dispersion/inflation parameter absorbs (heteroscedasticity via Student-t nu, missing zero-inflation via nbinom2 sigma, a constant-vs-covariate inflation mechanism for hurdle_nbinom2/ zero_one_beta) is a genuine structural blind spot, not evidence of adequacy; the same mechanism mis-spec for zi_poisson/zi_nbinom2 is sample-size-limited rather than structurally blind but stays far below 0.8 even at n = 3000. See residuals.drmTMB()'s Details for the full gated-campaign breakdown (tweedie: 99 of 400 seeds locally, full run deferred to Totoro) (docs/dev-log/simulation-artifacts/2026-07-12-dg3-power-arm-gated/).

Value

A ggplot object.

Examples

```
set.seed(20260712)
n <- 60
x <- stats::rnorm(n)
dat <- data.frame(y = 0.5 + 0.8 * x + stats::rnorm(n), x = x)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), family = gaussian(), data = dat)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  qq_plot(fit)
}
```

`random_effect_scale_formulas`

Random-effect scale formula syntax

Description

`sd(group) ~ predictors` and its siblings are formula-only syntax for modelling random-effect standard deviations. They are captured by `drm_formula()` or `bf()` and are not evaluated as ordinary R calls. In particular, `sd(group) ~ predictors` does not replace `stats::sd()`.

Details

The currently documented syntax is:

- `sd(group) ~ predictors` for univariate Gaussian location random-effect scale models;
- `sd1(group) ~ predictors` and `sd2(group) ~ predictors` for bivariate Gaussian location random-effect scale models targeting `mu1` and `mu2`;
- `sd(group, level = "phylogenetic") ~ predictors` for univariate Gaussian phylogenetic location random-effect scale models;
- `sd1(group, level = "phylogenetic") ~ predictors` and `sd2(group, level = "phylogenetic") ~ predictors` for bivariate Gaussian phylogenetic location random-effect scale models targeting `mu1` and `mu2`.

`sd(group, level = "phylogenetic")` and its bivariate siblings are the generic spelling for these targets. The historical spellings `sd_phylo(species) ~ predictors`, `sd_phylo1(species) ~ predictors`, and `sd_phylo2(species) ~ predictors` are **deprecated (soft)**: they still parse and fit identically, but emit a one-time deprecation warning per session and should not be used in new formulas. Future spatial, animal-model, and user-supplied relatedness direct-SD routes are planned to use the same `level =` grammar, such as `sd(group, level = "spatial")` or `sd1(group, level = "animal")`, rather than adding parallel `sd_spatial*()`, `sd_animal*()`, and `sd_relmat*()` families.

These formulas model the standard deviation of a latent random-effect block. They are distinct from residual scale formulas such as `sigma ~ predictors` and from latent correlation formulas such as `corpair()`. Non-Gaussian random-effect scale formulas, spatial/animal/relmat direct-SD formulas, and explicit coefficient-specific targets such as `sd(group, dpar = "mu", coef = "slope") ~ predictors` remain planned.

Value

A formula-syntax reference page; no object is returned.

Examples

```
bf(y ~ x + (1 | id), sigma ~ z, sd(id) ~ habitat)

bf(
  mu1 = y1 ~ x + (1 | p | id),
  mu2 = y2 ~ x + (1 | p | id),
  sd1(id) ~ habitat,
  sd2(id) ~ habitat,
  rho12 = ~ x
)

bf(
  y ~ x + phylo(1 | species, tree = tree),
  sd(species, level = "phylogenetic") ~ ecology
)
```

---

ranef *Extract conditional random-effect estimates*

Description

`ranef()` returns conditional random-effect estimates for one fitted random effect block, or all fitted random-effect blocks when `dpar = NULL`.

Usage

```
ranef(object, ...)

## S3 method for class 'drmTMB'
ranef(object, dpar = NULL, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.
`dpar`	Optional random-effect block name, such as `"mu"`, `"sigma"`, `"phylo_mu"`, or `"spatial_mu"`.

Details

The returned blocks use the internal `drmTMB` structure: `values` are on the model scale, `latent` are the corresponding standard-normal latent effects, and `terms` split model-scale values by random-effect term.

Value

A named list of random-effect blocks when `dpar = NULL`, otherwise one random-effect block.

Examples

```
set.seed(20260525)
id <- factor(rep(letters[1:8], each = 8))
x <- rep(seq(-1, 1, length.out = 8), times = 8)
u <- rnorm(nlevels(id), sd = 0.9)
dat <- data.frame(
  y = 0.2 + 0.7 * x + u[id] + rnorm(length(x), sd = 0.3),
  x = x,
  id = id
)
fit <- drmTMB(bf(y ~ x + (1 | id), sigma ~ 1), data = dat)
names(ranef(fit))
head(ranef(fit, "mu")$terms[["(1 | id)"]])
```

`relmat` *User-supplied relatedness structured-effect marker*

Description

`relmat()` marks syntax for a validated user-supplied relatedness matrix. It is the lower-level route for latent group-level dependence structures that are not best named as `animal()`, `phylo()`, or `spatial()`: for example a genomic relationship matrix, a laboratory relatedness kernel, or a graph, river-network, areal, or Gaussian Markov random-field precision matrix built outside `drmTMB` and checked by the analyst. If the matrix is known sampling covariance among observed estimates, use `meta_V()` instead.

Usage

```
relmat(term, K = NULL, Q = NULL)
```

Arguments

`term`	Structured random-effect term, such as `1 | id` or `1 + x | id`.
`K`	Known relatedness or covariance matrix for the documented fitted univariate and bivariate routes.
`Q`	Known precision or inverse covariance matrix for the documented fitted routes. The exact bivariate REML exception requires `K`; bivariate `Q` remains available under ML.

Details

Use K for a covariance or relatedness matrix and Q for an inverse covariance or precision matrix. A correlation matrix with diagonal 1 is a natural K input because the fitted relatedness SD supplies the latent variance scale. The fitted known-matrix routes are a univariate Gaussian mu random intercept, for example relmat(1 | line, Q = Q), the first bivariate Gaussian q=2 location covariance from matching labelled terms in mu1 and mu2, matching univariate Gaussian mu and sigma intercept terms estimate one relatedness mean-scale correlation, and the constant all-four q=4 location-scale block comes from matching labelled terms in mu1, mu2, sigma1, and sigma2, for example relmat(1 | p | line, Q = Q). The exact bivariate Gaussian REML exception requires matching labelled relmat(1 | p | line, K = K) location intercepts in mu1 and mu2, the same named supplied covariance K, intercept-only residual formulas, complete pairs, unit weights, and no additional model layer. This exception has point_fit_recovery evidence only; bivariate Q, slopes, q4+, intervals, and coverage remain outside it. The univariate Gaussian mu path also supports one numeric slope, for example relmat(1 + x | line, Q = Q), as independent intercept and slope fields with separate SDs and no intercept-slope correlation. The exact q1 sigma one-slope route is also fitted for K and Q. Implemented bivariate labelled ML cells extend through the q2, q4, q6, and q12 layouts recorded in the capability ledger; additional structured-slope layouts, structured slope correlations, predictor-dependent corpair() regression, and relatedness sd*() direct-SD grammar remain planned. relmat() is intentionally separate from meta_V(), which adds known sampling covariance among observations, and from residual rho12, which models within-observation bivariate residual correlation.

Value

A formula marker; never evaluated by users.

Examples

```
# Fitted: a genomic relatedness matrix for among-line genetic variance.
bf(seed_mass ~ temperature + relmat(1 | line, K = G),
  sigma ~ temperature
)

# Fitted: a user-built sparse precision for another dependence structure.
bf(growth ~ treatment + relmat(1 | plot, Q = Q_plot),
  sigma ~ treatment
)
```

---

residuals.drmTMB *Extract model residuals*

Description

residuals() returns response residuals or Pearson-style residuals from a drmTMB fit.

Usage

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

Arguments

object	A drmTMB fit.
type	Residual type: "response", "pearson", or "quantile".
...	Reserved for future residual options; for type = "quantile", forwarded to drm_quantile_residuals() (seed, nsim, response).

Details

For univariate Gaussian models, response residuals are y - mu. Pearson residuals divide by the fitted observation standard deviation. If a dense known sampling covariance was used, Pearson residuals are whitened by the fitted total observation covariance.

For lognormal models, response residuals are y - fitted_mean. Pearson residuals are computed on the log-response scale as (log(y) - mu) / sigma. For Gamma models, response residuals are y - mu and Pearson residuals divide by the fitted Gamma standard deviation mu * sigma, where sigma is the coefficient of variation. For Tweedie models, response residuals are y - mu and Pearson residuals divide by sqrt(sigma^2 * mu^nu). For zero-one beta models, response residuals are observed proportions minus the unconditional fitted mean, including exact zero-one boundary mass. For beta-binomial models, response residuals are observed success proportions minus fitted mu, and Pearson residuals divide by the fitted beta-binomial proportion standard deviation. For binomial models, response residuals are observed success proportions minus fitted event probability, and Pearson residuals divide by sqrt(mu * (1 - mu) / trials). For cumulative-logit ordinal models, response residuals are the observed ordered-category score minus the fitted expected score, and Pearson residuals divide by the fitted category-score standard deviation. For Poisson models, response residuals are y - mu and Pearson residuals divide by sqrt(mu). For zero-inflated Poisson models, response residuals are y - (1 - zi) * mu, and Pearson residuals divide by sqrt((1 - zi) * mu * (1 + zi * mu)). For negative-binomial 2 models, Pearson residuals divide by sqrt(mu + sigma^2 * mu^2). For zero-truncated NB2 models, response residuals are y - mu / (1 - Pr_NB2(0)), and Pearson residuals divide by the conditional positive-count standard deviation. For hurdle NB2 models, response residuals are y - (1 - hu) * mu / (1 - Pr_NB2(0)), and Pearson residuals divide by the unconditional standard deviation implied by the hurdle-zero mixture. For zero-inflated NB2 models, response residuals are y - (1 - zi) * mu, and Pearson residuals divide by the unconditional standard deviation implied by the structural-zero mixture.

For bivariate Gaussian models, response residuals are returned as a two-column matrix. Pearson residuals are standardized and whitened using the fitted residual sigma1, sigma2, and rho12, or using the full row-paired observation covariance when a dense bivariate known V was supplied.

type = "quantile" returns Dunn-Smyth randomized quantile residuals from drm_quantile_residuals(): qnorm(F(y; theta_hat)) at the fitted, fixed-effect distributional parameters. As of DO-T3 batch D, all 18 fitted model_type values have a promoted entry in drm_family_dpq(), so type = "quantile" is available for every family, including bivariate biv_gaussian (where response – 1 or 2 – is REQUIRED and selects which response's MARGINAL residuals to compute; omitting it errors clearly, as does supplying it for a univariate fit; see fitted_distribution()). Spike-status families (feasibility-only, not yet DG2/DG3-verified) emit a one-time warning that the residual is exploratory, not DG-verified; none currently have this status. See drm_quantile_residuals() for the fixed-effect-only adequacy caveat: for random-effect or structured fits, these residuals are conditional on the fixed-effect prediction, not marginal, so a departure (or its absence) is evidence about fixed-effect adequacy only – never a general validity claim. Pass seed, nsim, and/or response through ... to drm_quantile_residuals().

What this detects – and does not. Evidence: a 400-seed gated simulation campaign across all 18 fitted families (tweedie: 99 of 400 seeds locally, 66/99 dispersion-arm non-convergence, full run deferred to Totoro) (`docs/dev-log/simulation-artifacts/2026-07-12-dg3-power-arm-gated/`). Under a correctly specified fixed-effect model, type-I error stays at or below the nominal rate (0.0025-0.025 at alpha = 0.05 across families); the underlying KS+PIT statistic is conservative, so power is understated, not overstated. `type = "quantile"` **detects** distributional shape and atom mis-specification that a family cannot reabsorb through its own free parameters – heavy tails fit as Gaussian, overdispersion or zero-inflation ignored by a family with no free dispersion parameter, truncation ignored, a missing zero/one atom – with gated power >= 0.8 at n = 300-400 per arm (commonly 0.9-1.0). It has a genuine **structural blind spot**, not a bug: a mis-specification that a fitted family's own free nuisance/dispersion/inflation parameter absorbs leaves the fitted-model residual marginally N(0,1) and is **not detectable** here – for example heteroscedasticity absorbed by Student-t `nu` (power 0.035 at n = 300, versus 1.0 for the same heteroscedasticity under Gaussian, which has no absorbing parameter), missing zero-inflation absorbed by `nbinom2 sigma` (power 0.035, versus 0.9625 under Poisson), and zero-inflation/ hurdle/zero-one-inflation *mechanism* mis-specification (a constant inflation probability fit when it truly varies with a covariate), which splits into two patterns under the n-ladder (tested to n = 3000): for `hurdle_nbinom2`/`zero_one_beta`, power stays flat at or below about 0.01 at every n – a genuine structural blind spot; for `zi_nbinom2`/`zi_poisson`, power rises with n (to about 0.11/0.06 at n = 3000), so the marginal is not identical under the mechanism mis-spec, but power stays far below the >= 0.8 detectable benchmark even at n = 3000, so it remains impractical to detect at realistic sample sizes. A mean-structure diagnostic, not this one, is what catches an absorbed mis-specification. Separately, `gamma-vs-lognormal` wrong-family detection is sample-size limited rather than structurally blind: power rises from about 0.19 at n = 300 to 0.79 at n = 1000 and 1.0 at n = 3000, so that specific mis-specification needs n well above 1000 to be reliably caught. A distributional-output/adequacy tick on a family never changes or implies anything about that family's own inference-tier status; see `tests/testthat/test-dg-firewall.R`.

Value

A numeric vector for univariate models, or a two-column matrix for bivariate Gaussian models. For `type = "quantile"` with `nsim > 1`, an n-by-`nsim` matrix (see `drm_quantile_residuals()`).

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
residuals(fit)
residuals(fit, type = "pearson")
residuals(fit, type = "quantile")
```

`rho12` *Extract residual correlation rho12*

Description

`rho12()` returns the residual response-response correlation from a bivariate Gaussian, bivariate lognormal, or bivariate Student-t `drmTMB` fit. For a bivariate Gaussian fit this is a response-scale

residual correlation; for `biv_lognormal()` it is correlation of the log-response residuals, not a raw-scale correlation. For `biv_student()` it is the scatter/residual correlation; at finite shared `nu`, zero correlation does not imply independent margins. Use `type = "link"` for the Fisher-z-like linear predictor whose response transform is `0.999999 * tanh(eta)`.

Usage

```
rho12(object, ...)

## S3 method for class 'drmTMB'
rho12(object, newdata = NULL, type = c("response", "link"), ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.
`newdata`	Optional data frame for prediction.
`type`	Scale of returned values: `"response"` for correlation values or `"link"` for Fisher-z-like linear predictors.

Value

A numeric vector of residual correlations, or Fisher-z-like linear predictors when `type = "link"`.

Examples

```
set.seed(20260525)
n <- 36
x <- seq(-1, 1, length.out = n)
e1 <- rnorm(n)
e2 <- 0.4 * e1 + sqrt(1 - 0.4^2) * rnorm(n)
dat <- data.frame(
  y1 = 0.2 + 0.5 * x + e1,
  y2 = -0.1 + 0.3 * x + e2,
  x = x
)
fit <- drmTMB(
  bf(mu1 = y1 ~ x, mu2 = y2 ~ x, sigma1 = ~ 1, sigma2 = ~ 1, rho12 = ~ x),
  family = biv_gaussian(),
  data = dat
)
head(rho12(fit))
rho12(fit, newdata = data.frame(x = c(-0.5, 0, 0.5)))
```

rho_latent *Extract a latent-scale correlation from a legacy Julia-bridge fit*

Description

The cross-family Julia bridge is halted/deferred future work. This compatibility extractor is retained only for an existing `drmTMB_julia_xfam` object; it does not establish a current cross-family fitting or inference capability.

Usage

```
rho_latent(object, ...)
```

Arguments

`object`	A legacy `drmTMB_julia_xfam` cross-family fit.
`...`	Unused.

Value

The latent / link-scale correlation between the two responses.

sigma.drmTMB *Extract fitted scale or dispersion*

Description

`sigma()` returns the fitted scale-like parameter from a `drmTMB` model. For univariate Gaussian location-scale models this is the fitted residual `sigma_i` vector on the response scale. For Student-t models this is the Student-t scale parameter; when `nu > 2`, the residual standard deviation is `sigma * sqrt(nu / (nu - 2))`. For skew-normal models this is the response standard deviation under the public moment parameterization, not the native Azzalini scale `omega`. For lognormal models this is the fitted standard deviation of `log(y)`. For Gamma models this is the fitted coefficient of variation. For Tweedie models this is the public scale parameter where internal dispersion is `phi = sigma^2`. For beta, zero-one beta, and beta-binomial models this is the public scale parameter where internal precision is `phi = 1 / sigma^2`. Cumulative-logit ordinal, binomial, Poisson, and zero-inflated Poisson models have no fitted residual scale parameter and return a fixed unit dispersion vector for consistency with base-R `sigma()` conventions. For negative-binomial 2, zero-truncated negative-binomial 2, hurdle negative-binomial 2, and zero-inflated negative-binomial 2 models this is the fitted overdispersion scale in the untruncated NB2 component `Var(y | component) = mu + sigma^2 * mu^2`. For bivariate Gaussian, bivariate lognormal, and bivariate Student-t models it returns a roundable list with fitted `sigma1` and `sigma2` vectors. The bivariate-lognormal values are log-response SDs; the bivariate-Student values are Student-t scales, not marginal SDs.

Usage

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

Arguments

object	A drmTMB fit.
...	Reserved for future scale-extractor options.

Details

In meta-analytic models fitted with meta_V(V = V), this is the modelled residual heterogeneity scale, not the square root of the known sampling variance plus residual variance. Simulation and Pearson residuals combine known sampling covariance with residual scale internally.

Value

A numeric vector for univariate models, or a named, roundable list of numeric vectors for bivariate Gaussian, bivariate lognormal, or bivariate Student-t models.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ x), data = dat)
sigma(fit)
```

---

simulate.drmTMB *Simulate from a fitted model*

Description

simulate() draws new response values from the fitted drmTMB model. For univariate Gaussian models with known sampling covariance, simulation uses the total observation covariance implied by the known sampling covariance plus the fitted residual scale. For Student-t models, simulation uses fitted mu, sigma, and nu. For lognormal models, simulation uses fitted log-scale mu and sigma. For Gamma models, simulation uses fitted mean mu and coefficient of variation sigma. For beta models, simulation uses fitted mean mu and public scale sigma with internal phi = 1 / sigma^2. For Tweedie models, simulation uses fitted mu, public sigma, and power nu, with internal dispersion phi = sigma^2. For zero-one beta models, simulation draws exact boundary values from the fitted zoi/coi probabilities and interior values from the fitted beta component. For beta-binomial models, simulation draws latent success probabilities from the fitted beta distribution and then success counts from the stored trial denominators. For binomial models, simulation draws success counts from the fitted event probability and stored trial denominators. For cumulative-logit ordinal models, simulation draws ordered categories from the fitted cumulative-logit probabilities. For Poisson models, simulation uses the fitted mean mu. For zero-inflated Poisson models, simulation uses fitted conditional mean mu and structural-zero probability zi. For negative-binomial 2 models, simulation uses fitted mu and overdispersion scale sigma, with Var(y) = mu + sigma^2 * mu^2;

zero-truncated NB2 models draw from this NB2 component conditional on positive counts. The zero-inflated NB2 path adds structural-zero probability `zi`; the hurdle NB2 path adds hurdle-zero probability `hu` and draws nonzero counts from the zero-truncated NB2 component. For bivariate Gaussian models without known sampling covariance, simulation uses the fitted `mu1`, `mu2`, `sigma1`, `sigma2`, and residual `rho12`. If a dense bivariate known `V` was supplied, simulation uses the full row-paired observation covariance `V + Omega`. For bivariate Student-t models, simulation instead uses one shared chi-squared scale-mixture draw per response pair together with the fitted shared `nu`; independent marginal t draws would be a different model.

Usage

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

Arguments

`object`	A `drmTMB` fit.
`nsim`	Number of simulated data sets.
`seed`	Optional random-number seed. The previous `.Random.seed` state is restored after simulation.
`re.form`	`NULL` (the default) draws fresh random effects for every replicate (marginal simulation). `NA` holds every random effect fixed at its fitted conditional-mode estimate (conditional simulation).
`...`	Reserved for future simulation options.

Details

By default (`re.form = NULL`), models with ordinary grouped random effects draw a **fresh** random-effect realization for every replicate, so replicate-to-replicate variability reflects both the between-group and the residual sources of variance (matching the lme4/glmmTMB `re.form` convention). Pass `re.form = NA` to simulate conditionally on the fitted random effects instead (holding every random effect fixed at its conditional-mode estimate, the behaviour this function had before `re.form` was added). Marginal simulation also redraws a single-endpoint (`q == 1`) phylogenetic, spatial, relatedness-matrix (`relmat`), animal-model, or `phylo_interaction` structured `mu` random effect from its fitted covariance `sd^2 * Q^-1`. It does not yet support a structured mu random effect correlated across more than one trait or distributional parameter (`q > 1`, including a multi-endpoint `phylo_interaction` term), correlated covariance-block random effects, predictor-dependent random-effect correlation (`corpair`) regression, or a modelled (heteroscedastic) random-effect scale; `simulate()` throws an informative error naming the unsupported structure for those models and directs the user to `re.form = NA`. Models without random effects are unaffected by `re.form`.

Value

A data frame. Univariate models return one column per simulation. Bivariate models return paired columns named `sim_<j>_y1` and `sim_<j>_y2`.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
simulate(fit, nsim = 2, seed = 1)
```

`skew_normal` *Skew-normal response family*

Description

`skew_normal()` defines a one-response skew-normal distribution with formulas for location `mu`, residual standard deviation `sigma`, and residual slant `nu`.

Usage

```
skew_normal()
```

Details

The implemented family is univariate: `mu = eta_mu`, `log(sigma) = eta_sigma`, and `nu = eta_nu`. The likelihood transforms internally to the native Azzalini location `xi`, scale `omega`, and slant `alpha = nu`, but user-facing methods keep the public moment parameterization: `fitted()` returns `E[y] = mu`, `stats::sigma()` returns `SD[y] = sigma`, and `predict(..., dpar = "nu")` returns the residual slant. Positive `nu` gives right-skewed residuals, negative `nu` gives left-skewed residuals, and `nu = 0` reduces to the Gaussian location-scale likelihood.

Ordinary unlabelled random intercepts and independent numeric slopes such as `(1 | id)` and `(0 + x | id)` may enter `mu`. Correlated or labelled `mu` slopes, `sigma` or `nu` random effects, `sd(group)` scale formulas, structured effects, known sampling covariance, bivariate skew-normal models, residual `rho12`, and latent `skew(id)` syntax remain unsupported.

Value

A `drm_family` object.

Examples

```
skew_normal()
```

`spatial` *Spatial structured-effect marker*

Description

`spatial()` marks structured spatial dependence. The first fitted path is the univariate Gaussian location random intercept `spatial(1 | site, coords = coords)`, where `coords` is a matrix or data frame with one row per site or one row per observation. The univariate Gaussian location path also supports one numeric slope, `spatial(1 + x | site, coords = coords)`, as independent intercept and slope fields with separate SDs and no intercept-slope correlation. Matching labelled bivariate Gaussian `mu1`/`mu2` terms fit the first q=2 coordinate-spatial location covariance, matching univariate Gaussian `mu` and `sigma` intercept terms fit one spatial mean-scale correlation, and matching labelled all-four `mu1`/`mu2`/`sigma1`/`sigma2` terms fit the first constant q=4 location-scale block. Fitted non-Gaussian spatial routes cover the univariate student `mu` q=1 intercept or one-slope, ordinary Poisson and NB2 `mu` q=1 intercepts or one-slopes, zero-inflated Poisson `zi` q=1 intercept, and the NB2 `sigma` q=1 one-slope, all at recovery grade: trust the point estimate, not the interval. Mesh inputs, multiple structured slopes, residual-scale structured slopes beyond the NB2 one-slope gate, slope correlations, and predictor-dependent spatial `corpair()` regression remain planned.

Usage

```
spatial(term, coords = NULL, mesh = NULL)
```

Arguments

`term`	Structured random-effect term, such as `1 | site`.
`coords`	Coordinate object, such as a data frame or matrix of spatial coordinates.
`mesh`	Planned precomputed mesh object.

Value

A formula marker; never evaluated by users.

Examples

```
# Fitted for univariate Gaussian mu with coords:
bf(y ~ x + spatial(1 | site, coords = coords), sigma ~ z)
# Fitted first q=2 bivariate spatial location block:
bf(
  mu1 = y1 ~ x + spatial(1 | p | site, coords = coords),
  mu2 = y2 ~ x + spatial(1 | p | site, coords = coords),
  sigma1 = ~ 1,
  sigma2 = ~ 1,
  rho12 = ~ 1
)
# Planned:
bf(y ~ x + spatial(1 | site, mesh = mesh), sigma ~ z)
```

spatial_coords *Transform longitude and latitude to planar spatial coordinates*

Description

Converts WGS84 longitude/latitude columns to an explicitly selected projected coordinate reference system. Spatial mesh models use metric planar distances; this helper deliberately does not infer a UTM zone or treat decimal degrees as planar coordinates.

Usage

```
spatial_coords(data, longitude, latitude, crs_out)
```

Arguments

data	A non-empty data frame containing the longitude and latitude columns.
longitude, latitude	Unquoted or quoted names of the longitude and latitude columns, respectively.
crs_out	A projected output CRS understood by sf::st_crs(), for example 32610 or "EPSG:32610".

Value

A two-column numeric matrix with class drmTMB_coords. Its crs attribute records the projected CRS and it can be supplied to make_mesh().

Examples

```
if (requireNamespace("sf", quietly = TRUE)) {
  locations <- data.frame(lon = c(-123.2, -123.1), lat = c(49.2, 49.3))
  spatial_coords(locations, lon, lat, crs_out = 32610)
}
```

structured_effects *Extract structured-effect metadata*

Description

structured_effects() returns the structured random-effect markers that drmTMB parsed and fitted. It gives downstream tools a stable post-fit metadata table so they do not need to grep or re-parse formula text.

Usage

```
structured_effects(object, ...)

## Default S3 method:
structured_effects(object, ...)

## S3 method for class 'drmTMB'
structured_effects(object, ...)
```

Arguments

object	A drmTMB fit.
...	Reserved for future extractor options.

Details

The current fitted marker grammar is:

- phylo(1 | species, tree = tree)
- spatial(1 | site, coords = coords)
- animal(1 | id, pedigree = pedigree), animal(1 | id, A = A), or animal(1 | id, Ainv = Ainv)
- relmat(1 | id, K = K) or relmat(1 | id, Q = Q)
- phylo_interaction(1 | plant:pollinator, tree1 = plant_tree, tree2 = pollinator_tree)

Some Gaussian routes also fit matching location-scale, bivariate, or one-slope structured blocks. Those routes still return one row per parsed structured marker, with dpars and coef_names stored as list columns.

Value

A data frame with one row per fitted structured-effect marker. The args, dpars, coef_names, endpoint_members, member_levels, provider_levels, observed_levels, endpoint_blocks, and endpoint_covariance_labels columns are list columns. Empty fits return the same columns with zero rows.

Examples

```
dat <- data.frame(
  y = c(0.1, 0.4, 0.8, 1.1),
  x = c(-1, 0, 1, 2)
)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
structured_effects(fit)
```

`student` *Student-t response family*

Description

`student()` defines a one-response Student-t distribution with formulas for location `mu`, residual scale `sigma`, and degrees of freedom `nu`.

Usage

```
student()
```

Details

Here `sigma` is the Student-t **scale**, not the response standard deviation. The density is the location-scale t evaluated at `z = (y - mu) / sigma`, so the standard deviation of `y` is `SD[y] = sigma * sqrt(nu / (nu - 2))` for `nu > 2` and is strictly larger than `sigma` (about 73% larger at `nu = 3`, shrinking to `sigma` as `nu -> Inf`). This is the one implemented family whose public `sigma` is a scale rather than `SD[y]`: the location-scale t has no closed-form standard-deviation parameterization, and both `drmTMB` and its `DRM.jl` twin fit `sigma` as the scale.

The `nu` parameter uses a log link with a lower bound of 2: `nu = 2 + exp(eta_nu)`. This keeps the fitted distribution in the finite-variance region (`nu > 2`) while still allowing heavy tails. The lower bound is a deliberate design choice, not a standard-deviation requirement: it guarantees a finite variance and a well-defined `SD[y]`. The model therefore **cannot** represent the very heavy tails of `nu <= 2` (for example a Cauchy-like `nu = 1`); data that genuinely need `nu <= 2` would require lifting the floor, which is not implemented. `check_drm()` warns when the fitted `nu` approaches the boundary at 2, where the slant of the likelihood in `nu` is weakly identified. Ordinary `mu` random intercepts and independent numeric slopes such as `(1 | id)` and `(0 + x | id)` are supported in the first Student-t mixed-model slice; correlated slopes, `sigma` random effects, and `nu` random effects remain separate planned gates.

Value

A `drm_family` object.

Examples

```
student()
```

summary.drmTMB *Summarize a fitted model*

Description

summary() returns a compact summary of fixed-effect estimates, response-scale distributional, scale, shape, random-effect SD, correlation, and fitted random-effect covariance quantities when they are present. The covariance component reports currently fitted registry-backed rows and fitted bivariate phylogenetic covariance rows, including q=2 mean-mean and q=4 endpoint rows where present. The derived component reports simple point-estimate variance ratios, such as Gaussian random-intercept repeatability and phylogenetic signal, when the ingredients are unambiguous. Derived confidence intervals are marked as unavailable until a nonlinear interval method is implemented. When TMB::sdreport() succeeds, direct response-scale parameter rows also include delta-method standard errors; descriptive fitted ranges and derived variance ratios do not. Confidence intervals are opt-in: fast Wald intervals are available for fixed effects and direct response-scale parameter rows, and slower profile-likelihood intervals are available for selected direct profile targets. Profile summaries keep Wald intervals for fixed effects unless fixed-effect profile targets are selected. Interval-aware tables include conf.status so rows without intervals can say whether an interval was not requested, needs newdata, is ready but unselected, or is currently unavailable. Use summary(fit, conf.int = TRUE) for fixed-effect and direct parameter Wald confidence intervals, and use method = "profile" with ci_parm for direct response-scale targets such as sigma, rho12, or a random-effect SD. Correlation Wald intervals use the fitted TMB correlation-link scale, equivalent to a guarded Fisher z/atanh transform, before returning lower and upper bounds on the correlation scale.

Usage

```
## S3 method for class 'drmTMB'
summary(
  object,
  conf.int = FALSE,
  level = 0.95,
  method = c("wald", "profile"),
  ci_parm = NULL,
  trace = FALSE,
  profile_precision = c("default", "fast"),
  ...
)
```

Arguments

object	A drmTMB fit.
conf.int	Logical; include confidence intervals when TRUE.
level	Confidence level for intervals.
method	Interval method used when conf.int = TRUE: "wald" for fast direct intervals or "profile" for profile-likelihood intervals on selected direct targets. summary()

does not run bootstrap intervals yet; use `confint(..., method = "bootstrap")` for the current direct-target bootstrap route.

`ci_parm` Optional character or integer vector selecting confidence interval targets. For `method = "wald"` and `method = "profile"`, targets use the `profile_targets()` namespace, such as `"sigma"`, `"rho12"`, `"sd:mu:(1 | id)"`, or `"cor:mu:cor((Intercept),x | id)"`. `NULL` selects all direct Wald-ready targets for Wald intervals and currently ready direct non-fixed targets for profile intervals. This keeps large profile runs focused on scale, variance-component, and correlation rows unless fixed-effect profile targets are requested explicitly.

`trace` Logical; passed to `TMB::tmbprofile()` for profile intervals.

`profile_precision`
Profile-control shortcut used with `method = "profile"`. `"default"` leaves `TMB::tmbprofile()` controls unchanged, while `"fast"` supplies `ystep = 0.5` and `ytol = 2` unless the caller supplies those controls in `...`.

`...` Additional arguments passed to `TMB::tmbprofile()` when `conf.int = TRUE` and `method = "profile"`.

Value

An object of class `summary.drmTMB`.

See Also

The tier definitions and per-cell evidence behind these interval targets, including random-effect standard-deviation rows, are curated in `vignette("capability-and-limits", package = "drmTMB")`: `summary()` computes intervals generically for any target, and the tier a given cell belongs to is a documentation-level curation, not a runtime guard.

Examples

```
dat <- data.frame(y = c(0.2, 0.5, 1.1, 1.4), x = c(-1, 0, 1, 2))
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)
summary(fit)
summary(fit, conf.int = TRUE)
summary(
  fit,
  conf.int = TRUE,
  method = "profile",
  ci_parm = "sigma",
  profile_precision = "fast"
)
```

summary.drmTMB_julia *Summarise a legacy Julia-bridge* drmTMB *fit*

Description

The Julia bridge is halted/deferred future work. This compatibility method inspects an existing drmTMB_julia object; it does not make Julia a current fitting or inference option. For new analyses, use native TMB fits.

Usage

```
## S3 method for class 'drmTMB_julia'
summary(
  object,
  conf.int = FALSE,
  level = 0.95,
  method = c("wald", "profile"),
  ...
)
```

Arguments

object	A drmTMB_julia fit.
conf.int	Logical; append confidence-interval columns.
level	Confidence level for the interval columns.
method	"wald" (default) or "profile"; only used when conf.int = TRUE.
...	Unused.

Details

Builds a fixed-effect coefficient table (estimate, standard error, z value, and two-sided p value, all on the linear-predictor / link scale) from the coefficients and fixed-effect covariance DRM.jl marshals back through the bridge. Standard errors are the square roots of the diagonal of vcov(object); when DRM.jl did not return a finite covariance for a route the SE / z / p columns are NA and uncertainty$status records why. The random-effect SD block (e.g. a phylogenetic SD) is reported on its positive response scale.

Set conf.int = TRUE to append Wald (default) or profile confidence-interval columns. Profile / bootstrap intervals are available only for the Gaussian phylogenetic SD target; see confint.drmTMB_julia().

Value

An object of class summary.drmTMB_julia with coefficients, random (random-effect SDs), and fit-summary scalars.

summary.drmTMB_julia_xfam *Summary for a legacy Julia cross-family fit*

Description

The cross-family Julia bridge is halted/deferred future work. This compatibility summary reports point estimates only: the bridge does not retain a named covariance matrix, so standard errors and Wald intervals are deliberately unavailable.

Usage

```
## S3 method for class 'drmTMB_julia_xfam'
summary(object, conf.int = FALSE, ...)
```

Arguments

object	A drmTMB_julia_xfam cross-family fit.
conf.int	Logical; requesting intervals errors because the bridge did not return fixed-effect covariance.
...	Unused.

Value

An object of class summary.drmTMB_julia containing point estimates and an explicit unavailable-uncertainty status.

truncated_nbinom2 *Zero-truncated negative binomial 2 response family*

Description

truncated_nbinom2() defines a one-response positive-count distribution with formulas for the untruncated NB2 mean mu and overdispersion scale sigma.

Usage

```
truncated_nbinom2()
```

Details

Adding `hu ~ predictors` to the model formula fits the corresponding hurdle NB2 model: `hu` is the probability of a hurdle zero, and nonzero counts are drawn from the zero-truncated NB2 component.

The implemented contract is `log(mu) = eta_mu`, `log(sigma) = eta_sigma`, and the count response is distributed as NB2 conditional on being greater than zero. The untruncated NB2 variance is `Var(y) = mu + sigma^2 * mu^2`, with internal `size = 1 / sigma^2`.

Ordinary zero-truncated NB2 models support first-slice random intercepts and independent numeric slopes in the log-mean predictor, such as `bf(count ~ x + (1 | id) + (0 + x | id), sigma ~ z)`. Hurdle NB2 random effects, correlated zero-truncated slopes, structured effects, and overdispersion-side random effects remain planned.

Value

A `drm_family` object.

Examples

```
truncated_nbinom2()
```

---

tweedie *Tweedie response family*

Description

`tweedie()` defines a one-response Tweedie compound Poisson-Gamma distribution for non-negative continuous responses with exact zeros.

Usage

```
tweedie()
```

Details

The implemented contract is univariate: `log(mu) = eta_mu`, `log(sigma) = eta_sigma`, `nu = 1 + plogis(eta_nu)`, `phi = sigma^2`, `E[y] = mu`, and `Var(y) = sigma^2 * mu^nu`, with `1 < nu < 2`. The public `sigma` parameter is therefore the square root of the usual Tweedie dispersion `phi`. Ordinary unlabelled random intercepts and independent numeric slopes such as `(1 | id)` and `(0 + x | id)` may enter `mu`. Correlated or labelled `mu` slopes, `sigma` or `nu` random effects, predictor-dependent `nu`, bivariate Tweedie models, structured effects, zero-inflation aliases, and hurdle aliases remain unsupported.

Value

A `drm_family` object.

Examples

```
tweedie()
```

`vcov.drmTMB` *Extract standard model-fit quantities*

Description

These methods expose `drmTMB` fits to standard base-R model summary and comparison helpers.

Usage

```
## S3 method for class 'drmTMB'
vcov(object, ...)

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

## S3 method for class 'drmTMB'
AIC(object, ..., k = 2)

## S3 method for class 'drmTMB'
BIC(object, ...)

## S3 method for class 'drmTMB'
nobs(object, ...)

## S3 method for class 'drmTMB'
df.residual(object, ...)

## S3 method for class 'drmTMB'
deviance(object, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.
`k`	Numeric penalty per parameter for `AIC()`; the default is `2`.

Details

`logLik()` returns a `"logLik"` object with `df` and `nobs` attributes so `stats::AIC()` and `stats::BIC()` use the fitted likelihood, optimized top-level parameter count, and fitted-row count consistently. `nobs()` returns the number of fitted rows after complete-case filtering. `df.residual()` returns `nobs - df`, where `df` is the number of optimized top-level parameters recorded in `logLik()`. `deviance()` returns `-2 * logLik`; for these likelihood-based distributional models this is an absolute negative twice log-likelihood value, not a saturated-model GLM deviance. `vcov()` returns the fixed-effect covariance matrix from `TMB::sdreport()` when the fit contains an `sdreport` object with `pdHess = TRUE`; the same object is available as `fit$sdr` and `fit$sdreport`. If `sdreport()` was skipped,

failed, or returned pdHess = FALSE, Wald standard errors and Wald confidence intervals are unavailable while point estimates remain usable. vcov() intentionally does not include random-effect conditional modes or derived response-scale quantities.

Value

logLik() returns an object of class "logLik". vcov() returns a numeric covariance matrix. nobs(), df.residual(), and deviance() return numeric scalars.

Experimental MSPL fits

For a fit made with estimator = "mspl", vcov() behaves differently from an ordinary maximum-likelihood fit in two ways worth knowing before you use it.

First, **the shape differs**. It returns the full outer-parameter covariance after Laplace marginalisation — the fixed effects *and* the covariance parameters in their frozen Cholesky coordinates (log_sd_mu, and eta_cor_mu for a q = 2 block) — not the fixed-effect-only block an ordinary fit returns. Code that assumes vcov(fit) has the same dimension across estimators will break.

Second, **it is a standard error, not an interval**. The matrix inverts the Hessian of the *unpenalized* Laplace log-likelihood evaluated at the MSPL estimate: the penalty obtains a finite estimate but does not describe sampling variability. Do not form coef ± 1.96 * se. Kosmidis and Firth show that Wald intervals in this setting fail to cover regardless of the nominal level, and that the failure persists even for profile penalized-likelihood intervals; the mechanism is the finiteness of the penalized estimator and its standard error, not separation as such. A second reason applies here: the MSPL estimate maximises the *penalized* criterion, so the unpenalized score is not zero at it and the usual "evaluate at the maximum likelihood estimate" justification does not transfer. The size of that departure is recorded in fit$mspl$wald$unpenalized_gradient_max_abs.

Accordingly confint(), profile(), logLik(), AIC, BIC, and anova() deliberately error for MSPL fits. When the information matrix is not positive definite — which happens on exactly the strongly separated designs MSPL exists to handle — the standard errors are NA and a drmTMB_mspl_wald_unavailable warning is signalled, rather than a fabricated number.

Examples

```
set.seed(20260524)
n <- 36
x <- seq(-1.5, 1.5, length.out = n)
dat <- data.frame(
  y = 0.3 + 0.6 * x + rnorm(n, sd = 0.7),
  x = x
)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat)

logLik(fit)
nobs(fit)
df.residual(fit)
deviance(fit)
AIC(fit)
BIC(fit)
vcov(fit)
```

vcov.drmTMB_julia_xfam

Extractor unavailable for a legacy cross-family Julia fit

Description

The cross-family Julia bridge is retained only to inspect legacy fitted objects. It does not marshal a named coefficient covariance matrix, so covariance-based fixed-effect inference cannot be reconstructed safely.

Usage

```
## S3 method for class 'drmTMB_julia_xfam'
vcov(object, ...)
```

Arguments

object	A drmTMB_julia_xfam cross-family fit.
...	Unused.

Value

This method always errors with an explanation.

vcov.drm_pair_association

Alpha-scale covariance for a frozen-margin association

Description

Returns the association-coefficient block of the two-stage Godambe covariance for an admitted fixed-effect complete-pair association. Standard errors are sqrt(diag(vcov(object))). The covariance and standard errors are on the unbounded association-link (alpha) scale, not the bounded latent association (eta) scale. The method warns when coverage is uncalibrated or the fit lies outside the retained coverage domain, and errors rather than manufacturing a covariance when fit-specific diagnostics fail.

Usage

```
## S3 method for class 'drm_pair_association'
vcov(object, ...)
```

Arguments

object	A fitted drm_pair_association object.
...	Reserved for future options.

Value

A named symmetric covariance matrix for the alpha coefficients.

See Also

confint.drm_pair_association()

`weights.drmTMB` *Extract likelihood weights*

Description

`weights()` returns the row likelihood multipliers used by a fitted `drmTMB` model after model-row filtering. These weights multiply log-likelihood contributions. They are not known sampling variances or known sampling covariance; use `meta_V()` for that meta-analytic role.

Usage

```
## S3 method for class 'drmTMB'
weights(object, ...)
```

Arguments

`object`	A `drmTMB` fit.
`...`	Reserved for future extractor options.

Value

A numeric vector with one weight per modelled response row, or per complete response pair for bivariate Gaussian, bivariate lognormal, and bivariate Student-t models. The two exact-special development families currently expose implicit unit weights only.

Examples

```
dat <- data.frame(
  y = c(0.2, 0.5, 1.1, 1.4),
  x = c(-1, 0, 1, 2),
  w = c(1, 1, 0.5, 2)
)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), data = dat, weights = w)
weights(fit)
```

`worm_plot` *Worm plot of randomized quantile residuals*

Description

`worm_plot()` draws a detrended QQ plot (van Buuren & Fredriks-style worm plot) of `drm_quantile_residuals()` against their N(0,1) order-statistic theoretical quantiles: `deviation = sorted residual - theoretical quantile`. A flat scatter around the dotted zero reference line is no detectable departure from N(0,1); a systematic bend flags a mis-specification of the fitted distributional form (see the GAMLSS-Primer Fig-4c contrast: a location-only fit to heteroscedastic data bends, the matching location-scale fit is flat).

Usage

```
worm_plot(object, seed = NULL, nsim = 1L, response = NULL, ...)
```

Arguments

`object`	A `drmTMB` fit.
`seed`	Optional single integer seed, passed to `drm_quantile_residuals()`.
`nsim`	Number of Dunn-Smyth realizations to overplot as an envelope; passed to `drm_quantile_residuals()`.
`response`	For a bivariate `biv_gaussian` fit, `1` or `2`, selecting which response's marginal residuals to plot; see `fitted_distribution()`. Must be `NULL` (the default) for univariate model types.
`...`	Reserved for future options.

Details

When `nsim` > 1, a pale grey envelope (with a darker outline) overplots the per-rank range across the `nsim` Dunn-Smyth realizations, so a single randomized draw is not over-read; the first realization's points and fitted trend are drawn on top.

This is fixed-effect adequacy only – see `drm_quantile_residuals()`. A flat worm plot is "no detectable departure" evidence about the fixed-effect distributional form, never a general validity or calibration claim.

What this detects – and does not. A worm plot bends when the underlying quantile residuals depart from N(0,1); see `residuals.drmTMB()`'s Details for the full gated-campaign breakdown (400 seeds x 18 families, tweedie: 99 of 400 seeds locally, full run deferred to Totoro; `docs/dev-log/simulation-artifacts/2026-07-12-dg3-power-arm-gated/`). In short: it flags shape/atom mis-specification a family cannot reabsorb through its own free parameters (heavy tails, ignored overdispersion or zero-inflation in a no-free-dispersion family, ignored truncation, a missing zero/one atom – gated power >= 0.8), but it stays flat – a genuine structural blind spot, not evidence of adequacy – when a free nuisance/dispersion/inflation parameter absorbs the mis-specification (e.g. heteroscedasticity absorbed by Student-t `nu`, missing zero-inflation absorbed by `nbinom2 sigma`, or a constant-vs-covariate zero-inflation/hurdle mechanism mis-set for `hurdle_nbinom2`/`zero_one_beta`). For `zi_poisson`/`zi_nbinom2`, the same constant-vs-covariate mechanism mis-spec is NOT flat – power rises with n – but stays far below 0.8 even at n = 3000, so it is sample-size-limited in principle but impractical to detect at realistic sample sizes. A mean-structure diagnostic, not this one, is what catches an absorbed mis-specification.

Value

A ggplot object.

Examples

```
set.seed(20260712)
n <- 60
x <- stats::rnorm(n)
dat <- data.frame(y = 0.5 + 0.8 * x + stats::rnorm(n), x = x)
fit <- drmTMB(bf(y ~ x, sigma ~ 1), family = gaussian(), data = dat)
if (requireNamespace("ggplot2", quietly = TRUE)) {
  worm_plot(fit)
}
```

---

zero_one_beta *Zero-one beta response family*

Description

`zero_one_beta()` defines a one-response distribution for continuous proportions on `[0, 1]` when exact zeroes or ones are structural outcomes rather than binomial denominator outcomes.

Usage

```
zero_one_beta()
```

Details

The implemented fixed-effect contract is `logit(mu) = eta_mu`, `log(sigma) = eta_sigma`, `logit(zoi) = eta_zoi`, and `logit(coi) = eta_coi`. Here `zoi` is the probability that an observation is exactly 0 or 1, and `coi` is the conditional probability of an exact 1 given that the observation is on the boundary. Interior observations follow the same beta mean-scale contract as `beta()`, with internal precision `phi = 1 / sigma^2`.

`fitted()` returns the unconditional response mean `(1 - zoi) * mu + zoi * coi`. Ordinary unlabelled random intercepts and independent numeric slopes such as `(1 | id)` and `(0 + x | id)` may enter `mu`. The point-fit-only q1 gates additionally admit one ordinary intercept-only random effect in `sigma`, `zoi`, or `coi`, separately, plus one ordinary slope-only effect in any one of those parameters when the fixed and random terms use the same raw symbol. Examples are `zoi ~ x + (0 + x | id)` and `coi ~ x + (0 + x | id)`. The exact `coi` admissions are that slope form and `coi ~ 1 + (1 | id)`, both with fixed `sigma ~ 1` and `zoi ~ 1`. All of these routes carry point-fit recovery evidence only. The two `sigma` routes now admit direct profiling, so `confint(method = "profile")` reaches their SD target; that makes an interval computable, not calibrated, and no interval, coverage, or broader recovery claim is established for them. For the `zoi` and `coi` routes, direct profiling, intervals, coverage, and broader recovery claims remain unavailable. In the retained `coi` recovery rung (`M = 64`, 50 observations per group), population-level point recovery remained stable when a few groups had fewer than two observed zeroes or ones, but those groups can have weakly identified conditional modes; inspect the within-group atom counts before interpreting individual modes. Correlated or

labelled effects, intercept-plus-slope atom effects, transformed or mismatched `zoi` or `coi` slope symbols, structured atom effects, covariance blocks, and denominator syntax remain unsupported. One further point-fit-only structured route is available: an unlabelled `q1` `phylo(1 | group, tree = tree)` intercept in `mu`. It does not license other providers, slopes, labels, covariance, q2-plus structured effects, profiling, intervals, coverage, or inference claims.

Value

A `drm_family` object.

Examples

```
zero_one_beta()
```

Index