

Package: gllvmTMB (via r-universe)

October 28, 2026

Type Package

Title Generalised Linear Latent Variable Models

Version 0.7.1

Description Fits multivariate generalised linear latent variable models using either long-format data or a wide data frame. Models support several response distributions and trait covariance structures, and can estimate correlations, shared latent axes, and trait-specific variances. Binary presence-absence data can be used for joint species distribution analyses. Optional animal, phylogenetic, spatial, and kernel covariance components are available. Estimation uses 'Template Model Builder'. Spatial effects can use sparse Gaussian Markov random field constructions. This 0.7.1 release remains experimental: evidence for estimates is model- and regime-specific. Broad interval coverage is not certified; a 0.94 coverage floor is documented only for one two-sided Gaussian total-variance profile setting.

License GPL-3

Language en-GB

Copyright inst/COPYRIGHTS

URL <https://itchyshin.github.io/gllvmTMB/>,
<https://github.com/itchyshin/gllvmTMB>

BugReports <https://github.com/itchyshin/gllvmTMB/issues>

Depends R (>= 4.1.0)

Imports assertthat, cli, fmesher, generics, graphics, lifecycle,
Matrix, methods, rlang, stats, tidyselect, TMB (>= 1.8.0),
utils

LinkingTo BH, RcppEigen, TMB

Suggests ape, DHARMA, detectseparation (>= 0.4.0), dplyr, future,
future.apply, ggforce, ggplot2, galamm, glmmTMB, gllvm,
JuliaCall, jsonlite, knitr, lme4, MASS, MCMCglmm, mgcv, mirt,
mvtnorm, nadiv, rmarkdown, sf, testthat (>= 3.0.0), tibble,
tidyr, tweedie, vegan, vdiff, withr

VignetteBuilder knitr
ByteCompile true
Config/testthat/edition 3
Config/testthat/parallel true
Encoding UTF-8
SystemRequirements GNU make, C++17
Config/roxygen2/version 8.0.0
NeedsCompilation yes
Author Shinichi Nakagawa [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0002-7765-5182>)
Maintainer Shinichi Nakagawa <itchyshin@gmail.com>
Config/pak/sysreqs make
Repository https://cran.r-universe.dev
Date/Publication 2026-10-28 21:45:14 UTC
RemoteUrl https://github.com/cran/gllvmTMB
RemoteRef HEAD
RemoteSha ca1db6d43584ae5bd7c51096aa21793b363c37f4

Contents

add_utm_columns	5
animal_dep	6
animal_indep	7
animal_latent	8
animal_scalar	9
animal_slope	11
animal_unique	12
block_V	14
bootstrap_Sigma	15
categorical	18
check_auto_residual	19
check_gllvmTMB	19
compare_dep_vs_two_psi	23
compare_indep_vs_two_psi	26
compare_loadings	28
compare_Sigma_table	28
confint.gllvmTMB_multi	30
confint_inspect	34
confirmatory_lambda	36
cumulative_logit	38
dep	38
deviance.gllvmTMB_multi	40
diagnose_kernel_separability	40

diagnostic_table	41
extract_coevolution_modules	43
extract_communality	44
extract_correlations	46
extract_cross_correlations	49
extract_cutpoints	51
extract_Gamma	53
extract_loadings	54
extract_lv_effects	55
extract_Omega	56
extract_ordination	57
extract_phylo_signal	59
extract_proportions	61
extract_repeatability	63
extract_residual_cov	65
extract_rotated_loadings_table	65
extract_Sigma	67
extract_Sigma_table	71
fitted.gllvmTMB_multi	73
flag_unreliable_loadings	74
get_crs	75
gllvm_julia_fit	76
gllvm_julia_setup	78
gllvmTMB	79
gllvmTMB_check_consistency	89
gllvmTMB_diagnose	91
gllvmTMB_julia-methods	93
gllvmTMB_multi-methods	95
gllvmTMB_multi-vcov	97
gllvmTMB_va-methods	98
gllvmTMBcontrol	100
impute_model	106
imputed	107
indep	108
isdmsource	110
isdmsources	110
kernel_latent	112
kernel_scalar	113
kernel_unique	114
latent	114
loading_ci	116
loading_profile	118
make_cross_kernel	119
make_mesh	120
meta_V	122
miss_control	123
multinomial	124
ordinal_probit	126

pedigree_to_A	128
pedigree_to_Ainv_sparse	129
phylo	130
phylo_dep	133
phylo_indep	134
phylo_latent	136
phylo_scalar	139
phylo_signal_mi	140
phylo_slope	141
phylo_unique	142
plot.gllvmTMB_multi	145
plot.gllvmTMBmesh	147
plot.profile_derived	148
plot.profile_loadings	148
plot_anisotropy	149
plot_correlations	150
plot_loadings_confidence_eye	152
plot_rotated_loadings	154
plot_Sigma_comparison	156
plot_Sigma_heatmap	157
plot_Sigma_table	159
predict.gllvmTMB_multi	161
predict_cross_covariance	163
predict_missing	164
predictive_check	167
print.gllvmTMB_Sigma_phy_slope	168
profile_ci_total_variance	169
profile_cross_rho	170
profile_cross_rho_ci	172
profile_phylo_signal	172
profile_targets	173
residuals.gllvmTMB_multi	175
ridge_path	177
rotate_loadings	179
sanity_multi	181
scalar	182
screen_control	183
screen_gllvmTMB	184
screen_table	187
simulate.gllvmTMB_multi	188
simulate_site_trait	189
simulate_unit_trait	191
slope_sd_ci	193
spatial	196
spatial_dep	199
spatial_indep	200
spatial_latent	202
spatial_scalar	203

spatial_unique	205
standard_errors	206
suggest_lambda_constraint	208
suggest_lambda_constraints	210
tidy.gllvmTMB_multi	212
traits	213

Index	215
--------------	------------

add_utm_columns	<i>Add UTM coordinates to a data frame</i>
-----------------	--

Description

Transform longitude/latitude columns to an equal-distance UTM coordinate reference system with [sf::st_transform\(\)](#).

Usage

```
add_utm_columns(  
  dat,  
  ll_names = c("longitude", "latitude"),  
  ll_crs = 4326,  
  utm_names = c("X", "Y"),  
  utm_crs = get_crs(dat, ll_names),  
  units = c("km", "m")  
)
```

Arguments

dat	Data frame containing longitude and latitude columns.
ll_names	Character names of longitude then latitude columns.
ll_crs	Input coordinate reference system.
utm_names	Names for the two new coordinate columns.
utm_crs	Output UTM CRS; by default inferred with get_crs() .
units	Output coordinate units: kilometres or metres.

Value

A copy of dat with the requested UTM coordinate columns.

Examples

```
d <- data.frame(lat = c(52.1, 53.4), lon = c(-130, -131.4))  
add_utm_columns(d, c("lon", "lat"))
```

animal_dep	<i>Full unstructured animal-model trait covariance:</i> animal_dep(0 + trait id)
------------	--

Description

Full-rank unstructured $T \times T$ additive-genetic covariance matrix G , parameterised as the Cholesky factor of a rank- T Λ (i.e. the [animal_latent\(\)](#) case with $K = T$). Mathematical parallel to [phylo_dep\(\)](#).

Usage

```
animal_dep(formula, pedigree = NULL, A = NULL, Ainv = NULL)
```

Arguments

formula An lme4-bar formula of the form 0 + trait | id.
 pedigree, A, Ainv See [animal_scalar\(\)](#).

Value

See [animal_scalar\(\)](#).

See Also

[animal_latent\(\)](#), [phylo_dep\(\)](#).

Examples

```
# Full unstructured additive-genetic trait covariance (rank = n_traits).
# Grounded in test-animal-keyword.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
yvec <- as.numeric(MASS::mvrnorm(
  1, mu = rep(0, 2 * 12),
  Sigma = kronecker(diag(2), A) * 0.5 + diag(2 * 12) * 0.5
))
df <- data.frame(
  species = factor(rep(ped$id, each = 2), levels = ped$id),
  trait   = factor(rep(c("t1", "t2"), times = 12), levels = c("t1", "t2")),
  value   = yvec
)
fit <- gllvmTMB(
  value ~ 0 + trait + animal_dep(0 + trait | species, A = A),
```

```
data = df, family = gaussian(), unit = "species", cluster = "species"
)
```

animal_indep	<i>Independent per-trait animal-model random intercepts:</i> animal_indep(0 + trait id)
--------------	--

Description

Per-trait animal-model random intercepts with no cross-trait covariance, using the bar-form syntax; the .indep marker disambiguates printing. Mathematical parallel to [phylo_indep\(\)](#).

Usage

```
animal_indep(formula, pedigree = NULL, A = NULL, Ainv = NULL, common = FALSE)
```

Arguments

- formula An lme4-bar formula of the form 0 + trait | id.
- pedigree, A, Ainv See [animal_scalar\(\)](#).
- common FALSE (default) for a separate additive-genetic variance per trait; TRUE ties all traits to one shared additive-genetic variance (intercept-only). animal_indep(0 + trait | id, common = TRUE) is the canonical one-shared-variance spelling and fits the same model as the soft-deprecated animal_scalar(id).

Value

See [animal_scalar\(\)](#).

See Also

[phylo_indep\(\)](#).

Examples

```
# Independent per-trait animal-model intercepts via the bar form,
# passing the dense relatedness matrix A directly.
# Grounded in test-animal-keyword.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
yvec <- as.numeric(MASS::mvrnorm(
  1, mu = rep(0, 2 * 12),
  Sigma = kronecker(diag(2), A) * 0.5 + diag(2 * 12) * 0.5
```

```

))
df <- data.frame(
  species = factor(rep(ped$id, each = 2), levels = ped$id),
  trait   = factor(rep(c("t1", "t2"), times = 12), levels = c("t1", "t2")),
  value   = yvec
)
fit <- gllvmTMB(
  value ~ 0 + trait + animal_indep(0 + trait | species, A = A),
  data = df, family = gaussian(), unit = "species", cluster = "species"
)

```

animal_latent	<i>Reduced-rank animal-model latent factors:</i> animal_latent(id, d = K)
---------------	---

Description

Reduced-rank decomposition of the additive-genetic covariance matrix using K latent factors. Set `unique = TRUE` to add a per-trait diagonal Ψ companion: $\mathbf{G} = \mathbf{\Lambda}\mathbf{\Lambda}^\top + \mathbf{\Psi}$ with $\mathbf{\Lambda}$ a $T \times K$ loadings matrix (T = number of traits, $K \leq T$). The latent factors and diagonal companion both carry the same $\mathbf{A}$ structure across individuals.

Usage

```

animal_latent(
  id,
  d = 1,
  pedigree = NULL,
  A = NULL,
  Ainv = NULL,
  unique = FALSE
)

```

Arguments

id	Bare column name of the individual factor.
d	Number of latent factors ($K \leq T$). Default 1.
pedigree	A 3-column data frame with columns id, sire, dam (unknown parents encoded as NA). Converted internally to $\mathbf{A}$ via Henderson's recursive formula. Only one of pedigree, A, or Ainv should be given.
A	Dense relatedness matrix ($n \times n$); rownames / colnames must match levels of id.
Ainv	Precision matrix (inverse of A). Sparse matrix inputs are preserved for the sparse engine route.
unique	Logical; when TRUE, include the per-trait diagonal additive-genetic Ψ companion. The default FALSE preserves the loadings-only subset.

Details

This is the canonical "factor-analytic G-matrix" model from quantitative genetics (Kirkpatrick & Meyer 2004; Meyer 2009; the WOMBAT method). Mathematical parallel to `phylo_latent()` – same engine path with a pedigree-derived relatedness matrix instead of phylogenetic VCV.

Value

See `animal_scalar()`.

See Also

`animal_scalar()`, `animal_indep()`, `phylo_latent()`.

Examples

```
# Factor-analytic G-matrix (d latent factors plus diagonal Psi).
# Grounded in test-animal-keyword.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
yvec <- as.numeric(MASS::mvrnorm(
  1, mu = rep(0, 2 * 12),
  Sigma = kronecker(diag(2), A) * 0.5 + diag(2 * 12) * 0.5
))
df <- data.frame(
  species = factor(rep(ped$id, each = 2), levels = ped$id),
  trait    = factor(rep(c("t1", "t2"), times = 12), levels = c("t1", "t2")),
  value    = yvec
)
fit <- gllvmTMB(
  value ~ 0 + trait + animal_latent(species, d = 1, pedigree = ped,
                                     unique = TRUE),
  data = df, family = gaussian(), unit = "species", cluster = "species"
)
```

animal_scalar	Single-shared-variance animal_scalar(id)	animal-model	random	effect:
---------------	---	--------------	--------	---------

Description

Canonical name for the single-scalar (per-trait) animal-model random intercept. Each trait carries an independent length-`n_ids` draw $\mathbf{p}_t \sim \mathcal{N}(\mathbf{0}, \sigma_a^2 \mathbf{A})$ where $\mathbf{A}$ is the **additive genetic relatedness matrix** (the pedigree-derived numerator-relationship matrix). The variance σ_a^2 is shared across traits.

Usage

```
animal_scalar(id, pedigree = NULL, A = NULL, Ainv = NULL)
```

Arguments

<code>id</code>	Bare column name of the individual factor.
<code>pedigree</code>	A 3-column data frame with columns <code>id</code> , <code>sire</code> , <code>dam</code> (unknown parents encoded as NA). Converted internally to A via Henderson's recursive formula. Only one of <code>pedigree</code> , A , or Ainv should be given.
A	Dense relatedness matrix ($n \times n$); rownames / colnames must match levels of <code>id</code> .
Ainv	Precision matrix (inverse of A). Sparse matrix inputs are preserved for the sparse engine route.

Details

Mathematical parallel to [phylo_scalar\(\)](#) – same engine path; the only difference is that **A** is supplied via `pedigree` = (a 3-column data frame: `id`, `sire`, `dam`), **A** = (dense $n \times n$ relatedness matrix), or **Ainv** = (precision matrix). Sparse **Ainv** inputs use the sparse precision route.

Value

Used inside a [gllvmTMB\(\)](#) formula. Returns `invisible(NULL)` when called outside a formula – the keyword is a syntactic marker that the parser rewrites internally to the canonical [phylo_scalar\(\)](#) path (which is family-agnostic at the math level).

See Also

[animal_indep\(\)](#), [animal_latent\(\)](#), [animal_dep\(\)](#), [animal_slope\(\)](#), [phylo_scalar\(\)](#), [meta_V\(\)](#) (sampling variance, distinct from relatedness).

Examples

```
# Pedigree-derived additive-genetic relatedness; single shared
# variance across traits. Grounded in test-animal-keyword.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
yvec <- as.numeric(MASS::mvrnorm(
  1, mu = rep(0, 2 * 12),
  Sigma = kronecker(diag(2), A) * 0.5 + diag(2 * 12) * 0.5
))
df <- data.frame(
  species = factor(rep(ped$id, each = 2), levels = ped$id),
  trait   = factor(rep(c("t1", "t2"), times = 12), levels = c("t1", "t2")),
  value   = yvec
)
```

```
fit <- glvmTMB(
  value ~ 0 + trait + animal_scalar(species, pedigree = ped),
  data = df, family = gaussian(), unit = "species", cluster = "species"
)
```

animal_slope

Animal-model random slope on a continuous covariate

Description

Adds a per-individual random slope on a continuous covariate x , with individual-level slopes correlated according to the additive- genetic relatedness matrix $\mathbf{A}$. Mathematically:

$$\eta_{io} += \beta_a(i) x_{io}, \quad \beta_a \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{slope}}^2 \mathbf{A}).$$

Usage

```
animal_slope(formula, pedigree = NULL, A = NULL, Ainv = NULL)
```

Arguments

formula	An lme4-bar formula. $x \mid id$ retains the established one-predictor shared-slope helper. With two or more bare predictors and the response-column factor on the RHS, $x_1 + x_2 \mid trait$ is the full column-slope covariance alias and $x_1 + x_2 \mid \mid trait$ is its diagonal alias.
pedigree, A, Ainv	One relatedness input for the column-slope aliases: a pedigree, additive-genetic covariance matrix, or its precision.

Details

This is the canonical **quantitative-genetic random-regression** (reaction-norm) model – heritable variation in the slope on an environmental gradient. Mathematical parallel to [phylo_slope\(\)](#); the engine recycles the same $\mathbf{A}^{-1}$ prepared by [animal_scalar\(\)](#) / [animal_latent\(\)](#).

Value

See [animal_scalar\(\)](#).

See Also

[animal_scalar\(\)](#), [animal_latent\(\)](#), [phylo_slope\(\)](#).

Examples

```
# Additive-genetic random regression: one heritable slope variance on a
# continuous covariate x, slopes correlated by the relatedness matrix A.
# Grounded in test-animal-slope-recovery.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
df <- expand.grid(
  species = factor(rownames(A), levels = rownames(A)),
  trait   = factor(c("t1", "t2", "t3")),
  rep     = 1:5
)
df$x <- rnorm(nrow(df))
df$value <- rnorm(nrow(df))
fit <- glvmTMB(
  value ~ 0 + trait + animal_slope(x | species, A = A),
  data = df, unit = "species", cluster = "species"
)
```

animal_unique	<i>Per-trait independent</i>	<i>animal-model</i>	<i>random</i>	<i>intercepts:</i>
	animal_unique(id)			

Description

[Deprecated]

Usage

```
animal_unique(id, pedigree = NULL, A = NULL, Ainv = NULL)
```

Arguments

id	Bare column name of the individual factor.
pedigree	A 3-column data frame with columns id, sire, dam (unknown parents encoded as NA). Converted internally to A via Henderson's recursive formula. Only one of pedigree, A, or Ainv should be given.
A	Dense relatedness matrix ($n \times n$); rownames / colnames must match levels of id.
Ainv	Precision matrix (inverse of A). Sparse matrix inputs are preserved for the sparse engine route.

Details

`animal_unique()` is soft-deprecated as compatibility syntax in gllvmTMB 0.2.0. Use [animal_indep\(\)](#) for standalone marginal diagonal animal-model terms. Paired explicit-Psi use remains accepted; use `animal_latent(..., unique = TRUE)` when the folded latent term itself should carry this diagonal companion.

Canonical name for the **D independent** animal-model random intercept on a shared relatedness matrix **A**. Each trait gets its own variance $\sigma_{a,t}^2$; traits share the same per-individual draws (rows of **A** act on each trait independently). Mathematical parallel to [phylo_unique\(\)](#).

Value

See [animal_scalar\(\)](#).

Correlated intercept + slope reaction norm

Used with an intercept+slope bar, `animal_unique(1 + x | id, pedigree = ped)` fits a **correlated** additive-genetic random-regression (reaction norm) $\text{vec}(\mathbf{B}) \sim N(0, \Sigma_b \otimes \mathbf{A})$, where Σ_b is a 2×2 (intercept, slope) covariance with a **free** intercept-slope correlation. This routes through the same augmented engine as [phylo_unique\(\)](#) with a `1 + x | sp` bar; no separate likelihood. For the diagonal ($\rho = 0$) special case use [animal_indep\(\)](#) with the same `1 + x | id` bar. Recover Σ_b with `extract_Sigma(fit, level = "phy")`.

See Also

[animal_scalar\(\)](#), [animal_indep\(\)](#), [animal_latent\(\)](#), [animal_dep\(\)](#), [phylo_indep\(\)](#).

Examples

```
# Per-trait independent additive-genetic variances on a shared A.
# Grounded in test-animal-keyword.R.
ped <- data.frame(
  id   = paste0("i", 1:12),
  sire = c(rep(NA, 4), rep(c("i1", "i2"), length.out = 8)),
  dam  = c(rep(NA, 4), rep(c("i3", "i4"), length.out = 8))
)
A <- pedigree_to_A(ped)
yvec <- as.numeric(MASS::mvrnorm(
  1, mu = rep(0, 2 * 12),
  Sigma = kronecker(diag(2), A) * 0.5 + diag(2 * 12) * 0.5
))
df <- data.frame(
  species = factor(rep(ped$id, each = 2), levels = ped$id),
  trait    = factor(rep(c("t1", "t2"), times = 12), levels = c("t1", "t2")),
  value    = yvec
)
fit <- gllvmTMB(
  value ~ 0 + trait + animal_unique(species, pedigree = ped),
  data = df, family = gaussian(), unit = "species", cluster = "species"
)
```

block_V

*Build a block-diagonal sampling-variance matrix V***Description**

Constructs a block-diagonal $n \times n$ matrix V suitable for the `meta_V()` / `equalto()` covstruct in `gllvmTMB()`, where rows belonging to the same `study_id` are correlated and rows from different studies are independent. Each block is a compound-symmetric covariance matrix $V_s = D_s^{1/2} R_s D_s^{1/2}$ with $D_s = \text{diag}(\sigma_{s1}^2, \dots)$ the per-row sampling variances within study s and R_s the within-study correlation matrix (1 on the diagonal, `rho_within` on the off-diagonal).

Usage

```
block_V(study_id, sampling_var, rho_within = 0.5)
```

Arguments

<code>study_id</code>	A factor (or coercible to one) identifying which study each row belongs to. Rows with the same <code>study_id</code> form a block in V .
<code>sampling_var</code>	Numeric vector of per-row sampling variances, length <code>length(study_id)</code> .
<code>rho_within</code>	Either a single numeric in $(-1, 1)$ used for every study's off-diagonals, or a named numeric vector with one entry per <code>study_id</code> level (allowing study-specific within-study correlations). Default 0.5 - a common ad-hoc choice in ecology meta-analyses when the true ρ is unknown but plausibly moderate.

Details

Use this when you have multiple effect sizes per study (multiple outcomes, multiple traits per individual, etc.) and the within-study sampling errors share a common cause. The independent-rows / single-effect-per-study case is `block_V(..., rho_within = 0)`, which equals `diag(sampling_var)`.

Value

A symmetric positive-definite $n \times n$ matrix.

See Also

`meta_V()` for the canonical keyword that consumes the matrix produced here.

Examples

```
set.seed(1)
df <- data.frame(
  study = factor(rep(paste0("s", 1:3), each = 2)),
  var = c(0.04, 0.05, 0.06, 0.04, 0.05, 0.07)
)
V <- block_V(df$study, df$var, rho_within = 0.5)
round(V, 3)
```

bootstrap_Sigma

Bootstrap covariance, correlation, communality, and ICC summaries

Description

Use `bootstrap_Sigma()` when Hessian, Wald, or profile intervals are unavailable or unsafe but the fitted model still has useful point estimates. A `pdHess = FALSE` or skipped `sdreport()` is an inference warning, not automatic proof that the fitted mean or rotation-invariant covariance summaries are unusable. Inspect `check_gllvmTMB()` / `gllvmTMB_diagnose()` first, then use this helper for parametric simulate-refit uncertainty.

Usage

```
bootstrap_Sigma(
  fit,
  n_boot = 999,
  level = c("unit", "unit_obs", "phy", "B", "W"),
  what = c("Sigma", "R", "communality", "ICC", "cross_corr"),
  conf = 0.95,
  seed = NULL,
  n_cores = 1,
  progress = TRUE,
  keep_draws = FALSE,
  link_residual = c("auto", "none")
)
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> .
<code>n_boot</code>	Integer; number of bootstrap replicates. Default 999. Values below $2 / (1 - \text{conf}) - 1$ (39 at the default $\text{conf} = 0.95$) are refused: the widest interval B draws can produce is $[\min, \max]$, whose coverage is at most $(B - 1) / (B + 1)$, so a smaller B cannot reach the requested level whatever the data are. Values below 999 warn. Two different things bound <code>n_boot</code>, and they bite at different scales. Below the refusal threshold the <i>arithmetic ceiling</i> binds: the interval cannot reach the requested level however much data you have. Above it, what binds is Monte Carlo error in the endpoints, which is a precision question and shrinks with B. Conflating the two is how a replicate count comes to be treated as a free budget lever when the low end of it is a correctness constraint. The default is 999 rather than 1000 so that $(1 - \text{conf}) / 2 * (B + 1)$ is a whole number at $\text{conf} = 0.95$ (25), letting the percentile bounds land on order statistics rather than being interpolated between them. Raising <code>n_boot</code> reduces Monte Carlo error in the endpoints but does not make a percentile interval second-order accurate; its coverage asymptotes slightly below nominal regardless. Use <code>profile_ci_total_variance()</code> for the Sigma

	diagonals if that matters. For exploratory work a smaller B (200, say) is a reasonable time-for-precision trade; for a number you intend to publish, do not go below the default.
level	Character vector; which tier(s) to bootstrap. Use the canonical levels <code>c("unit", "unit_obs", "phy")</code> ; legacy aliases "B" and "W" are still accepted. Levels absent from the fit are silently dropped. Default: all available levels.
what	Character vector; which summaries to compute. Subset of <code>c("Sigma", "R", "communality", "ICC", "cross_corr")</code> . Default: all. "ICC" only makes sense at the site level and requires both B and W tiers in the fit. "cross_corr" bootstraps the aggregate <code>multiple_r</code> between a <code>multinomial()</code> trait and each partner (see extract_cross_correlations()); it is stored as a plain named numeric per tier (<code>multiple_r_B</code> , ...) and is silently skipped for fits without a nominal trait.
conf	Numeric in $(0, 1)$; confidence level for percentile CIs. Default 0.95.
seed	Optional RNG seed for reproducibility.
n_cores	Integer; number of workers for refits. Must be 1 or 2. Default 1 (sequential); 2 uses <code>future::multisession</code> .
progress	Logical; print a one-line status message at each replicate (sequential only). Default TRUE.
keep_draws	Logical; if TRUE, the full <code>n_boot</code> x ... matrices of bootstrap draws are returned as <code>\$draws</code> . Default FALSE (CIs only; saves memory for large <code>n_boot</code>).
link_residual	How to treat family-specific link-implicit residual variance when extracting Sigma, R, communality, and ICC. "auto" (default) matches extract_Sigma() and adds the family/link residual to non-Gaussian trait diagonals; "none" returns the fitted model covariance without link residuals. Use "none" when validating against a DGP target defined as $\Lambda\Lambda^\top + \Psi$.

Details

The function generates `n_boot` parametric bootstrap replicates of a fitted model and returns percentile confidence intervals for the canonical biological summaries: trait covariance matrices $\hat{\Sigma}_{\text{unit}}$, $\hat{\Sigma}_{\text{unit_obs}}$; the corresponding correlation matrices; per-trait communalities $c_t^2 = (\Lambda\Lambda^\top)_{tt}/\Sigma_{tt}$; and per-trait site-level ICCs $R_t = (\Sigma_{\text{unit}})_{tt}/[(\Sigma_{\text{unit}})_{tt} + (\Sigma_{\text{unit_obs}})_{tt}]$.

Scope: Gaussian bootstrap summaries and mixed-family refit plumbing are covered by current tests. Non-Gaussian bootstrap calibration is experimental: its repeated-sampling behaviour has not been certified, and there is no production calibration evidence for mixed-family intervals. Treat those intervals as indicative.

Each bootstrap replicate (1) draws a new response vector from `simulate(fit, nsim = 1)`, (2) refits the model with the same formula on the simulated data, (3) extracts the requested summaries via [extract_Sigma\(\)](#), [extract_communality\(\)](#), and [extract_ICC_site\(\)](#). Replicates whose refit fails to converge are recorded but excluded from CI calculation.

Parallel refits use `future + future.apply` and are limited to two workers. Set `n_cores = 2` to enable parallel refits. When parallel, replicates use `future.apply`'s L'Ecuyer-CMRG seed stream so the answers are reproducible given a fixed seed, but they are NOT bit-identical to an `n_cores = 1` run with the same seed (different RNG streams).

Value

A list with components:

`point_est` Named list of point estimates for each requested summary at each requested level (e.g. `Sigma_B`, `R_B`, `communality_B`, `ICC_site`).

`ci_lower`, `ci_upper` Named lists of percentile CI bounds, element-wise the same shape as the corresponding `point_est`.

`n_effective` Named list (vector summaries only) giving the per-element count of finite bootstrap draws. For `multiple_r_*` entries, any element whose effective count is below 80% of `n_boot` has its CI bounds set to NA (minimum-effective-B floor).

`boot_median` Named list (`multiple_r_*` entries only) of the per-element bootstrap median – a cheap sanity comparator against `point_est`; a large gap flags gross bootstrap corruption and is not a coverage certifier.

`ci_method` Character; currently "percentile".

`link_residual` Character; the link-residual convention used in point estimates and bootstrap refit summaries.

`conf`, `n_boot`, `n_failed` Configuration metadata.

`draws` NULL unless `keep_draws = TRUE`; otherwise a named list of bootstrap draw arrays.

Caveats

- Uses the existing `simulate.gllvmTMB_multi()` method. By default that method REDRAWS the random effects from the fitted covariance, which is what makes these intervals span between-unit variability. CIs reflect parametric simulate-refit variability, not a Bayesian posterior distribution for variance components.
- **Intervals are too narrow when the simulator cannot redraw a tier.** Redraw is not implemented for every random-effect tier, notably the SPDE spatial tier and the diagonal phylogenetic tier. For a fit using one of those, `simulate.gllvmTMB_multi()` falls back to reusing the fitted random-effect modes and emits a one-shot warning. That fallback understates between-unit variability, so the intervals returned here are **not calibrated** for such fits. Treat the warning as a signal that these intervals cannot be trusted. Unsupported families fall back through the simulator's own warning path.
- Refits use the same formula reconstructed from `fit$formula` and `fit$covstructs`, and forward the fit's auxiliary structure (`phylo_vcv`, `phylo_tree`, `mesh`, `lambda_constraint`) so each refit matches the original model.
- Convergence: replicates whose refit fails or whose optimiser does not return `convergence == 0` are counted in `n_failed` and excluded from CIs.

Examples

```
set.seed(1)
s <- simulate_site_trait(n_sites = 30, n_species = 1, n_traits = 3,
                        mean_species_per_site = 1,
                        Lambda_B = matrix(c(1, .5, -.4), 3, 1),
                        psi_B = c(.2, .15, .1))
fit <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = 1),
```

```

      data = s$data,
      trait = "trait",
      unit = "site")
# Fifty refits illustrate the API; they do not establish interval coverage.
boot <- bootstrap_Sigma(fit, n_boot = 50, level = "unit",
  what = c("Sigma", "R"), seed = 42)

boot$point_est$Sigma_B
boot$ci_lower$Sigma_B
boot$ci_upper$Sigma_B

```

categorical	<i>Baseline-category softmax unordered categorical missing-predictor family</i>
-------------	---

Description

`categorical()` declares the predictor-model family for an **unordered categorical** missing predictor used inside `mi()` with `impute_model(family = categorical())`. The missing unordered predictor is marginalised EXACTLY by a finite-state baseline-category softmax sum over its K categories ($K \geq 3$); there is no latent variable. The predictor link is a baseline-category softmax with the FIRST level as baseline ($\eta_{\text{state}}(1) = 0$); for state $k = 2, \dots, K$, $\eta_{\text{state}}(k) = \mathbf{X}_x \beta_x[\text{block}(k-1)]$, and $\Pr(x = k \mid z) = \exp(\eta_{\text{state}}(k)) / \sum_j \exp(\eta_{\text{state}}(j))$. The coefficient vector β_x is packed as $K - 1$ blocks of the covariate design width. The missing predictor must be an **unordered factor**, **character**, or **integer category scores** with at least three levels, and every category must appear among the observed values; a two-level predictor uses `binomial()` (the binary route) and an ordered predictor uses `cumulative_logit()`.

Usage

```
categorical()
```

Details

This is a predictor-model family tag consumed by `impute_model(family = )`, NOT a response family for `gllvmTMB()`.

Value

A `gllvmTMB_impute_family` object for the family argument of `impute_model()`.

See Also

`impute_model()`, `cumulative_logit()` for an ordered missing predictor, and `binomial()` for a binary missing predictor.

Examples

```
impute_model(habitat ~ z, family = categorical())
```

check_auto_residual	<i>Check whether link_residual = "auto" is coherent for this fit</i>
---------------------	--

Description

Inspects the fit's per-row family vector and flags two configurations that make the link_residual = "auto" path incoherent: within-trait family mixing (errors) and ordinal-probit traits (warns; the latent residual is already standardised at 1, so adding 1 again over-counts).

Usage

```
check_auto_residual(fit)
```

Arguments

fit	A fit returned by gllvmTMB() .
-----	--

Value

Invisibly, a list with components status (one of "ok", "warn", or "err" – though the "err" path is unreachable from a successful return because `cli::cli_abort` halts) and messages (a character vector of human-readable findings; empty when status = "ok").

Side effects

- "err" path: `cli::cli_abort` is fired and the function does not return.
- "warn" path: `cli::cli_warn` is fired before the function returns the list invisibly.
- "ok" path: silent.

See Also

[extract_correlations\(\)](#), [extract_Sigma\(\)](#), [extract_Omega\(\)](#).

check_gllvmTMB	<i>Check convergence, Hessian, gradients, and interval readiness</i>
----------------	--

Description

Run `check_gllvmTMB()` right after fitting, before interpreting confidence intervals or covariance summaries. It returns a stable table of optimiser, gradient, Hessian, `sdreport()`, restart, boundary, latent-identifiability, and binomial prevalence/loading diagnostics. It is the machine-readable companion to [gllvmTMB_diagnose\(\)](#): use this in simulations, tests, and reports where parsing printed messages would be brittle.

Usage

```
check_gllvmTMB(
  object,
  gradient_thresh = 0.01,
  se_thresh = 100,
  weak_axis_thresh = 0.05,
  psi_thresh = 1e-04,
  psi_rel_thresh = 0.01,
  sigma_eps_thresh = 1e-04,
  cross_loading_thresh = 0.6,
  binary_prevalence_thresh = 0.9,
  binary_saturation_prob_thresh = 0.99,
  binary_saturation_share_thresh = 0.5,
  loading_relative_thresh = 8,
  loading_runaway_thresh = 25,
  loading_absolute_thresh = 8,
  multinomial_collapse_floor = 1e-10,
  multinomial_collapse_rel_thresh = Inf,
  multinomial_rail_thresh = 0.99,
  multinomial_range_collapse_thresh = 0.02,
  ordinal_loading_runaway_thresh = Inf,
  ordinal_loading_absolute_thresh = Inf
)
```

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>gradient_thresh</code>	Maximum allowed absolute gradient component. Default 0.01.
<code>se_thresh</code>	Threshold above which a fixed-effect standard error is flagged as weakly identified. Default 100.
<code>weak_axis_thresh</code>	Minimum acceptable share of shared loading energy for a fitted latent axis. Default 0.05.
<code>psi_thresh</code>	Threshold below which a fitted per-trait <code>psi</code> standard deviation is flagged as near zero. Default 0.0001.
<code>psi_rel_thresh</code>	Threshold on the ratio of the smallest to the largest fitted per-trait <code>psi</code> standard deviation, below which the smallest is flagged as collapsed relative to its siblings. This catches boundary-pinned (Heywood) components whose absolute standard deviation still clears <code>psi_thresh</code> : <code>psi</code> is estimated on the log scale, so a component at the boundary is an interior point of the transformed space and <code>pdHess</code> stays positive definite there. Needs at least two components. Default 0.01, raised from 0.001 on measured evidence: across 360 gaussian and Poisson fits with a deliberately over-specified latent rank, 58% drove a unique standard deviation below a tenth of its true value while reporting convergence = 0 (208 of 209) and <code>pdHess</code> = TRUE (190 of 209). At 0.001 the row reported 73.7% of those; at 0.01 it reports 96.2%. The false-positive rate is zero at both, measured

on 151 healthy fits and again on 359 healthy fits whose *true* unique variances differ by up to a factor of 1000 — the case that decides transport, since a small ratio is then correct rather than pathological. Looser thresholds do not transport: 0.1 reaches full sensitivity but flags 19% of those healthy heterogeneous fits.

`sigma_eps_thresh`

Threshold below which an estimated residual `sigma_eps` is flagged as near boundary. Default 0.0001.

`cross_loading_thresh`

Minimum median trait dominance on a single latent axis before a multi-axis loading matrix is treated as block-structured enough for direct interpretation. Default 0.6.

`binary_prevalence_thresh`

Prevalence at or beyond which a binomial trait is treated as near-constant. Default 0.9.

`binary_saturation_prob_thresh`

Response-scale fitted probability threshold for saturation in binomial traits. Default 0.99.

`binary_saturation_share_thresh`

Minimum share of saturated fitted probabilities before a binomial trait is flagged. Default 0.5.

`loading_relative_thresh`

Threshold for the largest trait loading relative to the typical fitted loading size. At this level the loading is only reported alongside an extreme prevalence, because a sparse but genuine loading structure reaches it on healthy fits. Default 8.

`loading_runaway_thresh`

Threshold for the largest trait loading, relative to the typical fitted loading size, at which the loading is reported on its own without requiring an extreme prevalence. This catches an improper solution (Heywood case) from quasi-complete separation, which runs a loading away while the trait's marginal prevalence stays unremarkable. Default 25, calibrated on 6,824 simulated **single-family binomial** fits: no healthy fit reached it (the largest was 12.1), while it reported 96.3% of fits whose implied covariance was wrong by a factor of five or more. Two limits on that calibration are worth knowing. It has not been measured on mixed-family fits, and it would not transport to another family on its own, because a sparse but genuine loading structure pushes the ratio much higher there. The typical loading size is therefore taken over the binomial traits alone, so that a trait from another family cannot set the scale this threshold is judged against.

`loading_absolute_thresh`

Threshold on the largest trait loading itself, on the link scale, at which it is reported regardless of the other traits. A ratio cannot see a loading matrix inflated as a whole, because scaling every loading leaves every ratio unchanged; this supplies the absolute reference the ratio lacks. It is meaningful because the latent scores are standard normal by identification, so a binomial loading is the trait's latent standard deviation in link units: a value of this size already implies a fitted probability indistinguishable from 0 or 1 across an ordinary swing of the axis. Default 8, raised from 6 (issue #1098) after the earlier calibration pool (3,944

simulated binomial fits, no healthy fit above 3.99) turned out to be unrepresentative: its true loading scale never reached the regime where this arm misfires. A second pool built specifically to cross σ_{λ} in $c(0.7, 3.0)$ (928 healthy / 272 degenerate binomial-probit fits) – 3.0 chosen to hit issue #847’s aghq_ridge ridge-failure regime, not argued for realism – measured this arm as the SOLE source of every false positive found (232/928 at threshold 6, all attributable to this arm alone). Raising the threshold to 8 lowers the false-positive rate on that pool from 0.2500 to 0.1552 while sensitivity on its degenerate fits falls only from 1.0000 to 0.9963 (one additional missed fit out of 272). This is an interim improvement, not a fix. The arm is regime/effect-size dependent, not response-scale dependent (probit fixes the residual variance at 1, so there is no free response scale here for a $\tau * \text{sd}(y)$ -style rescale to absorb); false-positive rate 3.85% at a mild true loading scale of $\sigma_{\lambda} = 0.7$ versus 49.08% at $\sigma_{\lambda} = 3.0$, so no fixed constant is correct across loading scales, and this default should not be read as calibrated against that regime. aghq_ridge = 2 reduces but does not remove the problem (46.0% $\rightarrow$ 13.5% false positives at $\sigma_{\lambda} = 3.0$). Being a link-scale quantity it does not transport to families whose response scale is arbitrary, which is why this row is binomial-only. This gate applies to every link ($\text{family_id} == 1L$), but calibration is probit-only. No logit calibration is available, so do not interpret the probit results or thresholds as calibrated for logit fits.

multinomial_collapse_floor

Absolute floor on a multinomial() (fid 16) contrast pseudo-trait’s fitted loading energy ($\text{rowSums}(\Lambda^2)$), at or below which it is a collapsed contrast. Default $1e-10$, the campaign code’s own guard. Provisional pending the S3 calibration campaign; labeled evidence: 7/20 phylo_indep seeds at or below $1e-9$, every one reporting convergence = 0 and a positive-definite Hessian.

multinomial_collapse_rel_thresh

Threshold on the ratio of the smallest to the largest fitted contrast loading energy within one multinomial() response’s K-1 contrasts, below which the smallest is flagged as collapsed relative to its siblings. Default Inf (disarmed): those K-1 contrasts are $\pi^2/6$ -correlated siblings through their shared baseline category, not the independent siblings $\psi_{\text{rel_thresh}}$ was calibrated on, so this arm is provisional pending the S3 calibration campaign.

multinomial_rail_thresh

Threshold on the largest absolute off-diagonal correlation of the implied contrast-level covariance within one multinomial() response’s contrasts, at or above which two contrasts are reported as rail-correlated. Only evaluated at tiers with rank $d \geq 2$ – at $d = 1$ every healthy fit reaches $|\rho| = 1$ exactly by row proportionality. Default 0.99, provisional pending the S3 calibration campaign; labeled evidence: 8/20 seeds at or above it.

multinomial_range_collapse_thresh

Threshold on the fitted spatial practical range ($\sqrt{8} / \kappa$) relative to the coordinate-domain diameter (when the fit’s mesh coordinates are reachable), or on the practical range itself in absolute coordinate units (the fallback when they are not), at or below which a multinomial() response’s spatial field is reported as collapsed. Default 0.02, provisional pending the S3 calibration campaign; labeled evidence: collapsed ratios $7e-5$ to $3.4e-4$.

ordinal_loading_runaway_thresh

Threshold on an `ordinal_probit()` (fid 14) trait's largest loading relative to the typical loading among the other ordinal traits, at or above which the loading is reported on its own. Mirrors `loading_runaway_thresh`'s binomial arm. Default Inf (disabled): the tested ordinal thresholds did not meet the predeclared sensitivity and false-positive targets. Finite values are user-selected exploratory cutoffs, not calibrated defaults.

ordinal_loading_absolute_thresh

Threshold on an `ordinal_probit()` trait's largest loading on the link (liability) scale, unit tiers only, at or above which it is reported regardless of the other traits. Meaningful because the probit-liability residual variance is exactly 1 under the Wright/Falconer/Hadfield threshold convention, so a loading is the trait's latent standard deviation in liability units, mirroring `loading_absolute_thresh`'s binomial justification. Default Inf (disabled): the tested ordinal thresholds did not meet the predeclared sensitivity and false-positive targets. Finite values are user-selected exploratory cutoffs, not calibrated defaults.

Details

Scope: optimisation and inference-risk signals for fitted models, including latent-axis rotation, weak-axis, near-zero ψ , residual-scale boundary flags, a binomial near-constant/loading/saturation screen, and the intentional `gllvmTMBcontrol(se = FALSE)` point-estimate path. The table does not calibrate interval coverage, prove formal separation, or prove the selected latent rank by itself. Target-explicit known-DGP simulations will decide when broader interval or rank-selection claims can move beyond diagnostic status.

A WARN row, including `pdHess = FALSE`, means that Wald standard errors or curvature-based inference need more care; it is not by itself proof that the fitted mean, likelihood, or rotation-invariant covariance summaries are unusable.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = 2),
               data = dat, trait = "trait", unit = "site")
check_gllvmTMB(fit)
```

Value

A data frame with columns `component`, `status`, `value`, `threshold`, `message`, and `action`. Status values are "PASS", "WARN", or "FAIL".

compare_dep_vs_two_psi

Canonical likelihood-based cross-check for the paired phylogenetic decomposition

Description

Refits the user's data with full unstructured $T \times T$ phylogenetic and non-phylogenetic trait covariances (phylo_dep + dep) using the same engine, family, link, and unit / cluster grouping as the supplied two-psi fit. Compares the joint two-psi fit's implied $\Sigma_{\text{phy}} = \Lambda_{\text{phy}} \Lambda_{\text{phy}}^T + \Psi_{\text{phy}}$ (and the analogous Σ_{non}) against the unstructured baseline. Per-component RMSE plus a flag field (TRUE when any component disagrees beyond threshold) identifies two-psi identifiability failures.

Usage

```
compare_dep_vs_two_psi(
  fit_two_psi,
  threshold = 0.1,
  phylo_vcv = NULL,
  phylo_tree = NULL
)
```

Arguments

fit_two_psi	A joint two-psi fit returned by <code>gllvmTMB()</code> , e.g. one produced by <code>gllvmTMB(value ~ 0 + trait + phylo_latent(species, d = K_phy, unique = TRUE) + indep(0 + trait species), ...)</code> . The cross-check refits the same data and family with phylo_dep + dep and compares.
threshold	Numeric (default 0.10): relative-disagreement threshold (Frobenius RMSE / Frobenius magnitude of the unstructured estimate) above which a component is flagged.
phylo_vcv, phylo_tree	Optional. The phylogenetic correlation matrix or <code>ape::phylo</code> tree, only needed if <code>fit_two_psi</code> was produced by an older package version that did not store the phylogeny on the fit. Default is to recover them from the fit (<code>fit_two_psi\$phylo_vcv / fit_two_psi\$phylo_tree</code>).

Details

Conceptual basis: Williams et al. (2025) bioRxiv 2025.12.20.695312 Eq. 3, generalised across traits. The unstructured fit has $T(T + 1)/2$ free parameters per tier and is the strongest available benchmark for the implied total covariance. When the joint two-psi fit is well-identified, its implied total Sigma matches the unstructured baseline; when it isn't, the diagnostic flags the disagreement and the user knows to relax the rank or revisit identifiability.

The cross-check separates three questions:

Level 1: can we fit? Both fits converged.

Level 2: total Sigma_phy and Sigma_non agree? This diagnostic answers Level 2 directly. The two estimators target the SAME total covariance with different parameterisations, so agreement is the identifiability check.

Level 3: split into Lambda Lambda^T + Psi? Diagnostic does not test Level 3 directly; Level 2 agreement is the pre-requisite for Level 3 to be meaningful.

Computational scope: tractable for $T \leq \sim 30$. For larger T , prefer `compare_indep_vs_two_psi()`.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
library(ape)
tree <- ape::rcoal(200)
tree$tip.label <- paste0("sp", seq_len(200))
Cphy <- ape::vcv(tree, corr = TRUE)
fit <- gllvmTMB(
  value ~ 0 + trait + phylo_latent(species, d = 1, unique = TRUE) +
    indep(0 + trait | species),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_vcv = Cphy
)
diag <- compare_dep_vs_two_psi(fit)
diag$flag
diag$agreement
```

Value

A list with components:

joint The two-psi fit's implied Σ_{phy} and Σ_{non} ($T \times T$ matrices).

dep The unstructured phylo_dep + dep baseline's implied Σ_{phy} and Σ_{non} ($T \times T$ matrices). May contain NULL entries if the alt fit failed.

agreement Data frame with rows for Σ_{phy} and Σ_{non} , columns *rmse* (Frobenius RMSE between joint and dep), *dep_mag* (Frobenius magnitude of the dep estimate), *rel_disagreement* (*rmse* / *dep_mag*), and *flag* (*rel_disagreement* > *threshold*, or NA when the alt refit failed so the component could not be tested).

flag Logical: TRUE if any component is flagged, FALSE if all tested components agreed, NA if nothing could be tested (distinguishes agreement from a failed refit).

threshold The threshold used.

alt_fit The refit `gllvmTMB_multi` object (or NULL on failure), retained so users can inspect convergence and pull other extractor outputs.

References

Williams, M. J., McGillicuddy, M., Drobniak, S. M., Bolker, B. M., Warton, D. I., & Nakagawa, S. (2025). Fast phylogenetic generalised linear mixed-effects modelling using the `gllmmTMB` R package. *bioRxiv* 2025.12.20.695312. doi:10.64898/2025.12.20.695312

Hadfield, J. D. & Nakagawa, S. (2010). General quantitative genetic methods for comparative biology: phylogenies, taxonomies and multi-trait models for continuous and categorical characters. *Journal of Evolutionary Biology* 23, 494-508. doi:10.1111/j.14209101.2009.01915.x

Meyer, K. & Kirkpatrick, M. (2008). Perils of parsimony: properties of reduced-rank estimates of genetic covariance matrices. *Genetics* 180, 1153-1166. doi:10.1534/genetics.108.090159

Felsenstein, J. (2005). Using the quantitative genetic threshold model for inferences between and within species. *Philosophical Transactions of the Royal Society B* 360, 1427-1434. doi:10.1098/rstb.2005.1669

Felsenstein, J. (2012). A comparative method for both discrete and continuous characters using the threshold model. *American Naturalist* 179, 145-156. doi:10.1086/663681

See Also

[compare_indep_vs_two_psi\(\)](#) for the cheap diagonal fallback when T is large; [extract_Sigma\(\)](#) for the underlying covariance extractor.

compare_indep_vs_two_psi

Cheap diagonal cross-check for the paired phylogenetic decomposition (large T)

Description

Refits the user's data with phylo_indep + indep – a stacked multivariate generalisation of T univariate phylogenetic GLMMs in the spirit of Williams et al. (2025) Eq. 3 – using the same engine, family, link, and grouping as the supplied two-psi fit. Compares the per-trait diagonals of the joint two-psi fit's implied Σ_{phy} (and Σ_{non}) against the marginal-only baseline. Tests **only** the diagonals (cheap); does not test the off-diagonal cross-trait covariances.

Usage

```
compare_indep_vs_two_psi(
  fit_two_psi,
  threshold = 0.1,
  phylo_vcv = NULL,
  phylo_tree = NULL
)
```

Arguments

fit_two_psi	A joint two-psi fit returned by gllvmTMB() , e.g. one produced by <code>gllvmTMB(value ~ 0 + trait + phylo_latent(species, d = K_phy, unique = TRUE) + indep(0 + trait species), ...)</code> . The cross-check refits the same data and family with phylo_dep + dep and compares.
threshold	Numeric (default 0.10): relative-disagreement threshold (Frobenius RMSE / Frobenius magnitude of the unstructured estimate) above which a component is flagged.

phylo_vcv, phylo_tree

Optional. The phylogenetic correlation matrix or `ape::phylo` tree, only needed if `fit_two_psi` was produced by an older package version that did not store the phylogeny on the fit. Default is to recover them from the fit (`fit_two_psi$phylo_vcv / fit_two_psi$phylo_tree`).

Details

Use this diagnostic when T is large ($T \geq 30$) and the unstructured fit needed by `compare_dep_vs_two_psi()` is intractable. Otherwise prefer `compare_dep_vs_two_psi()` (gold standard).

Cross-check intent: the per-trait diagonals are the simplest summary the joint two-psi fit and the marginal baseline both target. If the diagonals already disagree beyond the threshold, no inspection of the cross-trait off-diagonals is warranted.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
diag <- compare_indep_vs_two_psi(fit)
diag$flag
diag$agreement
```

Value

A list with components:

joint Joint two-psi fit's implied `diag(Sigma_phy)` and `diag(Sigma_non)` (named numeric vectors of length T).

indep The marginal-only baseline's `diag(Sigma_phy)` and `diag(Sigma_non)` (same shape; may contain NULL if alt fit failed).

agreement Data frame with rows for `Sigma_phy_diag` and `Sigma_non_diag`, columns `rmse`, `indep_mag`, `rel_disagreement`, and `flag`.

flag, **threshold**, **alt_fit** As in `compare_dep_vs_two_psi()`.

References

Williams, M. J., McGillicuddy, M., Drobniak, S. M., Bolker, B. M., Warton, D. I., & Nakagawa, S. (2025). Fast phylogenetic generalised linear mixed-effects modelling using the `glmmTMB` R package. *bioRxiv* 2025.12.20.695312. doi:10.64898/2025.12.20.695312

See Also

`compare_dep_vs_two_psi()` (canonical, full Sigma); `extract_Sigma()`.

compare_loadings	<i>Compare two loading matrices after Procrustes alignment</i>
------------------	--

Description

When two latent-variable fits use the same traits, the loading matrices are only identified up to rotation and sign. Procrustes alignment finds the orthogonal transform that brings one matrix as close as possible to the other, then reports the residual disagreement. This is mainly a validation helper for comparing `gllvmTMB()` with another implementation.

Usage

```
compare_loadings(Lambda_a, Lambda_b)
```

Arguments

`Lambda_a`, `Lambda_b`
Two $n_{\text{traits}} \times d$ loading matrices.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit_g <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = 2),
  data = df, trait = "trait", unit = "site")
fit_t <- glmmTMB::glmmTMB(value ~ 0 + trait + rr(0 + trait | site, d = 2),
  data = df, REML = FALSE)
L_g <- extract_ordination(fit_g, "unit")$loadings
L_t <- attr(glmmTMB::ranef(fit_t)$cond$site, "loadings")
compare_loadings(L_g, L_t)
```

Value

A list with the optimal rotation R , the rotated `Lambda_a_rot`, and the Frobenius distance after alignment.

compare_Sigma_table	<i>Compare fitted Sigma-table rows with a known truth matrix</i>
---------------------	--

Description

`compare_Sigma_table()` joins report-ready `extract_Sigma_table()` rows to a known covariance or correlation matrix. It is designed for simulation and teaching articles that need estimate-vs-truth tables without hand-indexing matrices inside the article.

Usage

```
compare_Sigma_table(
  x,
  truth,
  level = "unit",
  part = c("total", "shared", "unique"),
  measure = c("covariance", "correlation"),
  entries = c("unique", "all", "upper", "lower", "offdiag", "diag"),
  link_residual = c("auto", "none")
)
```

Arguments

<code>x</code>	A fit returned by <code>gllvmTMB()</code> , an admitted engine = "julia" bridge fit, or a data frame returned by <code>extract_Sigma_table()</code> .
<code>truth</code>	Square numeric covariance or correlation matrix. Row and column names should match the trait names in <code>x</code> ; unnamed matrices are accepted only when their dimension matches the traits in <code>x</code> .
<code>level</code> , <code>part</code> , <code>measure</code> , <code>entries</code> , <code>link_residual</code>	Passed to <code>extract_Sigma_table()</code> when <code>x</code> is a fitted model.

Details

Scope: the helper compares fitted or precomputed `extract_Sigma_table()` rows against one supplied truth matrix (covered by the package's Sigma-table validation tests). It is a table helper only: it does not compute uncertainty, simulate data, or validate calibration. For the first visual comparison layer, use `plot_Sigma_comparison()`; richer article-specific calibration summaries remain future visualization work.

Value

A data frame with the columns from `extract_Sigma_table()` plus `truth`, `error`, `abs_error`, and `comparison_status`.

See Also

`extract_Sigma_table()`, `plot_Sigma_comparison()`, `plot_Sigma_table()`.

Examples

```
rows <- data.frame(
  estimand = "R_unit[length,mass]",
  trait_i = "length",
  trait_j = "mass",
  i = 1L,
  j = 2L,
  level = "unit",
  component = "total",
  matrix = "R",
```

```

estimate = 0.62,
lower = NA_real_,
upper = NA_real_,
interval_method = "none",
interval_status = "none",
scale = "correlation",
diagonal = FALSE,
triangle = "upper"
)
truth_R <- matrix(c(1, 0.6, 0.6, 1), 2,
  dimnames = list(c("length", "mass"), c("length", "mass")))
)
compare_Sigma_table(rows, truth_R, measure = "correlation")

```

confint.gllvmTMB_multi

Confidence intervals for a fitted gllvmTMB model

Description

Returns 95% (or other-level) confidence intervals for fixed effects, variance components, and trait covariance matrices, with three method choices:

Usage

```

## S3 method for class 'gllvmTMB_multi'
confint(
  object,
  parm,
  level = 0.95,
  method = c("profile", "wald", "bootstrap"),
  nsim = 500L,
  seed = NULL,
  ...
)

```

Arguments

object	A fit returned by <code>gllvmTMB()</code> .
parm	One of: • "Sigma_unit" – between-unit trait covariance matrix. • "Sigma_unit_obs" – within-unit trait covariance matrix. • "Sigma_cluster" and "Sigma_cluster2" – diagonal covariance matrices for the first and second extra grouping tiers. "profile" and "wald" return diagonal variance intervals; "bootstrap" is intentionally blocked until a simulate-refit calibration gate is added for these tiers. • "sigma_phy" – per-trait phylogenetic standard deviations.

	• Legacy aliases "Sigma_B" and "Sigma_W", retained for existing scripts. • "Lambda" (all free entries of Lambda_unit), "Lambda:i,j" (single entry), or "Lambda:i,j;k,l" (multiple entries, semicolon-separated). Routes to loading_ci() / loading_profile(). For these tokens the method choices are c("wald", "wald_asym", "profile") with default "wald". Wald and profile rows target raw Lambda; "wald_asym" targets the standardised loading rho and labels rows accordingly. • "icc" (all traits), "icc:<trait_name>" (one trait by name), "icc:<t1>;<t2>" (multiple by name), or "icc:[1,3]" (1-based trait indices). Routes to extract_repeatability() with "wald" (default) or "bootstrap". The former profile token is withdrawn because it estimated a diagonal-only ratio rather than canonical full-covariance repeatability. • "phylo_signal" / "phylo_signal:<trait>" etc. – same grammar as "icc". Routes to profile_ci_phylo_signal() for "profile" and the companion Wald/bootstrap helpers for "wald" / "bootstrap". For multi-component fits, "profile" falls back to numerical Wald bounds with an explicit method label until the full fix-and-refit profile is implemented. • "communality:<tier>" (one tier, all traits) or "communality:<tier>:<trait>" (one tier, one trait). Tier is one of "unit" / "unit_obs" / "phy" (legacy "B" / "W"). Defaults to Wald; bootstrap is available explicitly. The nonlinear profile prototype is withheld. • "rho:<tier>;i,j" (one pair) or "rho:<tier>;i,j;k,l" (multiple pairs). Tier is one of "unit" / "unit_slope" / "unit_obs" / "phy" / "spatial" (legacy "B" / "W" / "spde"). "unit_slope" indexes the augmented 2T coefficient vector by numeric position and is currently a not currently an interval target. Other tiers route to extract_correlations() ("fisher-z" / "wald" / "bootstrap"); Fisher-z is the default for this confint() token. The nonlinear profile prototype is withheld. • "proportion" (all components, all traits), "proportion:<component>" (one component, all traits), "proportion:<component>:<trait>" (one (component, trait)), "proportion:<component>:<t1>;<t2>" (one component, multiple traits), or "proportion:<c1>;<c2>" (multiple components). Components are by name ("shared_unit", "unique_unit", "shared_unit_obs", "unique_unit_obs", "unique_cluster", "unique_cluster2", "shared_phy", "unique_phy", "link_residual"). Defaults to Wald; bootstrap is available explicitly. The nonlinear profile prototype is withheld. • An integer index vector or character vector of fixed-effect term names (same as the standard confint() interface). • Missing (default) – all fixed-effect parameters.
level	Confidence level in (0, 1). Default 0.95.
method	One of "profile" (default), "wald", "bootstrap". For parm = "Lambda..." the accepted methods are c("wald", "wald_asym", "profile") with default "wald" (matching the base R confint() convention). "wald_asym" returns Fisher-z intervals for standardised loadings rather than raw Lambda.
nsim	Number of bootstrap replicates passed to bootstrap_Sigma() when method = "bootstrap". Default 500. Use a small value (e.g. 50) during development or testing.

seed	Optional integer RNG seed forwarded to <code>bootstrap_Sigma()</code> (only meaningful when method = "bootstrap").
...	Additional arguments currently unused.

Details

- `method = "profile"` (default): profile-likelihood bounds via `TMB::tmbprofile() + stats::uniroot()`. A successful numerical route can represent asymmetric likelihood shape, but its reference distribution and coverage remain target-specific.
- `method = "wald"`: Gaussian-approximation CIs from `sd_report`. Fastest; poor near boundaries.
- `method = "bootstrap"`: fitted-model parametric bootstrap via `bootstrap_Sigma()`. Slowest; inspect failed refits, Monte Carlo resolution, and whether simulation covers every active model component.

Fixed-effect and direct-parameter intervals use the requested method where supported, and Sigma-matrix intervals accept canonical `parm = "Sigma_unit" / "Sigma_unit_obs"` names, diagonal extra-grouping `"Sigma_cluster" / "Sigma_cluster2"` names, plus legacy `"Sigma_B" / "Sigma_W"` aliases. Profile intervals for full decomposed Sigma entries fall back to bootstrap because those entries are nonlinear functions of rotation-equivalent loadings and diagonal Ψ . Non-Gaussian bootstrap and broader derived-target coverage are not universally calibrated; inspect the returned method and the target-specific article before reporting bounds. Fits using non-unit likelihood weights target a weighted estimating objective, not the ordinary likelihood used by these interval routes. `confint()` therefore refuses them; no sandwich interval is currently certified.

Main `parm`-class dispatch paths:

- **Sigma matrices** – when `parm` is one of `"Sigma_unit"`, `"Sigma_unit_obs"`, `"Sigma_cluster"`, `"Sigma_cluster2"`, or `"sigma_phy"` (legacy aliases `"Sigma_B"` and `"Sigma_W"` still work), returns a tidy data.frame with columns `parameter`, `estimate`, `lower`, `upper`, `method`. Profile is computed element-wise via `TMB::tmbprofile()` for the diagonal entries; structural-zero off-diagonal entries in pure-diagonal tiers are labelled `"structural_zero"`; off-diagonals for reduced-rank tiers fall back to bootstrap (full Sigma sampling) since they mix two parameters in a non-linear way.
- **Fixed effects / variance components** – when `parm` is missing, an integer index, or a character vector of fixed-effect term names, returns a numeric matrix with rows = parameters and columns = lower / upper bounds (same shape as `stats::confint()`). Method choice applies.
- **Lambda and derived summaries** – `"Lambda..."`, `"icc..."`, `"phylo_signal..."`, `"communality..."`, `"rho..."`, and `"proportion..."` tokens dispatch to their corresponding direct-profile, Wald, or bootstrap helper where that route is implemented. Nonlinear penalty-profile prototypes are withheld.

Value

- **Sigma path** – a data.frame with columns `parameter` (character, e.g. `"Sigma_unit[t1,t1]"`), `estimate` (point estimate), `lower`, `upper`, and `method` (the method used for that row, or `"structural_zero"` for off-diagonal entries that are fixed at zero by a diagonal-only tier). The parameter prefix follows the requested `parm`, so legacy calls still return legacy `"Sigma_B[...]"` or `"Sigma_W[...]"` labels.

- **Lambda path** – a data.frame with columns parameter (e.g. "Lambda[trait_1,LV1]" for raw rows or "rho[trait_1,LV1]" for standardised rows), estimate, lower, upper, method, loading_scale, pd_hessian, and ci_status. When pdHess = FALSE on a Wald path, lower/upper are NA and ci_status flags the reason).
- **Derived-quantity path** ("icc" / "phylo_signal" / "communality" / "rho" / "proportion") – a numeric matrix with two columns named after the requested level (e.g. "2.5 %" / "97.5 %") and rownames identifying the entry, e.g. "icc:trait_1", "communality:unit:trait_1", "rho:unit:1,2", or "proportion:shared_unit:trait_1".
- **Fixed-effects / variance-component path** – a numeric matrix with rownames = parameter names and two columns named "2.5 %" / "97.5 %" (or the analogous quantiles for the requested level).

References

- Pawitan, Y. (2001). *In All Likelihood: Statistical Modelling and Inference Using Likelihood*, Oxford University Press, ch. 9.
- Venzon, D. J. & Moolgavkar, S. H. (1988). A method for computing profile-likelihood-based confidence intervals. *Applied Statistics* **37**, 87-94. doi:10.2307/2347496
- McCune, K. B., *et al.* (2024) coxme_icc_ci() – the Nakagawa-authored coxme-based profile-CI helper that inspired this work, in <https://github.com/kelseybmccune/Time-to-Event-Repeatability/blob/main/R/rptRsurv.R>.

See Also

[bootstrap_Sigma\(\)](#), [extract_Sigma\(\)](#), [extract_correlations\(\)](#), [extract_repeatability\(\)](#), [extract_communality\(\)](#), [tmbprofile_wrapper\(\)](#), [loading_ci\(\)](#), [loading_profile\(\)](#), [gllvmTMB\(\)](#).

Examples

```
## Fit a tiny example
set.seed(1)
s <- simulate_site_trait(
  n_sites = 30, n_species = 4, n_traits = 3,
  mean_species_per_site = 4,
  Lambda_B = matrix(c(0.9, 0.4, -0.3), 3, 1),
  psi_B = c(0.20, 0.15, 0.10),
  beta = matrix(0, 3, 2), seed = 1
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 1),
  data = s$data,
  trait = "trait",
  unit = "site"
)

## Bootstrap CIs for the between-site covariance matrix.
## Fifty refits illustrate the API; they do not establish interval coverage.
## Inspect failed refits and use more replicates for a scientific analysis.
ci_unit <- confint(fit, parm = "Sigma_unit", method = "bootstrap",
  nsim = 50, seed = 42)
```

```
ci_unit

## Wald CIs for fixed effects
confint(fit)
```

confint_inspect	<i>Inspect profile confidence-interval shape for a fitted model</i>
-----------------	---

Description

Use `confint_inspect()` when a profile confidence interval is missing, one-sided, very wide, or hard to trust. Most users should start with `confint.gllvmTMB_multi()` or the matching `extract_*()` interval helper; this function is the advanced visual check for a direct profile target.

Usage

```
confint_inspect(
  fit,
  parm,
  level = 0.95,
  ystep = 0.5,
  ytol = NULL,
  parm.range = c(-Inf, Inf)
)
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> .
<code>parm</code>	Character. A single profile-target label from <code>profile_targets()</code> . Examples: <code>"sigma_eps"</code> , <code>"sd_B[1]"</code> , <code>"phi_nbinom2[2]"</code> , <code>"b_fix[1]"</code> . Derived targets are not accepted by this direct-parameter diagnostic.
<code>level</code>	Confidence level. Default 0.95.
<code>ystep</code>	Profile grid spacing on the deviance scale, passed to <code>TMB::tmbprofile()</code> . Default 0.5.
<code>ytol</code>	Profile maximum-deviance budget, passed to <code>TMB::tmbprofile()</code> . Default NULL, which sizes the budget for the requested level (the crossing threshold plus a margin). A budget below the threshold cannot produce a crossing, so the bound would come back infinite; supply a number only if you want to override that.
<code>parm.range</code>	Two-element numeric range (in the TMB-link scale) for the profile walk, passed to <code>TMB::tmbprofile()</code> . Default <code>c(-Inf, Inf)</code> .

Details

It returns the full profile-likelihood curve, the deviance bounds, and (when ggplot2 is available) a ggplot visualisation showing the curve, the MLE, the chi-squared threshold, and the resulting confidence-interval bounds. It surfaces a small set of numerical shape warnings before the user interprets the bound; the returned label is not a goodness or calibration certificate.

This helper inspects direct profile-ready targets from `profile_targets()`. Nonlinear derived targets such as communality, repeatability, and trait correlation do not currently have a public profile-inspection route. A working direct curve does not establish empirical interval coverage for every model class.

Common patterns to look for in the returned plot (see the [profile-likelihood article](#)):

- **Quadratic and approximately symmetric near the MLE:** Wald and profile may agree closely for this target.
- **Asymmetric or skewed:** inspect the endpoint status and constrained-fit stability before preferring either interval.
- **Flat near the MLE:** the target may be weakly informed, or the numerical profile may need closer inspection.
- **Hits a parameter-range bound:** interpret the endpoint on the target's transformed scale and distinguish a natural boundary from a truncated search.
- **No usable crossing:** treat the endpoint as unavailable; do not replace it automatically with bootstrap output unless simulation, refits, failed replicate counts, and Monte Carlo resolution are credible for that target.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(value ~ 0 + trait +
               latent(0 + trait | site, d = 1),
               data = sim$data,
               trait = "trait",
               unit = "site")
inspect <- confint_inspect(fit, parm = "sigma_eps")
inspect$bounds      # the CI + Wald comparison
inspect$diagnostics # any shape flags
inspect$plot        # ggplot of the profile curve
```

Value

An object of class `gllvmTMB_confint_inspect` with components:

`$curve` Data frame with one row per profile-grid point. Columns: `parm` (user-facing label), `parm_value_natural` (parameter value on the natural scale after the registered transformation), `parm_value_link` (parameter value on the TMB optimisation scale), `nll` (negative log-likelihood), `deviance_drop` ($2 * (nll - mle_nll)$, the chi-squared statistic), `excess_over_threshold` (`deviance_drop - chi2_threshold`; negative is inside CI, positive is outside), `in_ci` (logical).

`$bounds` Data frame, 1 row: `parm`, `estimate_natural`, `lower_natural`, `upper_natural`, `wald_lower_natural`, `wald_upper_natural`, `wald_profile_disagree_lower`, `wald_profile_disagree_upper`.

Wald bounds come from `fit$sd_report$cov.fixed` via the delta method; disagreement flags are TRUE when the absolute difference exceeds 25% of the Wald half-width (a heuristic for "look at the plot").

`$plot` A ggplot object visualising the curve, MLE, chi-squared threshold, profile bounds (vertical lines), and Wald bounds (dashed vertical lines for comparison). NULL if ggplot2 is not installed.

`$diagnostics` Named character vector flagging any of: "no_heuristic_warning" (none of this helper's shape warnings fired; this does not prove a quadratic profile), "asymmetric" (skewed, Wald-vs-profile disagrees), "flat_at_mle", "hits_lower_bound", "hits_upper_bound", "no_lower_crossing", "no_upper_crossing", "profile_failed".

`$call` The `match.call()` of the invocation.

See Also

`confint.gllvmTMB_multi()` (the corresponding CI extractor; same parm vocabulary), `profile_targets()` (the target inventory), `tmbprofile_wrapper()` (the lower-level profile API), and the [profile-likelihood article](#).

confirmatory_lambda	<i>Build a confirmatory lambda_constraint matrix from group membership</i>
---------------------	--

Description

Construct the `n_species` by `d` confirmatory loading-constraint matrix `M` from a discrete functional-group hypothesis. Each named group is pinned to load on a specific latent axis (zero on all other axes), and one anchor species per axis is pinned at +1 to fix the scale. Groups present in `group` but absent from `loads_on` remain free on all axes.

Usage

```
confirmatory_lambda(
  species,
  group,
  d,
  loads_on,
  anchors = NULL,
  axis_labels = NULL
)
```

Arguments

<code>species</code>	Character vector of species names. Becomes the rownames of the returned matrix. Must match the species (i.e. trait levels) used in the <code>gllvmTMB()</code> fit.
<code>group</code>	Character or factor vector parallel to <code>species</code> ; the functional-group code for each species.

d	Integer; number of latent axes (matches the d argument of <code>latent(0 + trait site, d = d)</code>).
loads_on	Named list or named integer vector mapping group code to latent axis index. For example, <code>list(A = 1L, B = 2L)</code> means species in group A load on axis 1 only (zero on all other axes), species in group B load on axis 2 only. Groups present in group but absent from <code>loads_on</code> remain free on all axes.
anchors	Optional length-d character vector; one anchor species per axis (pinned at +1). If NULL (the default), anchors are auto-picked as the first species belonging to the group that loads on each axis. Use NA to skip the anchor on a specific axis.
axis_labels	Optional length-d character vector for the matrix column names. Defaults to <code>c("LV1", "LV2", ...)</code> .

Details

This is the recommended starting point for confirmatory JSDMs where prior knowledge takes the form "species in group A respond to gradient 1, species in group B respond to gradient 2, ...". For free-form constraint patterns (psychometric CFA, idiosyncratic species-by-species pins), build the matrix directly; for purely statistical identification scaffolding when no biology is in play, see [suggest_lambda_constraint\(\)](#).

Value

An `length(species)` by `d` numeric matrix with NA entries (estimated), 0 entries (pinned to zero), and 1 entries (pinned at the anchor), ready to pass to `gllvmTMB()` as `lambda_constraint = list(unit = M)`.

See Also

[suggest_lambda_constraint\(\)](#) for the lower-level statistical identification scaffold, `gllvmTMB()` for the `lambda_constraint` argument.

Examples

```
species <- c(paste0("A_", 1:3), paste0("B_", 1:3), paste0("C_", 1:4))
group    <- c(rep("A", 3), rep("B", 3), rep("C", 4))

# Group A loads on axis 1; group B loads on axis 2; group C is free.
M <- confirmatory_lambda(
  species = species,
  group   = group,
  d       = 2L,
  loads_on = list(A = 1L, B = 2L)
)
M
```

cumulative_logit

*Cumulative-logit ordered categorical missing-predictor family***Description**

`cumulative_logit()` declares the predictor-model family for an **ordered categorical** missing predictor used inside `mi()` with `impute_model(family = cumulative_logit())`. The missing ordered predictor is marginalised EXACTLY by a finite-state cumulative-logit sum over its K categories ($K \geq 3$); there is no latent variable. The predictor link is cumulative **logit**, with $K - 1$ free cut-points $c_1 < \dots < c_{\{K-1\}}$ and a cell probability $\Pr(x = k \mid z) = F(c_k - \eta_x) - F(c_{\{k-1\}} - \eta_x)$, where $\eta_x = X_x \beta_x$ (no separate predictor intercept; the cutpoints carry the location). The missing predictor must be an **ordered factor** (or integer category scores) with at least three levels; a two-level predictor uses `binomial()` (the binary route).

Usage

```
cumulative_logit()
```

Details

This is a predictor-model family tag consumed by `impute_model(family = )`, NOT a response family for `gllvmTMB()`. The predictor cumulative-logit link is DISTINCT from the `ordinal_probit()` RESPONSE family (a probit threshold model).

Value

A `gllvmTMB_impute_family` object for the family argument of `impute_model()`.

See Also

`impute_model()`, `binomial()` for a binary missing predictor, and `ordinal_probit()` for the ordinal RESPONSE family.

Examples

```
impute_model(score ~ z, family = cumulative_logit())
```

dep

*Full unstructured trait covariance: dep(0 + trait | g)***Description**

Canonical name for the **always-alone full-unstructured** trait covariance term. Fits a full $T \times T$ Σ with $T(T + 1)/2$ free parameters via a Cholesky parameterisation $\Sigma = \mathbf{L}\mathbf{L}^\top$. Mathematically identical to `latent(0 + trait | g, d = T)` standalone (where T is the number of traits) – the engine’s existing packed-triangular $\mathbf{A}$ at full rank IS the Cholesky factor of an unstructured Σ . The keyword choice is documentary: `dep` declares user intent that the full unstructured trait covariance is the model.

Usage

```
dep(formula)
```

Arguments

formula $0 + \text{trait} \mid g$ style formula (LHS is the trait factor, typically $0 + \text{trait}$; RHS is the grouping factor).

Details

This completes the structural-mode quartet:

Decomposition (latent **by default**) Shared low-rank loadings plus the default trait-specific Psi companion: $\Sigma = \Lambda\Lambda^\top + \Psi$. $TK + T$ parameters (rotation-removed).

Marginal (indep **standalone**) Per-trait variance with identity correlation: $\Sigma = \text{diag}(\sigma_t^2)$. T parameters.

Full / unstructured (dep **standalone**) Free unstructured Σ , PSD via Cholesky. $T(T + 1)/2$ parameters.

Use `dep()` when you want to commit to the full unstructured interpretation explicitly. This is a standard multivariate response covariance model. Computational scope: tractable for $T \leq \sim 30$ with the default optimiser; for larger T , prefer `latent(d = K)` with $K \ll T$ as the rank-reduced approximation.

Mutual exclusion:

Combining `dep(0 + trait | g)` with `latent(0 + trait | g, d = K)` on the same grouping is **over-parameterised**. Combining `dep` with `unique` or `indep` on the same grouping is **redundant** (`dep standalone` already includes the trait-level diagonal). The parser raises `cli::cli_abort()` in any of these cases.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
# Full unstructured trait covariance fit:
fit <- gllvmTMB(value ~ 0 + trait + dep(0 + trait | site),
               data = df, trait = "trait", unit = "site")

# The mathematically-equivalent decomposition form (full rank):
fit <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = T),
               data = df, trait = "trait", unit = "site")

# ERROR: dep + latent on the same grouping is over-parameterised.
# gllvmTMB(value ~ 0 + trait +
#           dep(0 + trait | site) +
#           latent(0 + trait | site, d = 2), data = df)
```

Value

A formula marker; never evaluated.

See Also

`latent()`, `indep()`, `phylo_dep()`, `spatial_dep()`, `extract_Sigma()`.

`deviance.gllvmTMB_multi`

Deviance of a fitted gllvmTMB model

Description

$-2 * \text{logLik}(\text{object})$, delegating through `logLik.gllvmTMB_multi()` rather than recomputing anything independently. This is deliberate: on a penalised (ridged) fit – `aghq_ridge` set to a finite value – `logLik()` already discloses, with a warning, that the returned value is the unpenalised log-likelihood evaluated AT a penalised MAP point, not at its own maximum. Because `deviance()` is defined purely in terms of `logLik()` here, that same disclosure fires exactly once and is not repeated or re-derived.

Usage

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

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>...</code>	Currently unused.

Value

A single numeric value, $-2 * \text{as.numeric}(\text{logLik}(\text{object}))$. On a ridged fit this carries the same MAP-point caveat as `logLik()`: the value is -2 times the unpenalised likelihood at the penalised estimate, not at its own maximum, and the warning issued by `logLik()` is inherited unchanged.

`diagnose_kernel_separability`

Diagnose fixed-kernel separability before fitting

Description

`diagnose_kernel_separability()` compares two or more dense fixed kernels on the same levels before they are used in a multi-kernel model. It is a pre-fit diagnostic rather than evidence that the components are separately identifiable: when candidate kernels are highly overlapping, component-specific Gamma blocks are weak evidence and should be treated as descriptive unless simulations justify the split.

Use this helper to screen candidate `K_phy` and `K_tip` definitions, including raw-network and residualized-network choices, before fitting `kernel_latent(..., name = ...)` tiers. Note its limits: this is a diagnostic, not recovery evidence, interval calibration, or an in-engine identifiability proof.

Usage

```
diagnose_kernel_separability(
  ...,
  thresholds = c(near_orthogonal = 0.25, high = 0.7)
)
```

Arguments

... Two or more named numeric square kernel matrices with the same dimensions and level order.

thresholds Named numeric vector with entries `near_orthogonal` and `high`. Similarity below `near_orthogonal` is labelled "near_orthogonal"; similarity below `high` is "moderate"; similarity at or above `high` is "high".

Value

A list of class `gllvmTMB_kernel_separability` with:

similarity A symmetric matrix of off-diagonal Frobenius-style similarities between kernels.

pairs A data frame with pair labels, similarity, overlap class, and a conservative recommendation.

thresholds The thresholds used for the classes.

note A short claim-boundary note.

Examples

```
A_H <- diag(2)
A_P <- diag(2)
rownames(A_H) <- colnames(A_H) <- c("H1", "H2")
rownames(A_P) <- colnames(A_P) <- c("P1", "P2")
W_phy <- matrix(c(1, 0.2, 0.2, 1), 2, 2,
  dimnames = list(rownames(A_H), rownames(A_P)))
W_tip <- matrix(c(0.2, 1, 1, 0.2), 2, 2,
  dimnames = list(rownames(A_H), rownames(A_P)))
K_phy <- make_cross_kernel(A_H, A_P, W_phy, rho = 0.5)
K_tip <- make_cross_kernel(A_H, A_P, W_tip, rho = 0.5)
diagnose_kernel_separability(phy = K_phy, tip = K_tip)
```

diagnostic_table

Extract report-ready tables from diagnostic objects

Description

`diagnostic_table()` turns the metadata attached by `predictive_check()` and `residuals.gllvmTMB_multi()` into plain data frames. It is the public path for articles, reports, and tests that need the plotted diagnostic rows, residual rows, row-status counts, or fit-health table without inspecting object attributes directly.

Usage

```
diagnostic_table(
  x,
  table = c("data", "row_status", "fit_health_status", "check_gllvmTMB")
)
```

Arguments

<code>x</code>	A ggplot returned by <code>predictive_check()</code> or a data frame returned by <code>residuals(fit, type = "randomized_quantile")</code> or <code>residuals(fit, type = "simulation_rank")</code> .
<code>table</code>	Table to extract. "data" returns plotted data for predictive-check plots or the residual rows themselves for residual objects; "row_status" counts residual row status values when present; "fit_health_status" returns counts of PASS / WARN / FAIL rows from <code>check_gllvmTMB()</code> ; "check_gllvmTMB" returns the full fit-health table attached to the diagnostic object.

Details

Scope: table extraction from existing `attr(x, "gllvmTMB_diagnostic")` metadata on `predictive_check()` plots and diagnostic residual data frames. This helper does not compute new diagnostics, refit models, run formal residual tests, or calibrate uncertainty. Richer article tables can build on this stable metadata path once the diagnostic examples are restored.

Value

A plain data frame.

Examples

```
set.seed(3)
n <- 24
df <- data.frame(
  unit = factor(rep(seq_len(n), each = 2)),
  trait = factor(rep(c("a", "b"), n)),
  value = rpois(2 * n, lambda = 2)
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | unit, d = 1),
  data = df,
  trait = "trait",
  unit = "unit",
  family = poisson()
)
p <- predictive_check(fit, type = "rq_qq", seed = 1)
diagnostic_table(p, table = "row_status")
diagnostic_table(p, table = "fit_health_status")
diagnostic_table(p, table = "data")

r <- residuals(fit, type = "randomized_quantile", seed = 1)
diagnostic_table(r, table = "data")
```

extract_coevolution_modules

Extract cross-lineage coevolutionary modules

Description

`extract_coevolution_modules()` standardizes a component-specific cross-lineage covariance block and decomposes it into coupled trait axes. For a named coevolution kernel tier, the helper computes $R = \text{Sigma_row}^{-1/2} \Gamma \text{Sigma_col}^{-1/2}$ from the shared covariance returned by [extract_Sigma\(\)](#), then applies a singular-value decomposition.

This is a point-estimate derived-output helper, covered by the package's validation tests, for fitted dense-kernel coevolution models whose component-specific Γ blocks are already extractable with [extract_Gamma\(\)](#). It does not report uncertainty, choose the biological rank, estimate ρ , or calibrate null thresholds. Use simulation, bootstrap, and kernel-separability checks before treating the returned axes as scientific evidence.

Usage

```
extract_coevolution_modules(
  fit,
  level,
  row_traits,
  col_traits,
  scale = c("shape", "effect"),
  n_modules = NULL,
  tol = sqrt(.Machine$double.eps)
)
```

Arguments

<code>fit</code>	A fitted <code>gllvmTMB_multi</code> object.
<code>level</code>	Character scalar naming the fitted covariance tier.
<code>row_traits, col_traits</code>	Character vectors naming the row-lineage and column-lineage traits.
<code>scale</code>	Character scalar. "shape" (default) uses $\Gamma_{\text{shape}} = \Lambda_{\text{row}} \Lambda_{\text{col}}^T$; "effect" uses the fixed- ρ Γ_{effect} available for kernels built by make_cross_kernel() .
<code>n_modules</code>	Optional positive integer limiting the number of returned singular axes. By default all axes are returned.
<code>tol</code>	Numerical tolerance for the generalized inverse square roots of the within-lineage shared covariance blocks.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```

mods <- extract_coevolution_modules(
  fit,
  level = "phy",
  row_traits = c("host_size", "host_defence"),
  col_traits = c("partner_size", "partner_attack")
)
mods$modules
mods$row_axes
mods$col_axes

```

Value

A list with `R`, `modules`, `row_axes`, and `col_axes`. `R` is the standardized cross-lineage correlation-like block. `modules` has one row per singular axis with its singular value and squared-value share. `row_axes` and `col_axes` contain trait loadings for each coupled axis.

extract_communality	<i>Communality of each trait</i>
---------------------	----------------------------------

Description

$$c_t^2 = \frac{(\mathbf{\Lambda}\mathbf{\Lambda}^\top)_{tt}}{(\mathbf{\Lambda}\mathbf{\Lambda}^\top)_{tt} + \psi_{tt}}.$$

Usage

```

extract_communality(
  fit,
  level = c("unit", "unit_obs", "phy", "B", "W"),
  link_residual = c("auto", "none"),
  ci = FALSE,
  conf_level = 0.95,
  method = c("wald", "bootstrap", "profile"),
  nsim = 500L,
  seed = NULL
)

```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> . A <code>bootstrap_Sigma()</code> result is also accepted when it contains communality summaries; in that case the function reuses the stored point estimates and percentile bounds rather than refitting.
<code>level</code>	"unit" (between-unit), "unit_obs" (within-unit), or "phy" (phylogenetic tier). Legacy aliases "B" and "W" are accepted with a deprecation warning.

link_residual	For binomial fits: "auto" (default) adds the link-specific implicit residual to the denominator; "none" returns communalities on the fitted model covariance scale without link-residual additions.
ci	Logical. When TRUE, returns a tidy data frame with confidence-interval columns; when FALSE (the default), returns a plain named numeric vector for backward compatibility.
conf_level	Confidence level when ci = TRUE. Default 0.95.
method	One of "wald" (default), "bootstrap", or "profile". Only used when ci = TRUE. The nonlinear penalty-profile prototype is currently withheld from the public release; requesting "profile" stops with an explanation. Bootstrap is a slower simulation-and-refit option and is not automatically coverage-calibrated.
nsim	Number of bootstrap replicates when method = "bootstrap". Default 500.
seed	Optional RNG seed for the bootstrap.

Details

The proportion of trait t 's variance that is *shared* with the other traits via the latent factors. Bounded between 0 and 1. Calls `extract_Sigma()` internally for the chosen level, so the diagonal uses the full $\Sigma = \Lambda\Lambda^\top + \Psi$ decomposition. Ordinary `latent()` includes Ψ by default.

Caveat: communality with no-Psi fits:

If the fit uses `latent(..., unique = FALSE)` at the requested level, then $\Psi = \mathbf{0}$ and $c_t^2 = 1$ for every trait. This is mathematically correct for the no-residual subset but tells you nothing about trait integration. The `extract_Sigma()` advisory message will fire to flag this. To get meaningful communalities, use ordinary `latent()` with the default `unique = TRUE`.

For binomial fits the link-specific implicit residual ($\pi^2/3$ for logit, 1 for probit, $\pi^2/6$ for cloglog) is added to the denominator by default; pass `link_residual = "none"` to suppress.

Value

When `ci = FALSE`: a numeric vector indexed by trait. When `ci = TRUE`: a data frame with columns `trait`, `tier`, `c2`, `lower`, `upper`, `method`. For a `bootstrap_Sigma` input, the interval columns are copied from the bootstrap object.

See Also

`extract_Sigma()`; `extract_ICC_site()`; `extract_correlations()`; `extract_repeatability()`; `confint.gllvmTMB_multi()`.

Examples

```
sim <- simulate_site_trait(
  n_sites = 20, n_species = 6, n_traits = 4,
  mean_species_per_site = 4, seed = 1
)
fit <- gllvmTMB(
  value ~ 0 + trait +
    latent(0 + trait | site, d = 2),
```

```

    data = sim$data,
    trait = "trait",
    unit = "site"
  )
  ## Per-trait between-unit communality.
  extract_communality(fit, level = "unit")
  ## With delta-method Wald CIs.
  extract_communality(fit, level = "unit", ci = TRUE, method = "wald")
  # Fifty refits illustrate the API; they do not establish interval coverage.
  boot <- bootstrap_Sigma(fit, n_boot = 50, level = "unit",
    what = "communality", progress = FALSE)
  extract_communality(boot, level = "unit", ci = TRUE)

```

extract_correlations *Extract cross-trait correlations*

Description

Returns the implied cross-trait correlations from a fit returned by [gllvmTMB\(\)](#) at one or more covariance levels. Use the canonical input names "unit", "unit_obs", "phy", and "spatial"; legacy aliases "B", "W", and "spde" still work. The helper returns 95% (or other-level) point estimates by default. Fisher-z and bootstrap intervals can be requested explicitly with the method argument; the former profile token is retained only to return a clear withdrawal error:

Usage

```

extract_correlations(
  fit,
  tier = "all",
  pair = NULL,
  level = 0.95,
  method = c("none", "fisher-z", "profile", "wald", "bootstrap"),
  n_eff = NULL,
  nsim = 500L,
  seed = NULL,
  link_residual = c("auto", "none")
)

```

Arguments

fit	A fit returned by gllvmTMB . Julia bridge (engine = "julia") fits expose the ordinary unit tier only and carry no correlation-interval payload, so they return point-only rows: the same schema with lower and upper set to NA, method = "none", and interval_status = "none". Use engine = "tmb" when you need correlation confidence intervals.
tier	Character vector. Use "all" (the default) to request every level present in the fit. Canonical inputs are "unit", "unit_obs", "phy", and "spatial"; legacy aliases "B", "W", and "spde" are accepted.

pair	Optional length-2 character or integer vector specifying one trait pair (c("trait_1", "trait_2") or c(1, 2)). When supplied, only that pair is returned for each requested tier. Default NULL (all pairs).
level	Confidence level in (0, 1). Default 0.95.
method	One of "none" (default), "fisher-z", "wald" (alias of "fisher-z"), or "bootstrap". The accepted "profile" token is withdrawn and stops with an explanation. See Details.
n_eff	Optional positive integer (≥ 4): override the effective sample size used in Fisher's $\widehat{SE}(\hat{z}) = 1/\sqrt{n_{\text{eff}} - 3}$ formula. This is a user-supplied sensitivity parameter, not an estimated effective sample size. It is consulted for method %in% c("fisher-z", "wald") and for explicit Wald fallback rows when a requested bootstrap tier is not yet routed. The default heuristic uses fit\$n_sites for tier "B" ("unit"), fit\$n_site_species for "W" ("unit_obs"), fit\$n_species for "phy", and fit\$n_sites elsewhere. These counts are only transparent heuristics: grouped, phylogenetic, spatial, latent, and non-Gaussian structure can make the iid Fisher variance inappropriate. If the automatic tier count is missing or below 4, Fisher-z rows return point correlations with unavailable interval bounds rather than substituting an arbitrary sample size. Set n_eff only when the analysis supplies a defensible target-specific rationale, and report that rationale.
nsim	Number of bootstrap replicates when method = "bootstrap". Default 500.
seed	Optional RNG seed for the bootstrap.
link_residual	How to treat the family-specific link-residual variance on the diagonal of the implied Σ before computing correlations: "auto" (default) For non-Gaussian fits, add the link-specific implicit residual (e.g. $\pi^2/3$ for binomial-logit; 1 for probit; trigamma() terms for Gamma / NB2 / Beta / etc.) to the diagonal before computing correlations. Returned correlations are on the latent-liability scale; this is the convention most readers expect. Gaussian fits are unaffected (link residual is 0). "none" Use the fitted model-implied Σ directly with no link-residual addition. For ordinary latent() fits this includes the default diagonal Psi companion; correlations come out on the model-implied scale without the family adjustment. Behaviour change in this release: the previous version hardcoded link_residual = "none". Non-Gaussian callers who relied on that behaviour will see different correlation values under the new default. A one-shot warning fires the first time per session that a non-Gaussian fit is processed without an explicit link_residual argument. Pass link_residual = "auto" to suppress the warning and lock the new behaviour, or link_residual = "none" to restore the previous behaviour.

Details

- "none" (default): point correlations only. No universal interval calibration has been established across covariance tiers, families, and targets.

- "fisher-z": Fisher's z-transform heuristic interval. Computes $\hat{z} = \text{atanh}(\hat{\rho})$, $\widehat{\text{SE}}(\hat{z}) = 1/\sqrt{n_{\text{eff}} - 3}$, constructs the CI on z, then back-transforms via $\tanh(\cdot)$ (so bounds are guaranteed inside $[-1, 1]$). Fast (seconds for any T), but the classical iid-correlation variance is not a calibrated mixed-model standard error. Treat these bounds as a sensitivity display and see `n_eff`.
- "profile": withdrawn. The nonlinear penalty-profile prototype did not yet provide a sufficiently strict constraint and constrained-optimizer diagnostic contract. This token stops with an explanation rather than returning bounds.
- "wald": backward-compat alias of "fisher-z" with the same numerics. Emits a one-shot inform pointing at the canonical name. Kept for scripts that filter on `method == "wald"`.
- "bootstrap": parametric bootstrap via [bootstrap_Sigma](#). Slowest (full sampling distribution); use when a fitted model gives useful point estimates but Hessian- or profile-based intervals are not the right uncertainty summary. Structured tiers not yet resampled by [bootstrap_Sigma](#), currently the SPDE spatial tier, return an explicit Wald/Fisher-z fallback with a message rather than fake bootstrap support.

For T traits at one tier, there are $T(T-1)/2$ unique correlations. A fit with T = 6 and four covariance levels present has up to 60 cross-trait correlations to report.

Value

A data frame (tibble-like) with columns:

`tier` Character level label. The current output stores internal labels "B", "W", "phy", and "spde"; use "unit" and "unit_obs" as input names in new calls.

`trait_i, trait_j` Trait names with $i < j$.

`correlation` Point estimate.

`lower, upper` Confidence-interval bounds.

`method` Method used to compute the CI.

`interval_status` Claim-boundary marker: "none" for point-only output, "heuristic_unvalidated" for Fisher-z/Wald bounds, and "target_specific_uncalibrated" for bootstrap bounds. The bootstrap route computes intervals, but this function does not certify their frequentist coverage for the fitted target.

For an engine = "julia" bridge fit, lower/upper are NA, method = "none", and interval_status = "none".

Caveats

- The former nonlinear penalty-profile route is withheld pending an exact constraint solver and explicit constrained-fit diagnostics.
- Bootstrap uses [bootstrap_Sigma](#) refits and is the practical fallback when point estimates are useful but Hessian- or profile-based intervals are unavailable. Inspect bootstrap warnings, failed replicates, and interval width before treating the intervals as final.

See Also

[extract_Sigma](#), [bootstrap_Sigma](#), [confint.gllvmTMB_multi](#), [extract_communality](#).

Examples

```
set.seed(1)
s <- simulate_site_trait(
  n_sites = 80, n_species = 6, n_traits = 4,
  mean_species_per_site = 4,
  Lambda_B = matrix(c(0.9, 0.4, -0.3, 0.5), 4, 1),
  psi_B = c(0.4, 0.3, 0.5, 0.2)
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 1),
  data = s$data,
  trait = "trait",
  unit = "site"
)
## Default: point correlations only.
cors <- extract_correlations(fit, tier = "unit")
## Opt-in Fisher-z sensitivity bounds with an explicitly justified n_eff.
cors2 <- extract_correlations(fit, tier = "unit", method = "fisher-z",
                             n_eff = 60L)
## Fifty refits illustrate the API, rather than establish coverage.
## Inspect failed refits and increase B for a scientific analysis.
cors_b <- extract_correlations(fit, tier = "unit", method = "bootstrap",
                              nsim = 50, seed = 42)
```

extract_cross_correlations

Cross-family correlations between a nominal (multinomial) trait and partners

Description

For a fit where a `multinomial()` trait shares a latent factor with other-family traits, report the association between the nominal trait and each partner trait. A nominal trait spans K-1 baseline-category contrasts, so its association with a single-scale partner is a *vector*, summarized two ways:

- `multiple_r`: the reference-invariant multiple correlation $R = \sqrt{\Sigma_{pc} \Sigma_{cc}^{-1} \Sigma_{cp} / \sigma_{pp}}$ between partner `p` and the whole K-1 contrast block `c`. Invariant to the baseline category; magnitude in $[0, 1]$.
- `contrast_r` (when `contrasts = TRUE`): the (K-1)-vector of individual contrast correlations, labelled by category-vs-baseline.

Usage

```
extract_cross_correlations(
  fit,
  level = "unit",
  contrasts = FALSE,
```

```

link_residual = c("auto", "none"),
method = c("point", "bootstrap", "profile", "wald"),
conf = 0.95,
nsim = 500L,
seed = NULL
)

```

Arguments

fit	a fitted <code>gllvmTMB_multi</code> with one <code>multinomial()</code> trait and a shared ordinary unit tier (<code>latent(0 + trait unit, d)</code>).
level	covariance tier. Only the ordinary "unit" tier is admitted.
contrasts	if TRUE, also return the per-contrast (K-1)-vector (as a list column).
link_residual	passed to <code>extract_Sigma()</code> ; "auto" (default) puts the nominal block on the observation scale via the $(\pi^2/6)(I + J)$ softmax residual, making <code>multiple_r</code> commensurable with single-scale partners; "none" uses the latent (loadings) scale. Note: with "none" and a loadings-only fit whose shared factor the contrast block spans (e.g. the block dimension $\geq$ the partner's), <code>multiple_r</code> degenerates to 1 (the partner is then a deterministic function of the block). This is correct but uninformative, which is why "auto" (full-rank via the residual) is the sensible default for the single-number summary; read the per-pair latent correlations from <code>extract_Sigma()</code> / <code>extract_correlations()</code> instead. An ordinal-probit partner is not admitted with "auto": its latent residual is already fixed at 1 by the threshold model, so adding another unit residual would double-count it. Use a supported non-ordinal partner set for the one-number summary, or report latent-scale pairwise quantities with <code>extract_Sigma(..., part = "shared", link_residual = "none")</code> .
method	one of "point" (default; point estimates only, no interval columns), "wald" (fast Fisher-z intervals on BOTH <code>multiple_r</code> and <code>contrast_r</code> ; no refits or root-finding; heuristic bounds can be unavailable when the effective sample size is too small or an estimate is on a boundary), "bootstrap" (parametric-bootstrap interval on the aggregate <code>multiple_r</code> via <code>bootstrap_Sigma()</code>), or "profile". The profile token is retained for compatibility but is withdrawn: it always stops with a typed error rather than returning an interval. Use point estimates, "wald", or "bootstrap" instead. None of the available intervals is coverage-calibrated yet — they are recovery-oriented; "wald" is the most approximate (<code>multiple_r</code> is not a Pearson correlation). Use them to explore + report bugs, not for inference.
conf	confidence level for the interval columns. Default 0.95.
nsim	number of parametric-bootstrap replicates when <code>method = "bootstrap"</code> . Default 500.
seed	optional RNG seed for the bootstrap.

Value

a data.frame, one row per (nominal, partner) pair. With `method = "point"` the columns are nominal, partner, `multiple_r` (and `contrast_r` when `contrasts = TRUE`). With `method = "wald", "bootstrap"`,

per-estimand interval columns are added: `multiple_r_lower/multiple_r_upper/multiple_r_method/multiple_r_interval` (scalar), and when `contrasts = TRUE` `contrast_r_lower/contrast_r_upper` (list columns) plus `contrast_r_method/contrast_r_interval_status`. The estimand NOT served by the requested method keeps point-only interval columns (NA bounds, `method = "point"`). Computed intervals carry an `interval_status` flag ("heuristic_unvalidated" for "wald", "target_specific_uncalibrated" for "bootstrap") – coverage is not yet certified for any route (recovery evidence for this route is not yet complete).

Examples

```
set.seed(12)
n_unit <- 120L
n_rep <- 3L
unit_id <- rep(seq_len(n_unit), each = n_rep)
score <- stats::rnorm(n_unit)
score_obs <- score[unit_id]
gaussian_y <- 0.8 * score_obs + stats::rnorm(length(unit_id), sd = 0.4)
category_y <- vapply(seq_along(unit_id), function(i) {
  eta <- c(0, 0.9 * score_obs[i], -0.6 * score_obs[i])
  prob <- exp(eta - max(eta))
  sample.int(3L, 1L, prob = prob / sum(prob))
}, integer(1L))
dat <- data.frame(
  unit = factor(c(unit_id, unit_id)),
  trait = factor(rep(c("g", "cat"), each = length(unit_id)),
    levels = c("g", "cat")),
  family = factor(rep(c("g", "m"), each = length(unit_id)),
    levels = c("g", "m")),
  value = c(gaussian_y, category_y)
)
families <- list(g = gaussian(), m = multinomial())
attr(families, "family_var") <- "family"
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | unit, d = 1, unique = TRUE),
  data = dat, family = families, trait = "trait", unit = "unit"
)
extract_cross_correlations(fit, contrasts = TRUE)
```

extract_cutpoints

Extract ordinal-probit cutpoints from a fitted gllvmTMB model

Description

For traits fitted with `ordinal_probit()`, returns a tidy data frame with the $K - 2$ estimated cutpoints $\tau_2, \dots, \tau_{K-1}$ per trait, with optional standard errors from the joint sdreport.

Usage

```
extract_cutpoints(fit, quiet = FALSE)
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> with at least one <code>ordinal_probit()</code> trait.
<code>quiet</code>	Suppress the note emitted when <code>tau_se</code> cannot be filled in because the fit has no <code>sd_report</code> . Internal callers that display only the cutpoint <i>estimates</i> (<code>print()</code> , <code>tidy()</code>) pass <code>TRUE</code> : a note explaining an absent standard-error column is noise when no such column is shown.

Details

Convention: `gllvmTMB` follows Hadfield (2015) — $\tau_1 = 0$ is fixed for identifiability and the $K - 2$ free cutpoints are reported as `cutpoint_2`, `cutpoint_3`, etc. This differs from `brms`, which reports $K - 1$ cutpoints as `Intercept[1..K-1]`.

This extractor includes Wald standard errors, so it refuses fits made with non-unit likelihood weights. The fitted cutpoint parameters remain point estimates, but weighted-objective uncertainty is not calibrated.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(
  value ~ 0 + trait + phylo_indep(species, tree = tree),
  data   = ordinal_dat,
  trait  = "trait",
  unit   = "individual",
  cluster = "species",
  family = ordinal_probit()
)
extract_cutpoints(fit)
```

Value

A data frame with columns

`trait` Trait label (factor level from `data[[trait]]`).

`cutpoint_index` Integer index $k \in \{2, \dots, K - 1\}$.

`cutpoint_label` Character label "`cutpoint_<k>`".

`tau_estimate` Estimated τ_k on the latent (probit) scale.

`tau_se` Standard error from the joint `sdreport`, or NA if the report is unavailable.

If the fit contains no `ordinal_probit()` traits, returns a zero-row data frame with the same columns.

References

Hadfield, J. D. (2015). Increasing the efficiency of MCMC for hierarchical phylogenetic models of categorical traits using reduced mixed models. *Methods Ecol. Evol.* 6:706-714. doi:10.1111/2041210X.12354

See Also

[ordinal_probit\(\)](#) for the family constructor and the threshold-trait theory reference list.

extract_Gamma

Extract a cross-lineage Gamma block

Description

`extract_Gamma()` slices the shared covariance matrix returned by [extract_Sigma\(\)](#) to give the row-lineage by column-lineage block $\text{Gamma_shape} = \text{Lambda_row} \text{Lambda_col}^T$. In a fitted cross-lineage kernel model, `level` is the named `kernel_*`() tier, `row_traits` are the host traits, and `col_traits` are the partner traits. This is a point-estimate extractor for the shared covariance block of a fitted dense-kernel model, covered by the package's validation tests at the fixed-rho scale: when the fitted kernel tier was built from [make_cross_kernel\(\)](#), `scale = "effect"` returns $\text{Gamma_effect} = \rho * \text{Gamma_shape}$ using the fixed ρ recorded on that kernel. Uncertainty for Gamma is not yet reported by this helper; use bootstrap workflows when interval estimates are required. Use [profile_cross_rho\(\)](#) to compare a fixed ρ grid; in-engine ρ estimation and profile intervals remain planned work.

Usage

```
extract_Gamma(fit, level, row_traits, col_traits, scale = c("shape", "effect"))
```

Arguments

<code>fit</code>	A fitted <code>gllvmTMB_multi</code> object.
<code>level</code>	Character scalar naming the covariance tier. For <code>kernel_*</code> () fits this is the name argument supplied in the formula.
<code>row_traits, col_traits</code>	Character vectors of trait names defining the rows and columns of the returned block.
<code>scale</code>	Character scalar. "shape" (default) returns $\text{Gamma_shape} = \text{Lambda_row} \text{Lambda_col}^T$. "effect" returns $\rho * \text{Gamma_shape}$ for fixed cross-lineage kernels built by make_cross_kernel() . The current engine does not estimate ρ .

Details

ρ is part of the supplied K matrix, not a fitted parameter in the current engine. To profile it, use [profile_cross_rho\(\)](#) to rebuild K_{star} over a small ρ grid, refit the same formula, and compare `logLik()` values. `scale = "effect"` is therefore a fixed-kernel transformation, not an estimate of ρ . Treat Gamma from a single association matrix W as data-condition sensitive: the C2 recovery test includes a sparse-versus-dense W check, and sparse or poorly replicated host-partner links should be reported as weaker evidence rather than as a precise coevolution estimate.

For fixed multi-kernel fits, `extract_Gamma()` also inspects the fitted kernel-similarity diagnostics. If the requested component participates in a high-overlap kernel pair, the returned block is still available, but a warning reminds the caller that component-specific separation is weak evidence.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
Gamma_HP <- extract_Gamma(
  fit,
  level = "cross",
  row_traits = c("host_size", "host_defence"),
  col_traits = c("partner_size", "partner_attack")
)
```

Value

A numeric matrix with rows `row_traits` and columns `col_traits`.

<code>extract_loadings</code>	<i>Extract the trait loading matrix</i>
-------------------------------	---

Description

Returns the trait loading matrix from a fit returned by `gllvmTMB()`. This is the canonical `snake_case` accessor in the `extract_*`() family; `getLoadings()` is an accepted compatibility spelling for readers coming from `gllvm`. Both return the same matrix.

Usage

```
extract_loadings(fit, level = "unit", rotate = c("none", "varimax", "promax"))
```

Arguments

<code>fit</code>	A fitted multivariate model returned by <code>gllvmTMB()</code> . Admitted engine = "julia" bridge fits expose raw unit-tier loadings and scores; rotated ordinations remain gated for Julia bridge fits.
<code>level</code>	"unit" (between-unit) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.
<code>rotate</code>	Optional "varimax" or "promax" rotation after fitting. Default "none" returns the engine's native lower-triangular Lambda. For Julia bridge fits only "none" is currently routed.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
extract_loadings(fit, level = "unit", rotate = "varimax")
```

Value

An `n_traits` x `d` numeric matrix.

See Also

`extract_ordination()` for scores and loadings together; `getLoadings()` for the `gllvm`-style spelling.

extract_lv_effects	<i>Predictor effects on latent-score means</i>
--------------------	--

Description

For a predictor-informed ordinary latent fit, `latent(..., lv = ~ x)` uses the unit-level score model

$$\mathbf{u}_s = \mathbf{z}_s + \mathbf{X}_s \boldsymbol{\alpha},$$

where the innovation $\mathbf{z}_s$ keeps the usual standard-normal prior. `extract_lv_effects()` reports either the raw axis-scale $\boldsymbol{\alpha}$ coefficients or the induced trait-scale contribution $\mathbf{B}_{lv} = \mathbf{A} \boldsymbol{\alpha}^\top$. The axis-scale table is the default because it matches the usual constrained latent-variable / ordination coefficient. It is conditional on the fitted loading constraint and axis orientation: like the loadings themselves (`extract_loadings()`), $\boldsymbol{\alpha}$ is identified only up to the same orthogonal (or oblique) rotation, so raw axis-scale coefficients from two fits – even of the same model on the same data – need not sit on a common axis and are not directly comparable without alignment. This is intrinsic to latent-variable identifiability, not a defect. For **cross-fit comparison**, prefer `type = "trait_effect"`, the rotation-invariant induced slope surface on the trait linear-predictor scale, described next.

Usage

```
extract_lv_effects(
  fit,
  level = "unit",
  type = c("axis_effect", "trait_effect"),
  conf.level = 0.95,
  method = c("wald", "profile", "bootstrap"),
  ...
)
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> .
<code>level</code>	Currently "unit" only. Legacy alias "B" is accepted.
<code>type</code>	"axis_effect" returns raw $\boldsymbol{\alpha}$ coefficients on the latent-axis scale. "trait_effect" returns $\mathbf{B}_{lv}$ on the trait linear-predictor scale.
<code>conf.level</code>	Confidence level for the interval (Wald, or the confidence level passed to the profile / bootstrap CI).
<code>method</code>	Interval method. "wald" is the only current public route. The nonlinear profile and predictor-effect bootstrap prototypes are withheld until constrained-fit diagnostics, failed-refit accounting, and repeated-sampling coverage evidence are available.
<code>...</code>	Reserved for future interval routes; currently unused.

Details

If the axis-scale representation is specifically required and cross-fit alignment matters, apply the same rotation used for the loadings: the mean latent score is $\mathbf{X}_s\alpha$, so it rotates exactly as the scores in `rotate_loadings()` do $-\alpha\mathbf{T}$ for the rotation matrix $\mathbf{T}$ returned by `rotate_loadings(fit, level, method)$T` (verified to reconstruct the rotated mean score to machine precision). There is no separate exported helper that performs this composition; it must be assembled from `rotate_loadings()`'s $\mathbf{T}$ and the estimate column here.

For native TMB fits, `std.error` is populated from a positive-definite `sdreport()` when available. Axis-effect SEs come from the fixed-parameter block for `alpha_lv_B`; trait-effect SEs come from TMB's delta-method `ADREPORT(B_lv_unit)` output. `lower` and `upper` are Wald intervals using `conf.level`. Broad repeated-sampling coverage has not been established, so finite bounds remain experimental rather than nominally calibrated. For Gaussian, Poisson, NB2, Gamma, Beta, and binomial logit/probit/cloglog engine = "julia" bridge fits, `ci_method = "none"` exposes point estimates only (`std.error`, `lower`, and `upper` are NA). When the bridge supplies a retained delta-method uncertainty data, `extract_lv_effects()` surfaces finite `std.error`, `lower`, and `upper`. Those Julia bridge values are not coverage-calibrated intervals.

Value

A data frame. For `type = "axis_effect"`, columns are `level`, `axis`, `predictor`, `estimate`, `std.error`, `lower`, `upper`, `rotation_status` and `uncertainty_status`. For `type = "trait_effect"`, columns are `level`, `trait`, `predictor`, `estimate`, `std.error`, `lower`, `upper`, and `uncertainty_status`. The `uncertainty_status` field states why standard errors are unavailable (for example, skipped `sdreport()`, a non-positive-definite Hessian, or non-finite standard errors) or labels finite Wald bounds as not yet coverage-calibrated.

See Also

[extract_ordination\(\)](#)

extract_Omega

Total trait covariance Omega summed across requested tiers

Description

Sum of Σ_{tier} matrices across the user- selected tiers. The canonical PGLLVM use case is the species-level three-piece decomposition:

$$\Omega = \Sigma_{\text{phy}} + \Sigma_{\text{non,shared}} + \Psi_{\text{non}}.$$

Here `tiers = c("phy", "B")` returns $\Sigma_{\text{phy}} + \Sigma_B$ where $\Sigma_B = \Lambda_B \Lambda_B^\top + \Psi_B$ aggregates the non-phylogenetic shared and unique components.

Usage

```
extract_Omega(fit, tiers = NULL, link_residual = c("auto", "none"))
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> .
<code>tiers</code>	Character vector. Subset of <code>c("B", "W", "phy")</code> . Default NULL auto-detects: includes "phy" if <code>phylo_latent()</code> is in the formula, "B" if any <code>latent()/indep()</code> at unit, "W" if any <code>latent()/indep()</code> at unit_obs.
<code>link_residual</code>	For non-Gaussian fits: "auto" (default) adds a per-trait link-specific implicit residual to the diagonal of the summed Omega (once, not per tier — see "Family-aware link residuals" in <code>extract_Sigma()</code>); mixed-family fits get the residual implied by each trait's family/link. "none" returns the summed fitted covariance without link-residual additions. Gaussian / lognormal-only fits are unaffected.

Details

For two-level behavioural-syndrome fits, `tiers = c("B", "W")` returns the **phenotypic** trait covariance $\Sigma_P = \Sigma_B + \Sigma_W$.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
# PGLLMV three-piece total
om <- extract_Omega(fit, tiers = c("phy", "B"))
round(om$R_Omega, 2)

# Phenotypic covariance from a two-level behavioural-syndromes fit
om2 <- extract_Omega(fit, tiers = c("B", "W"))
```

Value

A list with Omega ($T \times T$ summed covariance), R_Omega (correlation), tiers_used (which tiers were actually summed), note (notes from each underlying `extract_Sigma()` call), and (when `link_residual = "auto"`) `residual_split` — the per-trait $\sigma_d^2/\sigma_e^2/\sigma_{\text{total}}^2$ data frame from `extract_residual_split()`.

See Also

`extract_Sigma()`; `extract_phylo_signal()`; `extract_proportions()`; `extract_residual_split()`.

<code>extract_ordination</code>	<i>Extract ordination scores and loadings from a fitted multivariate model</i>
---------------------------------	--

Description

Extract ordination scores and loadings from a fitted multivariate model

Usage

```
extract_ordination(
  fit,
  level = "unit",
  component = c("total", "innovation", "mean")
)
```

Arguments

fit	A fitted multivariate model returned by <code>gllvmTMB()</code> . Admitted engine = "julia" bridge fits expose raw unit-tier loadings and scores. Gaussian bridge fits with <code>latent(..., lv = ~ x)</code> also expose retained "mean" and "innovation" score components. Within-unit, structured-tier, and rotated ordinations remain gated for Julia bridge extractors. A <code>gllvmTMB_va</code> (integration = "va") fit exposes level = "unit" loadings and scores only, with generic trait/unit row and column names synthesised (a variational fit carries no stored data or column names); level = "unit_obs" returns NULL because the variational route fits no within-unit tier.
level	"unit" (between-unit) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.
component	Score component to return. "total" returns the latent score entering the linear predictor. "innovation" returns the zero-mean latent innovation. "mean" returns the predictor-informed score mean and is non-zero only for <code>latent(..., lv = ~ x)</code> fits.

Value

A list with scores (units or within-unit observations in rows, latent axes in columns) and loadings (traits in rows, axes in columns).

Examples

```
sim <- simulate_site_trait(
  n_sites = 20, n_species = 6, n_traits = 4,
  mean_species_per_site = 4, seed = 1
)
fit <- gllvmTMB(
  value ~ 0 + trait +
    latent(0 + trait | site, d = 2),
  data = sim$data,
  trait = "trait",
  unit = "site"
)
ord <- extract_ordination(fit, level = "unit")
head(ord$scores)
ord$loadings
```

extract_phylo_signal *Phylogenetic-signal proportions per trait*

Description

For a phylogenetic species-level GLLVM, decomposes each trait's between-species latent variance into three additive components that sum to one:

$$H_t^2 = \frac{[\Sigma_{\text{phy}}]_{tt}}{V_{\eta,t}}, \quad C_{\text{non},t}^2 = \frac{[\Sigma_{\text{non,shared}}]_{tt}}{V_{\eta,t}}, \quad \psi_t = \frac{[\Psi_{\text{non}}]_{tt}}{V_{\eta,t}},$$

where $V_{\eta,t} = [\Sigma_{\text{phy}}]_{tt} + [\Sigma_{\text{non,shared}}]_{tt} + [\Psi_{\text{non}}]_{tt}$ is the total between-species latent variance for trait t .

Usage

```
extract_phylo_signal(
  fit,
  ci = FALSE,
  conf_level = 0.95,
  method = c("profile", "wald", "bootstrap"),
  nsim = 500L,
  seed = NULL,
  link_residual = c("none", "auto")
)
```

Arguments

fit	A fit returned by <code>gllvmTMB()</code> with a <code>phylo_latent()</code> term.
ci	Logical. When TRUE, adds confidence-interval columns to the output for the H^2 column. Default FALSE for backward compatibility.
conf_level	Confidence level when <code>ci = TRUE</code> . Default 0.95.
method	One of "profile" (default), "wald", "bootstrap". Only used when <code>ci = TRUE</code> . For 2-component decompositions (a phylo diagonal vs species-level diagonal component only) profile uses a linear contrast; for 3-component decompositions (PGLLVM with <code>phylo_latent</code> plus a species-level latent decomposition with <code>Psi</code>) the full profile path is not yet implemented and falls back to numerical delta-method Wald bounds labelled "wald(numeric)".
nsim	Number of bootstrap replicates when <code>method = "bootstrap"</code> . Default 500.
seed	Optional RNG seed for the bootstrap.
link_residual	"none" (default) keeps the historical species-level-latent denominator. "auto" adds each trait's distribution-specific link-scale residual σ_d^2 to the denominator and reports it as a fourth <code>link_residual</code> proportion column — the liability-scale phylogenetic heritability for threshold families (see the dedicated section below). Not yet compatible with <code>ci = TRUE</code> .

Details

Interpretation:

H_t^2 phylogenetic signal — proportion of between- species latent variance attributable to phylogenetically structured variation ("evolutionary conservatism"). When the model uses the folded `phylo_latent(..., unique = TRUE)` decomposition, Σ_{phy} is the sum $\Lambda_{\text{phy}}\Lambda_{\text{phy}}^\top + \Psi_{\text{phy}}$ and H_t^2 reflects the *total* phylogenetic variance.

$C_{\text{non},t}^2$ non-phylogenetic communality — proportion of variance attributable to shared non-phylogenetic axes ("coordinated tip-level lability" across traits).

ψ_t uniqueness — proportion of variance not captured by any shared axis ("relative modularity").

Requires `phylo_latent()` (optionally with `unique = TRUE` for a phylogenetic Psi) plus a species-level `latent()` term, which carries its diagonal Psi companion by default. If the species-tier Psi is absent (a `latent(..., unique = FALSE)` subset), $\psi_t = 0$ for all traits and a `cli::cli_inform()` advisory fires.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(
  value ~ 0 + trait + phylo_latent(species, d = 2) +
                        latent(0 + trait | species, d = 2) +
                        indep(0 + trait | species),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_tree = tree
)
extract_phylo_signal(fit)
```

Value

A data frame with columns `trait`, `H2`, `C2_non`, `Psi`, `V_eta` (the denominator), one row per trait. The three proportions sum to 1.0 by construction. With `link_residual = "auto"` an extra `link_residual` proportion column is inserted before `V_eta`, the four proportions sum to 1.0, and `V_eta` includes σ_d^2 . When `ci = TRUE`, three additional columns are added: `H2_lower`, `H2_upper`, `H2_method`.

Interval calibration

The point estimates are the supported claim. When `ci = TRUE`, the interval methods are provided for exploration: their empirical coverage is not certified for this estimand, so treat the intervals as exploratory rather than coverage-calibrated. See `NEWS.md` for the current coverage status.

Which variances enter the denominator

`V_eta` is the **species-level** latent variance: only components whose grouping is the `cluster` column contribute. Concretely $V_\eta = \sigma_{\text{phy}}^2 + \sigma_{\text{non}}^2$, where σ_{non}^2 collects the *species-grouped* non-phylogenetic variance.

- When `unit == cluster` (the usual `unit = "species"` setup) the ordinary `latent()` term and its `Psi` companion are species-level, so both enter.
- When `unit != cluster` (a crossed site x species design) `Lambda_B` and its `Psi` are **site**-level and do **not** enter; the species-level non-phylogenetic variance is the cluster-tier `q_it` term (`indep(0 + trait | <cluster>)`), reported as `sd_q^2`).

The denominator is defined at the species level. **Other defensible definitions exist** — for instance a *total*-variance denominator that also absorbs site-level variation, which would give a smaller H^2 for the same fit. Before 2026-07-08 this function used the unit tier unconditionally, which silently reported $H^2 = 1$ for every trait in a crossed design because the `q_it` variance was never read. Compare `extract_proportions()` if you want every component reported separately rather than folded into a species-level ratio.

Liability-scale heritability for categorical families

With the default `link_residual = "none"` the denominator contains only the *estimated* species-level components, so a phylogenetic-only fit reports $H^2 = 1$ for every trait. For threshold/liability families the conventional phylogenetic heritability (Mizuno et al. 2025 J. Evol. Biol., eq 4/18/19; de Villemereuil & Nakagawa 2014) instead includes the distribution-specific latent residual σ_d^2 in the denominator. Request it with `link_residual = "auto"`, which adds each trait's fixed link-scale residual (see `extract_Sigma()`'s "Family-aware link residuals" table) as a fourth component:

- `ordinal_probit()` (and binomial probit): $\sigma_d^2 = 1$ exactly, so a phylogenetic-only fit reports the paper's $H^2 = V_a / (V_a + 1)$ per trait.
- `multinomial()`: each of the $K-1$ baseline-category contrasts is a logit, so $\sigma_d^2 = \pi^2/3$ per *contrast* and a phylogenetic-only fit reports $H_{(k)}^2 = V_{a(k)} / (V_{a(k)} + \pi^2/3)$ for contrast k . **Contrast heritabilities are baseline-referenced:** the softmax link residual is the full matrix $(\pi^2/6)(\mathbf{I} + \mathbf{J})$ — the shared baseline category couples the contrasts with $\pi^2/6$ off-diagonals — so each $H_{(k)}^2$ is the heritability of the "category k vs baseline" liability under that baseline choice, not of "the trait". Do **not** average or otherwise collapse the per-contrast values into a single scalar heritability; changing the baseline changes the contrasts.

When a categorical family (fid 14/16) is present and `link_residual = "none"`, an advisory fires pointing at `link_residual = "auto"`; the default is unchanged for backward compatibility. `ci = TRUE` is not yet implemented for the liability-scale denominator and refuses with a typed error.

See Also

`extract_Sigma()`; `extract_Omega()`; `extract_proportions()`; `extract_repeatability()`; `extract_communality()`; `extract_correlations()`.

<code>extract_proportions</code>	<i>Per-trait proportion-of-variance decomposition across all model components</i>
----------------------------------	---

Description

For each trait, returns the proportion of total latent variance attributable to each component present in the model: shared (rr) and unique (diag) at each tier (unit, unit_obs, phy), diagonal cluster / cluster2 components when present, plus optionally the binomial link's implicit residual.

Usage

```
extract_proportions(
  fit,
  link_residual = c("auto", "none"),
  format = c("long", "wide")
)
```

Arguments

fit	A fit returned by gllvmTMB() .
link_residual	For non-Gaussian fits: "auto" (default) adds the link's implicit residual as its own per-trait component (e.g. $\pi^2/3$ for binomial logit, $\log(1 + 1/\hat{\mu}_t)$ for Poisson log; see "Family-aware link residuals" in extract_Sigma() for the full per-family table). Mixed-family fits get the residual implied by each trait's family/link. "none" omits the link-implicit component.
format	"long" (default) returns a tibble-like data frame with one row per (trait, component); "wide" returns one row per trait with one column per component.

Details

This is the most general proportion-decomposition function in the package. [extract_phylo_signal\(\)](#) is the focused PGLLVM $H^2/C_{\text{non}}^2/\psi^2$ convenience wrapper for the species-level case; [extract_community\(\)](#) gives the rr-only "shared" proportion at one tier; [extract_ICC_site\(\)](#) gives the between-vs-within proportion. [extract_proportions\(\)](#) returns all of these in one tidy frame.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
extract_proportions(fit) # long format
extract_proportions(fit, format = "wide") # one row per trait
```

Value

Long format: data frame with columns trait, component (e.g. "shared_unit", "unique_unit", "unique_cluster", "shared_phy", "link_residual"), variance (the absolute variance), proportion (the share of the total). Wide format: data frame with one column per component plus a total_variance column; the per-trait proportions sum to 1.

OLRE interpretation: for fits with a genuine observation-level diagonal term, usually written as per-row indep() in new code (see [extract_residual_split\(\)](#)), the historical unique_W component in this output corresponds to σ_e^2 (the estimated OLRE variance) and the link_residual component corresponds to σ_d^2 (the distribution-specific latent residual).

See Also

`extract_phylo_signal()` — the PGLLVM-specific shortcut; `extract_communality()`; `extract_ICC_site()`; `extract_residual_split()` — explicit σ_d^2/σ_e^2 decomposition for OLRE fits.

`extract_repeatability` *Per-trait repeatability with confidence intervals*

Description

Returns the per-trait repeatability

$$R_t = v_{B,t} / (v_{B,t} + v_{W,t}),$$

where

$$v_{B,t} = [\Lambda_B \Lambda_B^\top]_{tt} + \sigma_{B,t}^2$$

and

$$v_{W,t} = [\Lambda_W \Lambda_W^\top]_{tt} + \sigma_{W,t}^2 + \sigma_{d,t}^2.$$

The first two terms include the shared latent and diagonal companion variance at the unit and observation tiers; $\sigma_{d,t}^2$ is the family-specific link residual used for non-Gaussian traits. The function is intended for a fit returned by `gllvmTMB()` with the relevant ordinary unit and observation-level components. Also known as the intraclass correlation coefficient (ICC) at the unit level.

Usage

```
extract_repeatability(
  fit,
  level = 0.95,
  method = c("wald", "profile", "bootstrap"),
  nsim = 500L,
  seed = NULL
)
```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB</code> . A <code>bootstrap_Sigma</code> object is also accepted when it contains an <code>ICC_site</code> summary; in that case the function reuses the stored point estimates and percentile bounds rather than refitting.
<code>level</code>	Confidence level. Default 0.95.
<code>method</code>	One of "wald" (default), "profile", or "bootstrap". "profile" is accepted only for backwards compatibility and aborts because a defensible profile interval for canonical full-covariance repeatability is not available.
<code>nsim</code>	Number of bootstrap replicates when <code>method = "bootstrap"</code> . Default 500.
<code>seed</code>	Optional RNG seed for the bootstrap.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(
  value ~ 0 + trait +
    latent(0 + trait | site, d = 1) +
    latent(0 + trait | site_species, d = 1),
  data      = df,
  trait     = "trait",
  unit      = "site",
  unit_obs  = "site_species"
)
extract_repeatability(fit)
boot <- bootstrap_Sigma(fit, n_boot = 50, level = c("unit", "unit_obs"),
  what = "ICC", progress = FALSE)
extract_repeatability(boot)
```

Value

A data frame with columns trait, R (point estimate), lower, upper, method.

Interval calibration

The point estimates are the supported claim. The Wald and bootstrap interval methods here are provided for exploration: their empirical coverage is not certified for this estimand, so treat intervals as exploratory rather than coverage-calibrated. See `NEWS.md` for the current coverage status.

Method choice

- "wald" (default): Gaussian-approximation CI via the delta method on $\log(v_B) - \log(v_W)$, transformed with `plogis()`.
- "profile": withdrawn. It aborts rather than silently substituting a different interval method or estimand.
- "bootstrap": parametric bootstrap via `bootstrap_Sigma()`.

References

Nakagawa, S. & Schielzeth, H. (2010) Repeatability for Gaussian and non-Gaussian data: a practical guide for biologists. *Biological Reviews* **85**, 935-956. doi:10.1111/j.1469185X.2010.00141.x

See Also

[extract_communality](#), [extract_correlations](#), [extract_phylo_signal](#), [confint.gllvmTMB_multi](#).

extract_residual_cov *Implied trait covariance or correlation*

Description

Canonical snake_case accessors in the extract_*() family for the implied trait covariance and correlation at level = "unit" or "unit_obs". [getResidualCov\(\)](#) and [getResidualCor\(\)](#) are accepted gllvm-compatibility spellings; each pair returns the same matrix.

Usage

```
extract_residual_cov(fit, level = "unit")
```

```
extract_residual_cor(fit, level = "unit")
```

Arguments

fit	A fitted multivariate model returned by gllvmTMB() . Admitted engine = "julia" bridge fits expose raw unit-tier loadings and scores; rotated ordinations remain gated for Julia bridge fits.
level	"unit" (between-unit) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.

Value

An n_traits x n_traits matrix.

See Also

[extract_Sigma\(\)](#) for the unified Sigma API; [getResidualCov\(\)](#) for the gllvm-style spellings.

extract_rotated_loadings_table
Extract a tidy rotated loading table

Description

Returns one row per trait and latent axis from a fitted [gllvmTMB\(\)](#) model. This is the report-ready companion to [rotate_loadings\(\)](#) and to `plot(type = "ordination")`: the same rotation, axis-ordering, sign-anchor, and optional loading-standardization conventions are recorded in columns instead of left inside a matrix.

Usage

```
extract_rotated_loadings_table(
  fit,
  level = "unit",
  method = c("varimax", "promax", "none"),
  order_axes = TRUE,
  sign_anchor = c("auto", "none"),
  anchor_traits = NULL,
  loading_scale = c("raw", "standardized")
)
```

Arguments

<code>fit</code>	A fitted multivariate model returned by <code>gllvmTMB()</code> .
<code>level</code>	"unit" (between-unit) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.
<code>method</code>	One of "varimax", "promax", or "none"; passed to <code>rotate_loadings()</code> .
<code>order_axes</code>	Logical. When TRUE, reorder rotated axes by decreasing raw shared variance before tabulation. Ignored when <code>method = "none"</code> .
<code>sign_anchor</code>	One of "auto" or "none". "auto" flips each rotated axis so its anchor trait has a positive loading. Ignored when <code>method = "none"</code> .
<code>anchor_traits</code>	Optional character vector of trait names used for sign anchoring. Supply one trait per axis after ordering. Axes without a supplied anchor use the trait with the largest absolute loading.
<code>loading_scale</code>	One of "raw" or "standardized". "raw" returns the rotated loadings on the fitted response scale. "standardized" divides each trait loading by the square root of that trait's model-implied total variance, matching <code>plot(type = "ordination", standardize_loadings = TRUE)</code> .

Details

Scope: covered for fitted `latent()` components at `level = "unit"` or `level = "unit_obs"` with raw or standardized point-estimate loadings. This table does not attach uncertainty intervals to loadings. Bootstrap or simulation-based uncertainty is not yet implemented and remains possible future work for a later inference slice.

Value

A data frame with columns `level`, `trait`, `axis`, `loading`, `abs_loading`, `axis_variance`, `axis_share`, `rotation`, `order_axes`, `sign_anchor`, `anchor_trait`, and `loading_scale`. `axis_variance` and `axis_share` are computed from the rotated raw loading matrix before any optional standardization; these are the quantities used to order axes.

See Also

`rotate_loadings()`, `extract_ordination()`, `extract_Sigma()`, `plot.gllvmTMB_multi()`.

Examples

```

sim <- simulate_site_trait(
  n_sites = 60, n_species = 12, n_traits = 4,
  Lambda_B = matrix(c(1.0, 0.7, -0.3, 0.5,
                      0.3, -0.5, 0.8, 0.2),
                    nrow = 4, ncol = 2),
  psi_B = c(0.3, 0.3, 0.3, 0.3),
  seed = 1
)
fit <- gllvmTMB(value ~ 0 + trait +
               latent(0 + trait | site, d = 2),
               data = sim$data,
               trait = "trait",
               unit = "site")
extract_rotated_loadings_table(
  fit,
  level = "unit",
  method = "varimax",
  loading_scale = "standardized"
)

```

extract_Sigma

Extract the implied trait covariance / correlation at one tier

Description

Implements the decomposition

$$\Sigma_{\text{tier}} = \underbrace{\Lambda_{\text{tier}} \Lambda_{\text{tier}}^{\top}}_{\text{shared (latent)}} + \underbrace{\Psi_{\text{tier}}}_{\text{unique}}$$

where ordinary `latent()` now carries both Λ and the diagonal Ψ companion by default. This is the same decomposition the behavioural-syndromes / phenotypic-integration literature uses (Bartholomew et al. 2011).

Usage

```

extract_Sigma(
  fit,
  level = c("unit", "unit_slope", "unit_obs", "phy", "phy_slope", "column_slope",
            "spatial", "spde_slope", "cluster", "cluster2", "B", "B_slope", "W", "spde"),
  part = c("total", "shared", "unique", "psi"),
  link_residual = c("auto", "none"),
  .skip_warn = FALSE
)

```

Arguments

<code>fit</code>	A fit returned by <code>gllvmTMB()</code> . Admitted engine = "julia" bridge fits expose the ordinary unit tier only: <code>link_residual = "none"</code> reconstructs $\Lambda\Lambda^\top$ from retained loadings, while <code>link_residual = "auto"</code> uses the retained GLLVM.jl residual-augmented payload where available and keeps Gaussian / lognormal rows on the native no-op convention. <code>unit_obs</code> , structured tiers, and augmented-slope tiers remain gated for Julia bridge extractors.
<code>level</code>	One of "unit" (between-unit), "unit_obs" (within-unit), "phy" (phylogenetic), "column_slope" (the predictor-basis covariance from <code>phylo_indep(0 + x1 + ... + xP trait)</code> or <code>animal_indep(0 + x1 + ... + xP trait)</code>), "spatial", "cluster", or "cluster2" (a second, independent diagonal grouping alongside "cluster" – see the <code>cluster2</code> argument to <code>gllvmTMB()</code>). Legacy aliases "B", "W", and "spde" are accepted with a soft-deprecation message.
<code>part</code>	One of "total" (default), "shared", "unique". "psi" is an alias for "unique" (the per-trait residual Psi diagonal, now folded into <code>latent()</code> by default).
<code>link_residual</code>	For non-Gaussian fits. "auto" (default) adds a per-trait link-specific implicit residual variance to the diagonal of Sigma, giving the marginal latent-scale interpretation; in mixed- family fits each unambiguous trait gets the residual implied by its family/link (see "Family-aware link residuals" below for the full table). "none" returns the fitted model covariance without link-residual additions ($\Lambda\Lambda^\top + \Psi$ where a latent decomposition is present). For Gaussian or lognormal-only fits this argument is effectively a no-op (their implied $\sigma_d^2 = 0$).
<code>.skip_warn</code>	Internal flag (default FALSE). When TRUE, suppresses the once-per-session deprecation message that the internal <code>.normalise_level()</code> helper emits for legacy level aliases ("B", "W", "spde", "total"). Used by internal callers (e.g. <code>extract_Omega()</code>) that have already issued the deprecation message themselves; not part of the public API.

Details

Legacy covariance wrappers:

`extract_Sigma_B()` and `extract_Sigma_W()` are soft-deprecated compatibility wrappers. They still return their historical `Sigma_B / R_B` and `Sigma_W / R_W` field names, respectively, but new code should call this function with `level = "unit"` or `level = "unit_obs"`.

When a fit has no Psi component:

If the formula deliberately uses `latent(0 + trait | unit, d = K, unique = FALSE)`, the engine fits only the $\Lambda\Lambda^\top$ component. Calling `extract_Sigma(fit, level, part = "total")` then returns just the shared component. This is useful for an intentionally loadings-only target. It should not be described as a full latent-liability covariance unless the relevant family/link residual is also included. Whether the diagonal is incomplete depends on the estimand and response family.

For Gaussian / lognormal / Gamma fits this function emits an advisory note when a reduced-rank tier has no Psi component. Use the ordinary `latent(..., unique = TRUE)` default for $\Lambda\Lambda^\top + \Psi$.

For non-Gaussian families (binomial, Poisson, Gamma) the latent-scale residual variance has a closed-form approximation that should be added to the diagonal of Σ – see the `link_residual` argument below.

The part argument:

"total" (default) $\Sigma_{\text{tier}} = \Lambda\Lambda^\top + \Psi$ – the matrix users almost always want for reporting correlations.

"shared" $\Lambda\Lambda^\top$ only – the reduced-rank, rotation-invariant component. Diagonals are $\sum_\ell \Lambda_{\ell\ell}^2$; these are *not* the trait variances, they are the *shared* part of the trait variances.

"unique" Ψ_{tier} only – the trait-specific unique variances, returned as a length-T named numeric vector (the diagonal of Ψ).

Caveat: "shared" vs "unique" partition is only weakly identified:

The total $\Sigma_{\text{tier}} = \Lambda\Lambda^\top + \Psi$ is rotation-invariant, so `part = "total"` avoids arbitrary loading rotation. Rotation invariance does not guarantee that the data identify every covariance entry; fit health, sample design, and uncertainty still matter. The *split* between $\Lambda\Lambda^\top$ and Ψ is only weakly identified – different optimiser starts can flow trait t 's variance more into the shared ("shared") or unique ("unique") component, with the same total likelihood. In a synthetic Poisson + OLRE recovery run with target $\sigma_S^2 = (0.5, 0.4, 0.3, 0.6)$, `gllvmTMB` returned $(0, 0.67, 0, 0.57)$ – total trait variances correct, but the partition arbitrary.

This is the standard rotation-and-shift indeterminacy of factor models (`gllvm` and `Hmsc` have it too); communality c_t^2 computed from a single fit therefore inherits the same indeterminacy.

For interpretation, lean on `part = "total"` for correlations and use `extract_communality()` only when paired with `bootstrap_Sigma()` or the variance-decomposition consistency checks in `gllvmTMB_diagnose()`.

Family-aware link residuals:

For non-Gaussian responses each row carries an implicit observation-level residual on the latent (link) scale. Adding it to the per-trait diagonal of Σ defines one family-specific latent-scale convention. It can make the denominator explicit, but it does not make different response families directly commensurate without additional scientific assumptions.

Per-family formulas (Nakagawa & Schielzeth 2010; Nakagawa, Johnson & Schielzeth 2017):

<code>gaussian(identity)</code>	$\sigma_d^2 = 0$
<code>binomial(link = "logit")</code>	$\sigma_d^2 = \pi^2/3 \approx 3.290$
<code>binomial(link = "probit")</code>	$\sigma_d^2 = 1$
<code>binomial(link = "cloglog")</code>	$\sigma_d^2 = \pi^2/6 \approx 1.645$
<code>poisson(link = "log")</code>	$\sigma_d^2 = \log(1 + 1/\hat{\mu}_t)$ (lognormal-Poisson approx.)
<code>lognormal(link = "log")</code>	$\sigma_d^2 = 0$ (<code>sigma_eps</code> already models the log-scale residual)
<code>Gamma(link = "log")</code>	$\sigma_d^2 = \psi'(\hat{\nu})$ where $\hat{\nu} = \hat{\phi}_{\gamma,t}$ is the per-trait shape
<code>nbinom2(link = "log")</code>	$\sigma_d^2 = \psi'(\hat{\phi})$ where $\hat{\phi}$ is the per-trait NB2 dispersion
<code>tweedie(link = "log")</code>	$\sigma_d^2 = \log(1 + \hat{\phi}\hat{\mu}_t^{\hat{p}-2})$ (delta method)
<code>Beta(link = "logit")</code>	$\sigma_d^2 = \psi'(\hat{\mu}_t\hat{\phi}) + \psi'((1 - \hat{\mu}_t)\hat{\phi})$ (Smithson & Verkuilen 2006)
<code>betabinomial(link = "logit")</code>	$\sigma_d^2 = \pi^2/3 + \psi'(\hat{\mu}_t\hat{\phi}) + \psi'((1 - \hat{\mu}_t)\hat{\phi})$

`multinomial()` is the exception to the scalar-per-trait table: its distribution-specific residual is not a scalar but the $(K - 1) \times (K - 1)$ matrix $(\pi^2/6)(\mathbf{I} + \mathbf{J})$ over the category-contrast pseudo-traits – $\pi^2/3$ on the diagonal (each baseline contrast is a logit, exactly as for `binomial(link = "logit")`) and $\pi^2/6$ off-diagonal (the shared baseline category couples the contrasts). This is the residual covariance implied by the softmax's random-utility (Gumbel) representation (McFadden 1974), and it reduces to the binomial $\pi^2/3$ at $K = 2$. (It is distinct from `MCMCglmm`'s *arbitrary* fixed identification residual $(1/K)(\mathbf{I} + \mathbf{J})$ and that package's MCMC-specific c^2 overdispersion correction,

neither of which a direct softmax fit needs.) `link_residual = "auto"` (the default) applies this full-matrix residual: the $(\pi^2/6)(\mathbf{I} + \mathbf{J})$ block is added to the multinomial trait's $(K - 1)$ -contrast block of Σ (the diagonal $\pi^2/3$ entry-by-entry with the other families, the $\pi^2/6$ off-diagonal as a block); "none" returns the latent-scale $\mathbf{V}$ unchanged. The addition is tier-agnostic – it applies to whichever tier Σ is requested (`level = "phy"`, a spatial or kernel tier, or an ordinary latent term) – and it makes the categorical block commensurable with any single-scale (e.g. Gaussian) trait sharing the fit. Relatedly, a *diagonal* among-category $\mathbf{V}$ is not independence: the independence null for the contrasts is itself $(\mathbf{I} + \mathbf{J})$ -structured (equal diagonal, equal off-diagonal).

For mixed-family fits the residual is computed *per trait* when that trait has one unambiguous family/link, then added to the diagonal of Σ entry-by-entry. The default `link_residual = "auto"` applies this; "none" returns the fitted model covariance without link-residual additions ($\Lambda\Lambda^\top + \Psi$ where a latent decomposition is present). For continuous-only Gaussian or lognormal fits "auto" is a no-op (their per-trait σ_d^2 is zero).

Citations: Nakagawa & Schielzeth (2010); Nakagawa, Johnson & Schielzeth (2017); McFadden (1974) for the multinomial-logit random-utility residual — see **References** below.

Future: 3+ latent tiers:

The engine currently supports two `latent()` tiers ("unit" and "unit_obs"); legacy `level = "B"` and "W" aliases are still accepted with a soft-deprecation message. `level = "phy"` extracts the phylogenetic implied Σ_{phy} from `phylo_latent()`. If a future release adds 3+ latent tiers, `level = "<colname>"` will dispatch to the corresponding tier without API change. For now, custom strings error with a clear unsupported-level message.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(
  value ~ 0 + trait +
    latent(0 + trait | unit, d = 2),
  data = df,
  trait = "trait",
  unit = "unit"
)
extract_Sigma(fit, level = "unit", part = "total")$Sigma # full T x T cov
extract_Sigma(fit, level = "unit", part = "shared")$Sigma # rr-only
extract_Sigma(fit, level = "unit", part = "unique")$s    # diag(s_unit)
```

Value

For `part = "total"` or "shared": a list with components `Sigma` ($T \times T$ matrix), `R` ($T \times T$ correlation matrix; only for "total"), `level`, `part`, and `note` (character vector of advisory messages, e.g. about a no-Psi unique = FALSE fit).

For `part = "unique"`: a list with `s` (length- T named numeric vector of unique variances), `level`, `part`, `note`.

For a `phylo_dep(1 + x1 + ... + xs | species)` fit with one or more slopes, call with `level = "phy"`: the result is the single full unstructured $(1+s)T \times (1+s)T$ covariance over the trait-stacked (intercept, slope_1, ..., slope_s) random-effect columns – a list with `Sigma` and `R` carrying INTERLEAVED dimnames (per trait: `intercept.<t>`, then `slope.<t>` for `s == 1` or `slope.<x_j>.<t>`).

for $s \geq 2$), `level = "phy_dep"`, `part = "dep"`, and a note. The `part` and `link_residual` arguments do not apply to this single unstructured block and are ignored. (The `unit / unit_obs` tiers return NULL for a dep-only fit, as it carries no between/within-unit covariance term.)

Current `phylo_indep(1 + x | species)` uses the same interleaved 2T report channel with all cross-trait blocks fixed to zero. Its return has `level = "phy_indep_slope"` and `part = "indep"`; each trait retains its own intercept-slope block. This is distinct from the legacy shared 2×2 augmented `phylo_unique` channel.

For augmented spatial slopes, call with `level = "spatial"`. `spatial_unique(1 + x | coords)` retains the shared 2×2 field-scale channel and returns `level = "spde_base_slope"`. Current `spatial_indep(1 + x | coords)` returns the interleaved, cross-trait block-diagonal $2T \times 2T$ field covariance with `level = "spde_indep_slope"` and `part = "indep"`; `spatial_dep` returns the full unstructured block with `level = "spde_dep"` and `part = "dep"`. These SPDE matrices are on the fitted field-covariance scale; the returned note records the marginal-scale conversion.

For an ordinary individual-level random-regression fit with `latent(1 + x | unit, d = K)`, `indep(1 + x | unit)`, or their long-form equivalents, call with `level = "unit_slope"`: the result is the augmented $2T \times 2T$ covariance over trait-specific intercept and slope coefficients, with row names `intercept.<trait>` and `slope.<x>.<trait>`. As for `level = "unit"`, `part = "shared"` returns `Lambda_aug Lambda_aug^T`, `part = "unique"` returns the augmented diagonal, and `part = "total"` returns their sum.

References

- Nakagawa, S. & Schielzeth, H. (2010). Repeatability for Gaussian and non-Gaussian data: a practical guide for biologists. *Biological Reviews* 85, 935-956. doi:10.1111/j.1469185X.2010.00141.x
- Nakagawa, S., Johnson, P. C. D., & Schielzeth, H. (2017). The coefficient of determination R^2 and intra-class correlation coefficient from generalized linear mixed-effects models revisited and expanded. *Journal of the Royal Society Interface* 14(134), 20170213. doi:10.1098/rsif.2017.0213
- McFadden, D. (1974). Conditional logit analysis of qualitative choice behavior. In P. Zarembka (Ed.), *Frontiers in Econometrics* (pp. 105-142). Academic Press, New York. (Random-utility origin of the multinomial-logit $(\pi^2/6)(\mathbf{I} + \mathbf{J})$ contrast residual.)

See Also

[extract_communality\(\)](#) for the per-trait shared / unique variance share at one tier; [extract_proportions\(\)](#) for the canonical per-trait variance decomposition across tiers; [extract_Omega\(\)](#) for the multi-tier sum.

<code>extract_Sigma_table</code>	<i>Extract a report-ready table of Sigma or correlation entries</i>
----------------------------------	---

Description

`extract_Sigma()` returns matrices because that is the most convenient interactive representation. `extract_Sigma_table()` returns the same covariance target as one row per matrix entry so articles, tables, and plot helpers can work without hand-indexing matrices. It can also turn a [bootstrap_Sigma\(\)](#) result into the same row schema with bootstrap interval columns filled in.

Usage

```
extract_Sigma_table(
  fit,
  level = "unit",
  part = c("total", "shared", "unique"),
  measure = c("covariance", "correlation"),
  entries = c("unique", "all", "upper", "lower", "offdiag", "diag"),
  link_residual = c("auto", "none")
)
```

Arguments

fit	A fit returned by <code>gllvmTMB()</code> , an admitted engine = "julia" bridge fit, or a <code>bootstrap_Sigma()</code> result.
level	Character vector of covariance levels, or "all" for every level present in the fit. Canonical levels are "unit", "unit_obs", "cluster", "cluster2", "phy", and "spatial"; named kernel_*() tiers are accepted by their fitted name; legacy aliases "B", "W", and "spde" are accepted. For <code>gllvmTMB_julia</code> objects, only "unit" is currently routed; "all" maps to that tier.
part	One of "total" (default), "shared", or "unique", passed to <code>extract_Sigma()</code> .
measure	One of "covariance" (default) or "correlation". Correlation tables are available only for part = "total" because the report-ready correlation target is based on the full $\Lambda \Lambda^T + \Psi$ covariance.
entries	Which symmetric matrix entries to return. "unique" (default) returns the diagonal plus upper triangle, one row per unique estimand. "all" returns every cell, useful for heatmaps. "upper", "lower", "offdiag", and "diag" return the corresponding subsets.
link_residual	Passed to <code>extract_Sigma()</code> . "auto" (default) adds family/link implicit residual variances to non-Gaussian trait diagonals; "none" returns only the fitted model-implied covariance.

Details

The helper reshapes the same covariance and correlation matrices returned by `extract_Sigma()`. A `bootstrap_Sigma()` object can supply interval columns. Julia bridge fits currently expose only point estimates for the ordinary unit tier. The helper does not create profile or Wald intervals.

The table is a point-estimate view over `extract_Sigma()`. It does not compute confidence intervals from a fitted model directly. Use `extract_correlations()` when you need pairwise correlation intervals, or `bootstrap_Sigma()` followed by `extract_Sigma_table()` when you need bootstrap intervals for Sigma or correlation matrix entries.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(
  value ~ 0 + trait +
    latent(0 + trait | unit, d = 2),
```

```

    data = df,
    trait = "trait",
    unit = "unit"
  )
  extract_Sigma_table(fit, level = "unit")
  extract_Sigma_table(fit, level = "unit", measure = "correlation")
  extract_Sigma_table(fit, level = c("unit", "unit_obs"), entries = "all")
  boot <- bootstrap_Sigma(fit, n_boot = 50, level = "unit",
    what = "Sigma", progress = FALSE)
  extract_Sigma_table(boot, level = "unit", entries = "upper")

```

Value

A data frame with one row per requested entry and stable columns: estimand, trait_i, trait_j, integer indices i and j, level, component, matrix, estimate, lower, upper, interval_method, interval_status, scale, diagonal, and triangle. Interval columns are NA with interval_method = "none" because this helper is point-estimate only.

See Also

[extract_Sigma\(\)](#) for the underlying matrix extractor; [extract_correlations\(\)](#) for pairwise correlation intervals; [bootstrap_Sigma\(\)](#) for bootstrap uncertainty.

fitted.gllvmTMB_multi *Fitted values from a fitted gllvmTMB model*

Description

A thin wrapper over [predict.gllvmTMB_multi\(\)](#) at the training rows (newdata = NULL). Returns the same long data frame [predict\(\)](#) returns – one row per training observation, carrying its unit/species/trait identifiers alongside the fitted value – rather than a bare vector, so row identity against the original data is preserved. ... is forwarded to [predict\(\)](#) unchanged, so its other arguments (se.fit, re_form) work here too.

Usage

```

## S3 method for class 'gllvmTMB_multi'
fitted(object, type = c("response", "link"), ...)

```

Arguments

object	A fit returned by gllvmTMB() .
type	One of "response" (default, the fitted() convention) or "link".
...	Forwarded to predict.gllvmTMB_multi() (e.g. se.fit, re_form); see its documentation for what each accepts.

Value

A data frame with the original row identifiers (unit, species, trait columns) plus an est column on the requested scale – identical in shape to `predict(object, newdata = NULL, type = type, ...)`. **For a multinomial() fit** this is `.predict_multinomial()`'s output: K rows per observation (one per category, including the baseline), not one.

flag_unreliable_loadings

Flag loadings whose CI overlaps a "biologically negligible" region

Description

A simple decision aid in the spirit of Zientek & Thompson (2007, doi:10.3758/BF03193163): given per-entry CIs on Lambda, flag the entries whose `conf_level` confidence interval **does not exclude** a user-supplied "null region" — a band around zero considered biologically negligible on the interval's recorded `loading_scale`. Loadings flagged as `unreliable = TRUE` are the ones for which the data do not provide evidence that the species responds non-trivially to the axis.

Usage

```
flag_unreliable_loadings(
  fit,
  null_region = c(-0.1, 0.1),
  level = c("unit", "unit_obs"),
  method = "wald",
  conf_level = 0.95,
  loading_scale = NULL
)
```

Arguments

<code>fit</code>	A multivariate <code>gllvmTMB()</code> fit, or a data frame already produced by <code>loading_ci()</code> .
<code>null_region</code>	Length-2 numeric, sorted ascending. The "negligible" band. Defaults to <code>c(-0.1, 0.1)</code> .
<code>level, method, conf_level, loading_scale</code>	Forwarded to <code>loading_ci()</code> when <code>fit</code> is a fit object. Ignored when <code>fit</code> is already a <code>loading_ci()</code> data frame, except that an explicitly supplied <code>loading_scale</code> must match the frame's scale.

Details

Pinned entries (set explicitly by `lambda_constraint`) are reported with `unreliable = NA` because no inference is being made about them.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
## Grounded in tests/testthat/test-loading-ci.R. flag_unreliable_loadings()
## accepts either a fit or a loading_ci() data frame directly.
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 2L),
  data = df, family = stats::binomial(link = "probit")
)
## From a fit (computes per-entry Lambda CIs internally):
flag_unreliable_loadings(fit, null_region = c(-0.1, 0.1))
## Or pass a precomputed loading_ci() data frame:
ci <- loading_ci(fit, level = "unit")
flag_unreliable_loadings(ci, null_region = c(-0.1, 0.1))
```

Value

The `loading_ci()` data frame with one extra logical column `unreliable`: TRUE if the CI overlaps `null_region`, FALSE if it lies entirely outside, NA for pinned entries.

See Also

[loading_ci\(\)](#).

get_crs

Infer a UTM coordinate reference system

Description

Infer a UTM coordinate reference system

Usage

```
get_crs(dat, ll_names = c("longitude", "latitude"))
```

Arguments

<code>dat</code>	Data frame containing longitude and latitude columns.
<code>ll_names</code>	Character names of longitude then latitude columns.

Value

An EPSG code for the predominant UTM zone.

gllvm_julia_fit	<i>Fit a GLLVM with the Julia engine (GLLVM.jl bridge_fit).</i>
-----------------	---

Description

Fit a GLLVM with the Julia engine (GLLVM.jl bridge_fit).

Usage

```
gllvm_julia_fit(
  y,
  family = "gaussian",
  num.lv = 2L,
  N = NULL,
  X = NULL,
  X_lv = NULL,
  coef_fixed = NULL,
  mask = NULL,
  units_are_rows = FALSE,
  ci_method = c("none", "wald", "profile", "bootstrap"),
  ci_level = 0.95,
  ci_nboot = 200L,
  ci_seed = 0L,
  ...
)
```

Arguments

y	Response matrix, p x n (traits x units), or n x p (set units_are_rows).
family	A family object/string, or a list of them (one per trait -> mixed).
num.lv	Number of latent variables (K).
N	Binomial trials (matrix or scalar), or NULL.
X	Fixed-effect design (p x n x q array), or NULL. Routed for Gaussian and selected one-part non-Gaussian bridge families.
X_lv	Unit-level predictor-informed latent-score design (n x q_lv matrix), or NULL. Routed only for complete Gaussian, Poisson, NB2, Gamma, Beta, and binomial logit/probit/cloglog bridge rows with no fixed-effect X, no response mask, and ci_method in c("none", "wald"). The returned object carries point-estimate alpha_lv, lv_effects, scores_mean, and scores_innovation payloads; for ci_method = "wald" it additionally carries lv_effects_se, lv_effects_lower, and lv_effects_upper, which extract_lv_effects() surfaces as finite std.error, lower, and upper when called with type = "trait_effect". If future bridge payloads include alpha_lv_se, alpha_lv_lower, and alpha_lv_upper, extract_lv_effects() surfaces those on the default axis-effect table. These are bridge-reader Wald

	payloads, not calibrated coverage claims. Profile/bootstrap X_{lv} confidence intervals, NB1, ordinal, mixed-family, masked, and fixed-effect-X-plus- X_{lv} rows remain gated.
coef_fixed	Optional logical vector of length $\dim(X)[3]$. TRUE entries are fixed at zero by the Julia bridge. Most R users should prefer the named <code>Xcoef_fixed</code> argument to <code>gllvmTMB()</code> , which is translated to this positional mask after the expanded fixed-effect design is known.
mask	Optional logical response-observation mask with the same orientation as <code>y</code> ; TRUE cells contribute to the likelihood and FALSE cells are ignored. Currently routed for one-part no-X non-Gaussian point fits, with masked no-X CI payloads for Poisson, Bernoulli binomial, NB2, NB1, Beta, and Gamma.
units_are_rows	If TRUE, <code>y</code> is $n \times p$ and is transposed to $p \times n$.
ci_method	Confidence-interval route for admitted no-X bridge rows: Gaussian, Poisson, Bernoulli binomial, and grouped-dispersion NB2, NB1, Beta, and Gamma. One of "none" (default), "wald", "profile", or "bootstrap". Response-mask CIs are routed for the same non-ordinal non-Gaussian rows when $X = \text{NULL}$; complete-response fixed-effect-X CIs are routed for Gaussian, Poisson, Bernoulli binomial, NB2, Beta, and Gamma. Predictor-informed X_{lv} rows admit "wald" confidence intervals. Per-trait ordinal rows, NB1-X rows, mixed-family vectors, profile/bootstrap X_{lv} rows, and response masks combined with fixed-effect X remain loud gates.
ci_level	Nominal confidence level when <code>ci_method != "none"</code> .
ci_nboot	Number of parametric bootstrap replicates when <code>ci_method = "bootstrap"</code> .
ci_seed	Seed passed to the Julia bootstrap CI route.
...	Passed to <code>gllvm_julia_setup()</code> (<code>j1_path</code> , <code>julia_home</code>).

Value

A list of class `gllvmTMB_julia` with the bridge contract fields (loadings, Sigma, correlation, loglik, aic, bic, converged, ...). If `ci_method != "none"` and the row is admitted, the list also carries flat CI fields consumed by `confint()`.

Examples

```
## Not run:
# Requires a local GLLVM.jl install (see [gllvm_julia_setup()]).
# `y` is traits x units (p x n): two Gaussian traits, 40 units, K = 2.
set.seed(1)
y <- matrix(rnorm(2 * 40), nrow = 2)
fit <- gllvm_julia_fit(y, family = "gaussian", num.lv = 2)
fit$loglik
fit$Sigma

# Predictor-informed latent-score means are point-only on the bridge.
rownames(y) <- paste0("trait", seq_len(nrow(y)))
colnames(y) <- paste0("site", seq_len(ncol(y)))
x <- seq(-1, 1, length.out = ncol(y))
fit_lv <- gllvm_julia_fit(
```

```

    y,
    family = "gaussian",
    num.lv = 1,
    X_lv = matrix(x, ncol = 1, dimnames = list(colnames(y), "x"))
  )
  extract_lv_effects(fit_lv)

# Wald confidence intervals for an admitted no-X row:
fit_ci <- gllvm_julia_fit(
  y, family = "gaussian", num.lv = 2, ci_method = "wald"
)

## End(Not run)

```

gllvm_julia_setup	<i>Set up JuliaCall and load the GLLVM.jl engine (once per R session).</i>
-------------------	--

Description

Set up JuliaCall and load the GLLVM.jl engine (once per R session).

Usage

```

gllvm_julia_setup(
  jl_path = getOption("gllvmTMB.GLLVM.jl.path", Sys.getenv("GLLVM_JL_PATH", "")),
  julia_home = getOption("gllvmTMB.julia_home", Sys.getenv("JULIA_HOME", ""))
)

```

Arguments

jl_path	Path to the GLLVM.jl project that provides bridge_fit (default: option gllvmTMB.GLLVM.jl.path or env GLLVM_JL_PATH).
julia_home	Julia bin directory (default: option gllvmTMB.julia_home or env JULIA_HOME; if unset, JuliaCall auto-discovers).

Value

Invisibly TRUE once ready.

Description

Top-level entry point for models with several responses per site, individual, species, or study. Start from a wide data frame with `traits()` on the formula left-hand side, or from already-stacked long data with one row per (unit, trait) observation. Both routes estimate the same trait covariance, pairwise correlations, shared latent axes, and trait-specific variance. The formula syntax also supports fixed effects plus covariance-structure keywords organised by *correlation source* (none / animal / phylo / spatial / kernel) and three *modes* (independent / dependent / latent). The `common = TRUE` modifier on `*_indep()` gives the one-shared-variance special case:

Usage

```
gllvmTMB(
  formula,
  data,
  trait = "trait",
  unit = "site",
  unit_obs = NULL,
  cluster = NULL,
  cluster2 = NULL,
  family = gaussian(),
  weights = NULL,
  REML = FALSE,
  mesh = NULL,
  phylo_vcv = NULL,
  phylo_tree = NULL,
  known_V = NULL,
  lambda_constraint = NULL,
  Xcoef_fixed = NULL,
  control = gllvmTMBcontrol(),
  missing = miss_control(),
  impute = NULL,
  silent = TRUE,
  engine = c("tmb", "julia"),
  ci_method = c("none", "wald", "profile", "bootstrap"),
  ci_level = 0.95,
  ci_nboot = 200L,
  ci_seed = 0L,
  site = NULL,
  species = NULL,
  estimator = c("ml", "mspl")
)
```

Arguments

formula	A glmmTMB-style formula, e.g. <code>value ~ 0 + trait + (0 + trait):env_temp + (0 + trait):env_precip</code>. Fixed effects and any of the three-mode grid cov-structs above are supported (plus <code>phylo_slope()</code>, <code>animal_slope()</code>, and <code>meta_V()</code>). An <code>offset()</code> term is supported for count responses only: <code>poisson()</code>, <code>nbinom1()</code>, <code>nbinom2()</code>, <code>truncated_poisson()</code>, and <code>truncated_nbinom2()</code>. There it is the usual exposure or effort adjustment. Supply it already on the link scale, e.g. <code>offset(log(trap_nights))</code>. It must be its own additive term; it is not interacted with <code>trait</code>. In long format the offset column carries one value per (unit, trait) row, so it can differ by trait. In wide format, <code>offset(w)</code> names one unit-level column and applies it to every trait, while <code>offset(e1, e2, ...)</code> names one column per trait, in <code>traits()</code> order. A nonzero offset on a non-count trait is an error naming that trait. Zero is always allowed and does nothing (on the log scale it is a multiplier of one), which is how a mixed-family model gives an offset to its count traits and not the rest.
data	A data frame. With an ordinary response LHS such as <code>value ~ ...</code>, data is long: one row per (unit, trait) observation, with the trait column named by <code>trait</code>. With a <code>traits()</code> LHS, data is wide: one row per unit and one column per response named inside <code>traits(...)</code>. Missing response values are allowed: by default they are dropped before fitting, while <code>missing = miss_control(response = "include")</code> masks them out of the likelihood and keeps original-row accounting for <code>predict_missing()</code>. Ordinary missing predictors, grouping variables, and design-matrix values still error; explicitly modelled missing predictors use <code>mi(x)</code> with <code>missing = miss_control(predictor = "model")</code> and <code>impute = list(...)</code>.
trait	Name of the column holding the trait factor (the "trait" dimension of the unit $\times$ trait response matrix). Default "trait".
unit	Name of the column holding the between-unit grouping factor (the "unit" dimension of the unit $\times$ trait response matrix). Examples: "site" for site $\times$ species data, "individual" for behavioural-syndrome data, "species" for PGLLVM, "paper" for systematic mapping. Default "site".
unit_obs	Optional. Name of the column holding the within-unit grouping factor, with one level per (unit, replicate) cell, used by <code>latent(0 + trait unit_obs, ...)</code> and <code>indep(0 + trait unit_obs)</code> for the W-tier covariance. Default NULL, which resolves to "site_species" (the conventional name in joint species distribution modelling; safe for site $\times$ species data). If your data has no such column, omit <code>unit_obs</code>; the engine synthesises it from <code>unit $\times$ cluster</code>. For other domains pass e.g. <code>unit_obs = "obs"</code> for behavioural syndromes. If supplied but no covariance keyword uses that column, <code>gllvmTMB()</code> warns; omit the argument or use the column in the intended covariance keyword.
cluster	Optional. Name of the column holding the third grouping factor (the "cluster" slot). When the column name matches the row/column names of <code>phylo_vcv</code> (or the tip labels of <code>phylo_tree</code>), this slot also drives the phylogenetic random effects (<code>phylo_latent</code>, <code>phylo_scalar</code>, <code>phylo_indep</code>, <code>phylo_slope</code>). When the column does not match a phylogenetic correlation, the slot still functions as a

regular crossed/nested third grouping (e.g. `cluster = "population"` for 3-level personality data, `cluster = "study"` for multi-study meta-analysis). Default NULL, which resolves to "species". For datasets with no third grouping the column may be absent from data entirely (with or without passing `cluster`); the engine will synthesise a placeholder. If supplied but no covariance keyword uses that column, `gllvmTMB()` warns; omit the argument or use the column in the intended covariance keyword.

The engine does **not** enforce nesting between `unit_obs`, `unit`, and `cluster`; crossed and nested designs both fit. Two canonical patterns:

- Functional biogeography (crossed): `unit = "site"`, `unit_obs = "site_species"`, `cluster = "species"`. Sites and species are crossed; `site_species` indexes the intersection cell.
- 3-level personality (strictly nested): `unit = "individual"`, `unit_obs = "session_id"`, `cluster = "population"`.

`cluster2` Optional name of the column holding a **second independent grouping** factor (the "cluster2" slot), for fitting two crossed (or nested) plain diagonal per-trait variance components at once. Default NULL (slot inactive). It is a plain crossed/nested diagonal grouping only – it carries no phylogenetic or spatial correlation (those stay bound to `cluster` / `coords`). A `indep(0 + trait | <cluster2 col>)` term then fits a per-trait variance at this second grouping, exactly as `cluster` does at the third slot. Example: `cluster = "site"`, `cluster2 = "year"` to fit a site variance and a year variance simultaneously. As with the other slots, nesting is not enforced (crossed and nested both fit). The `cluster2` column must be disjoint from the `unit` / `unit_obs` / `cluster` columns (a diagonal term routes to whichever slot its grouping column matches).

`family` A family object. The multivariate engine (formulas with `latent()`, `indep()`, etc.) supports `gaussian()`, `binomial()` (logit / probit / cloglog), `poisson()` (log link), `ordinal_probit()` (the gllvmTMB-native ordinal threshold family with `sigma_d = 1` fixed exactly; no delta-method approximation), `lognormal()` (log link), `Gamma(link = "log")`, `nbinom2()` (NB2 negative binomial; log link), `nbinom1()` (NB1 negative binomial; log link), `tweedie()` (compound Poisson-Gamma; log link), `Beta()` (logit link, mean-precision parameterisation; y in $(0, 1)$), `betabinomial()` (logit link, mean-precision parameterisation; k -of- n data), `student()` (heavy-tailed continuous; identity link), `truncated_poisson()` (zero-truncated count; log link), `truncated_nbinom2()` (zero-truncated NB2; log link), and the hurdle / two-part families `delta_lognormal()` / `delta_gamma()` (Bernoulli for presence + Lognormal/Gamma for the positive component; share one linear predictor under the current implementation – see Details). For multi-trial binomial or beta-binomial (k -of- n) data, write the LHS of formula as `cbind(successes, failures)` (the canonical R / `glm()` / `glmmTMB` convention); the engine then uses the per-row trial count `successes + failures` as the binomial size. A `weights = numeric` vector of trial counts is also accepted as an alternative API (see Details). The negative-binomial families fit one log-dispersion parameter per trait: `nbinom1()` uses $\text{Var} = \mu * (1 + \phi)$, while `nbinom2()` uses $\text{Var} = \mu + \mu^2 / \phi$. The `tweedie()` family fits one log- ϕ and one logit- $(p-1)$ per trait, with p in $(1, 2)$. The `Beta()` and `betabinomial()` families each fit one log-precision per trait ($a = \mu * \phi$, $b = (1 - \mu) * \phi$; Smithson & Verkuilen 2006). The `student()` family fits one log-sigma and one

	$\log(df-1)$ per trait ($df > 1$; pass <code>student(df = 3)</code> to fix df at a known value). The <code>truncated_*</code> families require strictly positive integer responses ($y \geq 1$). The delta families fit one log-dispersion of the <i>positive</i> component per trait (<code>sigma_lognormal</code> or <code>phi_gamma</code>).
weights	Optional numeric vector of length <code>nrow(data)</code>. Family- conditional semantics (<code>lme4</code> / <code>glmmTMB</code> convention): - For non-binomial families, <code>weights</code> is interpreted as per- observation likelihood multipliers: each row's log-likelihood contribution is multiplied by <code>weights[i]</code>. This matches the <code>weights</code> argument of <code>lme4::lmer()</code>, <code>glmmTMB::glmmTMB()</code> and <code>stats::glm()</code>. Use this for relative-abundance weighting (<code>sqrt(p_si)</code>), inverse-variance weighting (<code>1 / var_i</code> for heteroscedastic Gaussian fits), or to down-weight outliers. Must be non-negative and finite; <code>weights[i] = 0</code> zeroes that row's contribution to the joint NLL. Non-unit likelihood weights define a weighted estimating objective, not an ordinary maximum-likelihood fit. The package retains point estimates but refuses likelihood-based model-selection methods; Hessian/Wald uncertainty is not sandwich-corrected and must not be interpreted as calibrated inference for the weighted objective. - For binomial families fit without a <code>cbind(succ, fail)</code> LHS, <code>weights</code> continues to be interpreted as the per-row trial count (binomial size; alternative API to <code>cbind(succ, fail)</code>). This pre-existing semantics is preserved on a per-row basis: in mixed- family fits, binomial rows use <code>weights[i]</code> as their trial count and non-binomial rows use it as the likelihood multiplier. Default NULL is equivalent to a length-<code>nrow(data)</code> vector of ones (un-weighted Bernoulli on binomial rows). Wide matrix calls through <code>gllvmTMB_wide()</code> and wide data-frame calls through <code>traits()</code> normalise their accepted weight shapes to this same stacked response vector before fitting.
REML	Logical; use restricted maximum likelihood for Gaussian-only fits. The current REML pilot is deliberately narrow: all response rows must be Gaussian, observation weights are not supported, and <code>mi()</code> predictor models, predictor-informed latent($\dots$, $lv \sim x$) scores, and <code>Xcoef_fixed</code> maps are not supported. The observed fixed design must be full rank with positive residual degrees of freedom. Either missing-response policy may be used: masked cells (<code>miss_control(response = "include")</code>) contribute nothing to the likelihood, so the restricted likelihood is formed over the observed rows and matches the corresponding "drop" fit. The default FALSE keeps the historical ML fit. REML is validated for all-Gaussian fits only. For non-Gaussian families an opt-in escape hatch exists (<code>gllvmTMBcontrol(allow_nongaussian_reml = TRUE)</code>) that realises the Cox-Reid adjusted profile likelihood rather than an exact restricted likelihood – this is unvalidated: a hypothesis under test, not an inherited result. Users estimating variance components or heritability on non-Gaussian data should be aware that plain Laplace can carry substantial downward bias on latent standard deviations. An opt-in AGHQ engine exists (<code>gllvmTMBcontrol(aghq = k)</code>) with its own, separately-scoped evidence; AGHQ and REML = TRUE cannot be combined.
mesh	Optional mesh object from <code>make_mesh()</code>. Required for any <code>spatial_*</code> or <code>spatial()</code> term unless that term supplies its own <code>mesh =</code> argument. A supplied

	mesh with no spatial term is unused: the fit warns, and a raw fmesher/INLA mesh is rejected, rather than being silently ignored.
phylo_vcv	(legacy global) Optional tip-only $n_{\text{species}} \times n_{\text{species}}$ phylogenetic correlation matrix. The canonical syntax is <code>vcv =</code> inside each <code>phylo_*</code> () keyword. [Superseded] ; prefer <code>tree =</code> . The dense path inverts via <code>Matrix::solve()</code> , giving the same MLE as the sparse path but at $O(n^2)$ memory and $O(n^3)$ Cholesky cost. Use only when you have a Cphy in hand and no Newick tree (e.g. comparing against <code>nlme::corPage1</code>).
phylo_tree	(legacy global) Optional <code>ape::phylo</code> tree. The current syntax is to pass <code>tree =</code> inside each <code>phylo_*</code> () keyword (e.g. <code>phylo_latent(species, d = K, tree = tree)</code>); this argument is the older outer-level fallback and is slated for soft deprecation. When supplied (in either form) the <code>phylo_*</code> () terms build a sparse $\mathbf{A}^{-1}$ over tips + internal nodes natively, using only ape and Matrix – no MCMCglmm dependency . The construction is the deterministic Hadfield & Nakagawa (2010) sparse phylogenetic inverse (appendix eqs. 26-29), adopted from that method and ported from the sister package <code>drmTMB</code> (see <code>inst/COPYRIGHTS</code>). The result is $\sim 5n$ non-zeros for an n -tip tree, vs n^2 for the dense path. This is the recommended path at any n_{species} ; the speedup grows to $\sim 24\times$ at $n_{\text{species}} = 1000$.
known_V	Optional list of block-diagonal sampling-error matrices $\mathbf{V}_t$. Used by the <code>equalto()</code> two-stage workflow when sampling-error matrices are available from a prior stage of estimation.
lambda_constraint	Optional list with elements <code>unit</code> and / or <code>unit_obs</code> (plus <code>phy</code> , <code>spde</code> for structural levels), each an $n_{\text{traits}} \times d$ matrix of confirmatory loading constraints (galamm-style). NA entries are estimated; numerical entries are pinned. Upper-triangle entries are silently ignored because the engine's lower-triangular parameterisation already fixes those at zero. Default NULL uses the engine's exploratory lower-triangular convention. See confirmatory_lambda() to build the matrix from functional-group membership, or suggest_lambda_constraint() for a minimum statistical identification scaffold when you have no biological hypothesis. Deprecated legacy element names <code>B</code> and <code>W</code> are still accepted (with a one-shot soft deprecation message) and map to <code>unit</code> and <code>unit_obs</code> respectively.
Xcoef_fixed	Optional named numeric vector of fixed-effect coefficient constraints. Names must match the expanded fixed-effect design columns (<code>fit\$X_fix_names</code>); values must currently be 0, pinning those coefficients exactly at structural zero. Use this when a predictor is meaningful for some responses but should be fixed at zero for others. Native TMB fits use a parameter map; admitted engine = "julia" fixed-effect-X rows pass the same zero mask to GLLVM.jl. This is ML-only: <code>REML = TRUE</code> stops loudly. Native fitted objects still report all fixed-effect rows; pinned rows have <code>estimate = 0</code> , <code>std.error = NA</code> , and <code>status = "fixed"</code> in <code>tidy(fit, "fixed")</code> .
control	Output of <code>gllvmTMBcontrol()</code> .
missing	Output of miss_control() configuring missing-data handling. The default <code>miss_control()</code> (<code>response = "drop"</code> , <code>predictor = "fail"</code>) omits each missing response <i>cell</i> : a unit keeps every trait it does have, so this is not case-wise deletion. <code>miss_control(response = "include")</code> instead keeps those cells and

masks them out of the likelihood, preserving original-row accounting in `fit$missing_data`; it reaches the same optimum as the default and differs in which cells `predict_missing()` can return.

<code>impute</code>	Optional specification of the covariate model for a predictor declared missing with <code>mi(x)</code> in formula, used only when <code>missing = miss_control(predictor = "model")</code> . Supply a two-sided predictor-model formula (for Gaussian sugar, e.g. $x \sim z$) or an <code>impute_model()</code> object for an explicit predictor family. The default NULL is appropriate when no <code>mi()</code> term is present.
<code>silent</code>	Logical; suppress TMB and gllvmTMB chatter. Default TRUE.
<code>engine</code>	Character; "tmb" (default) fits with the native TMB engine, "julia" routes the fit through the experimental GLLVM.jl bridge fitting path via JuliaCall (see <code>R/julia-bridge.R</code>). The Julia path currently maps the unconstrained-ordination core (a single <code>latent()</code> block + per-trait intercepts) and errors on structures it does not yet support.
<code>ci_method</code>	Confidence-interval route requested at fit time for admitted engine = "julia" no-X rows. One of "none" (default), "wald", "profile", or "bootstrap". Native engine = "tmb" fits use <code>confint()</code> after fitting; non-default <code>ci_*</code> arguments therefore error unless engine = "julia". Grouped-dispersion rows, per-trait ordinal rows, response masks, mixed-family vectors, and fixed-effect-X rows remain gated.
<code>ci_level</code>	Nominal confidence level when <code>ci_method != "none"</code> on the Julia bridge.
<code>ci_nboot</code>	Number of parametric bootstrap replicates when <code>ci_method = "bootstrap"</code> on the Julia bridge.
<code>ci_seed</code>	Seed passed to the Julia bootstrap CI route.
<code>site</code>	(deprecated) alias for <code>unit</code> . Kept for backward compatibility. Use <code>unit = ...</code> in new code.
<code>species</code>	(deprecated) alias for <code>cluster</code> . Kept for backward compatibility. Use <code>cluster = ...</code> in new code.
<code>estimator</code>	Estimation criterion. "ml" (the default) preserves the ordinary maximum Laplace-likelihood route. "mspl" opts into the experimental maximum softly penalised Laplace-likelihood point estimator for the currently admitted complete-Bernoulli surface. This means exactly one ordinary <code>latent()</code> block with no free Ψ and $d = 1$ or 2 , standalone <code>spatial_indep()</code> , or standalone <code>spatial_latent()</code> with $d = 1$ or 2 ; one common logit, probit, or complementary-log-log link; complete single-trial responses; a full-rank resolved fixed-effect design; all-zero offsets; and the native Laplace engine. Unsupported structures fail before optimisation. MSPL does not activate automatically after a separation warning and never changes the default ML fit.

Details

source \ mode	independent	dependent	latent
<i>none</i>	<code>indep()</code>	<code>dep()</code>	<code>latent()</code>
<i>animal</i>	<code>animal_indep()</code>	<code>animal_dep()</code>	<code>animal_latent()</code>
<i>phylo</i>	<code>phylo_indep()</code>	<code>phylo_dep()</code>	<code>phylo_latent()</code>

<i>spatial</i>	<code>spatial_indep()</code>	<code>spatial_dep()</code>	<code>spatial_latent()</code>
<i>kernel</i>	<code>kernel_indep()</code>	<code>kernel_dep()</code>	<code>kernel_latent()</code>

The grid lists available syntax. Reliability depends on the family and uncertainty method. Gaussian `indep()` and `dep()` point fits are the narrow tested starting routes; structured-source and non-Gaussian routes need their named evidence boundary. General interval coverage is not claimed.

The three covariance modes (`indep` / `dep` / `latent`) encode `covstruct` intent across traits:

- **latent**: the **decomposition** mode $\Sigma = \Lambda\Lambda^T + \Psi$: a low-rank shared part plus a diagonal Ψ companion. Ordinary `latent()` carries Ψ by default; the `unique =` argument controls it (`latent(..., unique = FALSE)` for the loadings-only subset, `*_latent(..., unique = TRUE)` to fold the Ψ companion into a source-specific term).
- **indep**: the **marginal-only** mode: each trait gets its own variance, no cross-trait covariance.
- **dep**: the **full unstructured** mode: Σ is free with $T(T + 1)/2$ parameters via a Cholesky factor.

Plus the supporting `phylo_slope()` / `animal_slope()` (random slopes), `meta_V()` (known-V meta-analytic; `meta_known_V()` is a deprecated alias), and the engine-internal `propto()` / `equalto()` `covstructs` (used by the canonical keywords above; not typically called directly). See the [formula keyword grid article](#) for a one-paragraph tour of each cell.

`gllvmTMB()` parses the `glmmTMB`-style formula, converts wide `traits()` input to the same internal stacked-trait representation as explicit long data, and dispatches to the underlying TMB template. Covariance-structure terms (`latent()`, `indep()`, `propto()`, `equalto()`, `spatial()`) are processed by extending the formula parser and the TMB template.

Per the manuscript, when stacking traits one should set `dispformula = ~ 0` so that no implicit residual variance competes with structured `indep(0 + trait | ...)` terms. `gllvmTMB()` enforces this internally.

Multi-trial binomial. The TMB engine evaluates `dbinom(y, n_trials, p)` so binomial fits are not restricted to Bernoulli (`size = 1`). To pass a per-row trial count, use either:

- `cbind(successes, failures) ~ ...` on the LHS of the formula (canonical R / `glmmTMB` convention; recommended), or
- a flat response `y` with `weights = n_trials` (the alternative `glmmTMB` API).

Both interfaces produce identical fits. For Bernoulli data, omit `weights` and use a 0/1 response; the engine sets `n_trials = 1` and the likelihood is identical to the previous Bernoulli-only behaviour.

Missing responses and predictors. NA response cells are treated as unobserved unit-trait cells. Under the default `miss_control(response = "drop")`, they are dropped before the TMB likelihood is built; with `miss_control(response = "include")`, they are kept in the model data, masked out of the likelihood, and available through `predict_missing()`. For `cbind(successes, failures)` binomial responses, a row is treated as missing when either response component is missing. Observation weights, when supplied, are subset to retained likelihood rows before validation. Ordinary missing predictors, grouping variables, or fixed-effect design values still error because the model cannot construct a design row; explicitly modelled missing predictors use `mi(x)`, `miss_control(predictor = "model")`, and `impute = list(...)`. The modelled-`mi()` route is native Laplace only; the variational route refuses `mi()` predictors. Dense missing-response masks

have their own narrower VA fence and do not imply predictor-model support. These contracts are covered by the package's missing-data validation tests, both for responses and for predictors.

Delta (hurdle) families. `delta_lognormal()` and `delta_gamma()` use a *single* linear predictor for both components: presence is $\Pr(y > 0) = \text{invlogit}(\eta)$ and the positive-component mean is $E[y \mid y > 0] = \exp(\eta)$. This matches `sdmTMB`'s standard delta default (one fixed-effects matrix, one set of trait random effects). The shared-predictor scheme is the simplest hurdle formulation; a future release may decouple the two predictors so presence and abundance can have independent fixed and random effects. Only `type = "standard"` (the default) is currently wired in the multivariate engine; `type = "poisson-link"` is not implemented. Each delta family carries one per-trait dispersion of the *positive* component (no extra Bernoulli dispersion). The response must be non-negative.

Per-trait residual variance: when does it activate?:

"Residual" in a mixed-effects model is **scale-relative**: what counts as residual variance shifts as you add levels to the model. In a Gaussian fit without a per-row diagonal term, the residual is row-level noise captured by a single shared `sigma_eps`. Once you add a per-row `indep(0 + trait | obs)` term, the *row-level* residual is now T per-trait random-effect variances and `sigma_eps` is auto-suppressed to avoid double-counting. If you also add a site-level diagonal term on top, the row-level term remains the residual and the site-level term is now an additional, *higher-level* random effect, not a residual at all. The dispatch table below records this explicitly for the configurations the engine supports today.

Gaussian and lognormal responses have one residual scale parameter, `sigma_eps`. Ordinary Gamma responses instead carry a per-trait shape `phi_gamma` ($CV = 1 / \sqrt{\phi_gamma}$). Per-trait residual variances only appear if you explicitly add a per-row `indep(...)` term. The dispatch is automatic:

No per-row indep, Gaussian/lognormal present One shared `sigma_eps` across Gaussian/lognormal rows. *Not* per-trait.

No per-row indep, no Gaussian/lognormal rows `sigma_eps` is mapped off; the family's intrinsic dispersion handles the residual (for ordinary Gamma, `phi_gamma`).

`indep(0 + trait | g)` **where g has fewer levels than rows (e.g. g = "site")** `sigma_eps` is still estimated as the row-level residual; the diagonal term adds a per-trait random effect at level `g` on top.

`indep(0 + trait | obs)` **at the per-row level (one level per row), Gaussian/lognormal rows fitted** `sigma_eps` is auto-suppressed (mapped off, fixed at a tiny stabiliser); the T per-trait diagonal random effects *are* the residual. A one-shot `cli::cli_inform` fires at fit time announcing the auto-suppression.

`indep(0 + trait | obs)` **at the per-row level, non-Gaussian or mixed-family fit** Treated as observation-level random effects (OLRE). For Bernoulli traits the OLRE is statistically unidentifiable and is mapped off; for hurdle / delta families a warning is emitted (see "Per-family-aware OLRE selection" below).

Mnemonic: Gaussian/lognormal `sigma_eps` is the default; ordinary Gamma uses `phi_gamma`; per-row `indep` replaces the Gaussian/lognormal scalar residual with T per-trait residuals; non-per-row `indep` adds a higher-level random effect on top of the family residual; non-continuous families never carry `sigma_eps` regardless.

Per-family-aware OLRE selection. When `indep(0 + trait | <unit_obs>)` is at per-row resolution, i.e. one row per (`trait`, `unit_obs`) cell, the resulting per-trait random effects on the linear predictor are an observation-level random effect (OLRE). The engine now decides per trait what to do with the OLRE variance based on the trait's response family:

- **single-trial Bernoulli** (`binomial()`, all rows have `n_trials == 1`): `theta_diag_W[t]` and the corresponding `s_W` column are mapped off. The OLRE is statistically unidentifiable here (no within-cell replication) and the MLE is the trivial boundary $\sigma_W \rightarrow 0$; pinning the parameter at $\sigma_W \approx 10^{-6}$ removes the spurious free parameter and emits a one-shot informational message. Multi-trial binomial (via `cbind(succ, fail)` or `weights = n_trials`) is identifiable and fit normally.
- **delta / hurdle families** (`delta_lognormal()`, `delta_gamma()`): OLRE is fit as before, but a one-shot warning is emitted because the OLRE acts on the *shared* linear predictor of the hurdle and mixes presence and positive-component noise.
- **all other families**: OLRE is fit as before.

In a mixed-family fit (per-row family list) the per-trait decision applies only to the affected trait. References: Nakagawa & Schielzeth (2010) *Biol. Rev.* 85: 935-956; Nakagawa, Johnson & Schielzeth (2017) *J. R. Soc. Interface* 14: 20170213.

Value

A `gllvmTMB` object. With no covariance-structure terms in the formula the result has class `"gllvmTMB"` (single-response engine); with `latent()` or other `covstruct` sugar it has class `c("gllvmTMB_multi", "gllvmTMB")` (multi-trait engine). S3 methods are registered on the **subclasses**, not on bare `gllvmTMB`: for a multi-trait fit (`gllvmTMB_multi`), `tidy()`, `predict()`, `summary()`, `confint()` and `logLik()` are available, and `AIC()` / `BIC()` dispatch through `logLik()` only when the fitted point is an unpenalised native-Laplace ML maximum. AGHQ uses a different integration objective and a loading ridge returns a penalised MAP point, so ordinary likelihood comparisons do not apply. An MSPL result has leading class `"gllvmTMB_mspl"`; likelihood comparison, standard errors, intervals, profiles, and hypothesis-test methods fail closed because this experimental surface is point estimation only. Trait-level extractors such as `extract_ICC_site()` and `extract_communality()` are multi-trait only.

`vcov()` and `coef()` are available too; see [gllvmTMB_multi-vcov](#). `coef()` works on any fit; `vcov()` needs the `sdreport()` and raises the same typed errors `confint()` does when it is missing or non-finite.

History, 2026-08-04: this block used to say `vcov()` dispatched on `gllvmTMB`, which was **never true of any release**. It and `coef()` existed only for `gllvmTMB_va`, where they refuse. The wording was corrected first, then the two methods were added, which is why the promise now holds.

Dispersion parameters in the report slot

The fitted object's `$report` carries per-family dispersion quantities on the natural scale, each a vector of length `n_traits` indexed by trait (except `sigma_eps`, a single scalar). Their names follow the engine's internal parameterisation, **not** standard R distribution arguments; several are easy to misread. What each one IS, and how it maps to standard R arguments ($\mu = \exp(\eta)$ unless noted):

Family (report name)	What it is
gaussian, lognormal (<code>sigma_eps</code>)	residual SD / sdlog; one scalar shared by all gaussian <i>and</i> lognormal traits
Gamma (<code>phi_gamma</code>)	the shape , not a dispersion
nbinom2 (<code>phi_nbinom2</code>)	the NB size
nbinom1 (<code>phi_nbinom1</code>)	linear overdispersion, $\text{Var}(y) = \mu * (1 + \phi)$
tweedie (<code>phi_tweedie</code> , <code>p_tweedie</code>)	dispersion and power p in $(1, 2)$

Beta (phi_beta)	the precision
betabinomial (phi_betabinom)	precision of the Beta mixing
student (sigma_student, df_student)	the scale , not the SD
truncated_nbinom2 (phi_truncnb2)	the NB size; a separate vector from phi_nbinom2
delta_lognormal (sigma_lognormal_delta)	sdlog of the positive part
delta_gamma (phi_gamma_delta)	the CV of the positive part, not a shape

Objective provenance

A fitted object carries two different surfaces that both look like "the objective", and they answer different questions.

`report$joint_nll_unpenalized` and `report$joint_nll_penalized` are **conditional joint** quantities computed on the TMB tape (the negative log-likelihood of the data and random effects at their current values, before and after TMB-side MSPL penalty terms are added). They see only penalties applied *inside* the C++ template – the LA-MSPL penalties documented above – and never see anything computed at the R level.

`fit$objective_components` is the **marginal** surface: the value TMB's Laplace approximation integrates the random effects out of, decomposed into `likelihood_nll` (the unpenalised marginal negative log-likelihood), `ridge_penalty` (the R-level loading ridge from `aghq_ridge`, added *outside* the TMB objective – see `.gllvmTMB_penalised_gradient()`), `optimization_nll` (their sum, the objective the optimiser actually minimised), and `optimizer_reported` (the value `nlminb()` / `optim()` returned). This is the **only** place the R-level loading ridge appears; `report$joint_nll_*` cannot see it because it never reaches the TMB tape.

The two surfaces are **not expected to match**, on a ridged fit or otherwise, and a difference between them is not itself an error: they are conditional-joint and marginal quantities from different penalty regimes.

References

Sterzinger, P. and Kosmidis, I. (2023). Maximum softly-penalized likelihood for mixed effects logistic regression. *Statistics and Computing*, 33, 53. doi:10.1007/s11222023102173. This paper is the logistic mixed-effects antecedent for the fixed-effect Jeffreys component. The probit, complementary-log-log, reduced-rank GLLVM, and spatial penalties implemented here are gllvmTMB-specific extensions; that paper does not validate those extensions.

See Also

`screen_gllvmTMB()` for the opt-in fixed-design separation certificate to use before fitting; `estimator = "mspl"` is documented in this topic. `traits()` for wide data-frame formula input; `gllvmTMB_wide()` for wide matrix/data-frame input; `simulate_site_trait()` for generating recovery test data; `extract_Sigma()` for the unified post-fit covariance API; `gllvmTMB_diagnose()` for a one-stop convergence + identifiability health check; `ordinal_probit()` for the gllvmTMB-native ordinal threshold family.

Examples

```
set.seed(1)
sim <- simulate_site_trait(n_sites = 50, n_species = 8, n_traits = 3,
```

```

                                mean_species_per_site = 5)
# Long format: one response value per site-species-trait row.
fit_long <- gllvmTMB(
  value ~ 0 + trait + (0 + trait):env_1 + (0 + trait):env_2,
  data   = sim$data,
  family = gaussian(),
  trait  = "trait",
  unit   = "site_species"
)

# Wide format: one row per site-species, with the same three traits.
# `traits()` expands `1 + env_1 + env_2` to the explicit long terms above.
wide_values <- stats::reshape(
  sim$data[c("site_species", "trait", "value")],
  idvar = "site_species", timevar = "trait", direction = "wide"
)
wide <- merge(
  unique(sim$data[c("site_species", "env_1", "env_2")]), wide_values,
  by = "site_species", sort = FALSE
)
names(wide) <- sub("^value[.]", "", names(wide))
fit_wide <- gllvmTMB(
  traits(trait_1, trait_2, trait_3) ~ 1 + env_1 + env_2,
  data   = wide,
  family = gaussian(),
  unit   = "site_species"
)
summary(fit_long)
summary(fit_wide)

```

gllvmTMB_check_consistency

Run an advanced Laplace-consistency check

Description

`gllvmTMB_check_consistency()` is an advanced validation helper for the Laplace approximation, not a first-use diagnostic. Start with `check_gllvmTMB()` or `gllvmTMB_diagnose()`. Use this function when a fit looks numerically acceptable but you need a direct check of whether the approximate marginal score is centred.

Usage

```
gllvmTMB_check_consistency(fit, n_sim = 100L, seed = NULL, estimate = FALSE)
```

Arguments

`fit` A fit returned by `gllvmTMB()`.

<code>n_sim</code>	Integer number of simulate-evaluate replicates. Default 100L. Cost is roughly <code>n_sim</code> joint likelihood evaluations; budget 5-30 seconds on a Tier-1 fixture and proportionally longer for larger fits.
<code>seed</code>	Optional integer seed for the simulation RNG. NULL (default) draws a random seed.
<code>estimate</code>	Logical. When TRUE, also refits the model on each simulated dataset and reports the joint-score <code>p_value + bias</code> . Default FALSE (much faster; marginal-score check only).

Details

It simulates `n_sim` datasets from the fitted model under the joint parameter vector at the MLE, then tests whether the approximate marginal score function is centred at zero across the simulations. A non-centred score is a sign that the Laplace approximation is **unreliable** for this fit – typically because the random-effects posterior is far from Gaussian (saturated binomial / Beta, sparse counts, weakly identified random effects) or the data don't constrain the random effects well.

Scope: local Laplace-consistency diagnostics for a fitted model. This is a diagnostic signal rather than a substitute for bootstrap, profile, or external Bayesian calibration. Broader calibration evidence is not yet available.

This is a complementary signal to `sanity_multi()` (which checks Hessian definiteness, gradient magnitude, convergence flags) and to a target-explicit known-DGP study of rank-selection behaviour. Slower than the first-line checks – `n_sim` likelihood evaluations – but the only diagnostic that targets the **Laplace approximation itself** rather than the optimisation outcome or the parameter identification.

A large marginal-score p-value is an absence of evidence against centring in this finite simulation; it is not a certificate that the Laplace approximation is accurate. If the result is small (`p_value <= 0.05`) or large flagged-parameter list), consider:

- Refitting with a richer random-effects structure (so the conditional posterior is more Gaussian).
- Switching to a parametric bootstrap CI (which does not depend on the Laplace approximation being a good fit).
- Validating against `tmbsan::tmbsan(fit$tmbsan_obj)` (the audit's recommended Bayesian-comparison path).

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(value ~ 0 + trait +
               latent(0 + trait | site, d = 1),
               data = sim$data,
               trait = "trait",
               unit = "site")
res <- gllvmTMB_check_consistency(fit, n_sim = 50L, seed = 1)
res
res$marginal_bias
res$flagged_parameters
```

Value

An object of class `gllvmTMB_check_consistency` with components:

`$marginal_p_value` Joint chi-squared p-value across all marginal-score components. NA if TMB's information matrix could not be inverted (often the case on tiny fixtures).

`$marginal_bias` Named numeric vector: per-parameter bias (mean of the simulated score, normalised by its SE). Large absolute values flag specific parameters where the Laplace approximation is biased.

`$joint_p_value` Joint-score p-value. NA unless `estimate = TRUE`. Slower because each replicate refits.

`$flagged_parameters` Character names of parameters whose marginal bias exceeds $|\text{bias}| > 0.5$ (a heuristic; finite-`n_sim` noise can produce small spurious biases, but $|0.5|$ is generally beyond noise for `n_sim = 100`).

`$n_sim` The `n_sim` actually run.

`$threshold` The marginal-bias threshold used.

`$diagnostics` Character vector of one or more of: "no_centring_warning" (the finite simulation did not detect non-centring; this is not an accuracy certificate), "marginal_score_non_centred", "joint_score_non_centred", "information_matrix_singular" (TMB couldn't invert it; interpret with caution), "marginal_p_value_unavailable" (TMB returned NA without a captured warning; usually happens on tiny / weakly-identified fixtures).

`$raw` The full `TMB::checkConsistency()` return value in case the user wants the per-replicate gradient matrix.

`$call` `match.call()` of the invocation.

See Also

`sanity_multi()` (structural / convergence checks), `check_auto_residual()` (mixed-family safeguard), `gllvmTMB_diagnose()` (holistic fit summary), `TMB::checkConsistency()` (the underlying TMB call).

`gllvmTMB_diagnose`

Diagnose a fitted model and suggest next actions

Description

This is the human-readable diagnostic to call right after `fit <- gllvmTMB(...)`. It combines the quick numerical screen (`sanity_multi()`), the rotation identifiability advisory, and key biological summaries (correlation diagonals, ICCs, communalities) into a single report with explicit next-step hints for any WARN signal. Use `check_gllvmTMB()` when you need the same fit-health checks as a stable table for scripts or reports.

Usage

```
gllvmTMB_diagnose(
  object,
  gradient_thresh = 0.01,
  se_thresh = 100,
  big_corr_thresh = 0.5,
  verbose = TRUE
)
```

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>gradient_thresh</code> , <code>se_thresh</code>	Forwarded to <code>sanity_multi()</code> .
<code>big_corr_thresh</code>	Threshold above which a Sigma_B correlation off-diagonal is flagged as worth highlighting. Default 0.5.
<code>verbose</code>	If TRUE (default), prints the report. Always returns the structured list invisibly.

Details

Scope: first-line convergence, Hessian, standard-error, restart, and rotation diagnostics. It reports risks and summaries but does not replace profile, bootstrap, or simulation calibration. Target-explicit validation will decide which interval warnings can be promoted to broader guarantees.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = 2),
               data = dat, trait = "trait", unit = "site")
gllvmTMB_diagnose(fit)
```

Value

Invisibly a list with components: `sanity` (the `sanity_multi()` flags), `rotation` (rotation-advisory list), `Sigma_B`, `Sigma_W`, `ICC_site`, `communality_B`, `communality_W`, and `hints` (character vector of suggested actions).

See Also

`check_gllvmTMB()`, `sanity_multi()`, `suggest_lambda_constraint()`, `extract_Sigma()`, `extract_communality()`, `compare_dep_vs_two_psi()` / `compare_indep_vs_two_psi()` for identifiability cross-checks on the paired phylogenetic fit.

gllvmTMB_julia-methods

Methods for Julia bridge fits

Description

Small S3 surface for fits returned by `gllvmTMB(..., engine = "julia")` or `gllvm_julia_fit()`. These methods expose the flat bridge payload that has already passed the R admission gates. They are point-estimate summaries: prediction and ordinary response/Pearson residuals are in-sample retained-payload reconstructions only. Simulation is conditional on retained fitted values for admitted one-family rows and complete balanced mixed-family rows. Unit-tier covariance honors native `link_residual` scale semantics, and raw ordination accessors are routed; richer extractor parity remains a separate bridge row. Confidence intervals are routed for admitted no-X Gaussian, Poisson, Bernoulli binomial, NB2, NB1, Beta, and Gamma rows; response-mask CIs are routed for the same non-Gaussian rows when `X = NULL`; complete-response fixed-effect-X CIs are routed for Gaussian, Poisson, Bernoulli binomial, NB2, Beta, and Gamma rows. They may be requested at fit time through `gllvmTMB(ci_method = ...)`, or retrieved and recomputed through `confint()`. Predictor-informed latent(..., `lv = ~ x`) rows admit `ci_method = "wald"` at fit time; retained Wald payloads are surfaced as `std.error`, `lower`, and `upper` through `extract_lv_effects()` rather than `confint()`.

Usage

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

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

## S3 method for class 'gllvmTMB_julia'
coef(object, ...)

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

## S3 method for class 'gllvmTMB_julia'
fitted(object, type = c("response", "link", "prob", "class"), ...)

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

## S3 method for class 'gllvmTMB_julia'
```

```

simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata = NULL,
  condition_on_RE = TRUE,
  ...
)

## S3 method for class 'gllvmTMB_julia'
confint(
  object,
  parm,
  level = 0.95,
  method = c("stored", "wald", "profile", "bootstrap"),
  ci_nboot = 200L,
  ci_seed = 0L,
  ...
)

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

## S3 method for class 'summary.gllvmTMB_julia'
print(x, digits = 3, ...)

```

Arguments

object, x	A fit returned by <code>gllvmTMB(..., engine = "julia")</code> or <code>gllvm_julia_fit()</code> .
...	Unused.
newdata	Unsupported for Julia bridge fits; current prediction and simulation methods return in-sample values from the retained bridge payload.
type	Prediction or residual scale. For <code>predict()</code> and <code>fitted()</code> , "link" returns the fitted linear predictor and "response" applies the inverse link for supported non-ordinal bridge families. For per-trait ordinal bridge fits, "response" and "prob" return fitted category probabilities and "class" returns the modal category. For <code>residuals()</code> , "response" returns observed-minus-fitted residuals on the response scale and "pearson" divides by the family-specific standard deviation. Binomial rows use the observed proportion, y / N , on the response scale.
nsim	Number of replicate response vectors to draw for <code>simulate()</code> .
seed	Optional RNG seed for <code>simulate()</code> .
condition_on_RE	Logical for <code>simulate()</code> . The Julia bridge currently routes only conditional, in-sample simulation from retained fitted values, so FALSE stops with a not-yet-routed message.
parm	Optional integer or character vector of CI terms for <code>confint()</code> .

level	Confidence level requested by <code>confint()</code> . Stored Julia bridge payloads can only be read at their stored level; post-fit requests recompute the admitted Julia CI payload at the requested level.
method	CI route for <code>confint()</code> . "stored" reads an existing payload. "wald", "profile", and "bootstrap" recompute from retained bridge input for admitted no-X Gaussian, Poisson, Bernoulli binomial, NB2, NB1, Beta, and Gamma rows, including masked non-Gaussian rows when <code>X = NULL</code> . Complete-response fixed-effect-X recomputation is routed for Gaussian, Poisson, Bernoulli binomial, NB2, Beta, and Gamma rows. If omitted, <code>confint()</code> reads a stored payload when present and otherwise uses "wald" for current fits that retain their bridge input.
ci_nboot	Number of parametric bootstrap replicates when <code>method = "bootstrap"</code> .
ci_seed	Seed passed to the Julia bootstrap CI route.
digits	Number of digits printed by summary methods.

Value

`logLik()` returns a "logLik" object. `coef()` returns a named list of available point-estimate components. `confint()` returns a conventional confidence-interval matrix for stored or recomputed Julia CI payloads. `predict()` returns an in-sample data frame. Non-ordinal rows use `trait`, `unit`, and `est` columns; ordinal probability rows use `trait`, `unit`, `category`, and `prob`; ordinal class rows use `trait`, `unit`, and `est`. `fitted()` returns the in-sample fitted matrix with traits in rows and units in columns for non-ordinal rows, and an array of category probabilities or a modal-class matrix for ordinal rows. `residuals()` returns an in-sample residual matrix with the same shape as `fitted()` for non-ordinal rows and keeps masked response cells as NA. `simulate()` returns an `n_obs x nsim` matrix in the same trait-major cell order as `predict()` and keeps masked response cells as NA. `summary()` returns a list with header, coefficients, covariance, and status fields.

`gllvmTMB_multi-methods`

Methods on a fitted gllvmTMB model

Description

Standard model-object accessors for a multivariate fit returned by `gllvmTMB()`, whether the call started from wide `traits(...)` data or already-stacked long data. Internally the fit has class `gllvmTMB_multi`, which is what these S3 methods dispatch on, but you call `print(fit)` and `summary(fit)` as usual. Likelihood and inference accessors depend on the estimator, as described below.

Usage

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

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

```
## S3 method for class 'summary.gllvmTMB_multi'
print(x, digits = 3, ...)

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

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

Arguments

<code>x, object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>...</code>	Currently unused.
<code>digits</code>	Decimal digits in the printed summary. Default 3.

Details

- For ML fits, `print()` shows the active covstructures, the number of fixed effects, and the converged log-likelihood.
- For ML fits, `summary()` adds a fixed-effects table with SEs, the global and local trait correlation matrices, per-trait ICCs, and global / local communalities.
- For an unpenalised native-Laplace ML fit, `logLik()` is the converged maximum with `df = length(opt$par)` and `nobs` equal to the number of likelihood-contributing response cells. AGHQ has a distinct integration objective; a loading ridge is penalised MAP; and non-unit likelihood weights define an estimating objective. Ordinary likelihood comparison is unavailable on each of those restricted surfaces.
- For estimator = "mspl", `print()` and `summary()` identify the experimental softly penalised Laplace point estimator and show the unpenalised Laplace value at that point only as provenance. `logLik()`, AIC, BIC, likelihood-ratio tests, standard errors, intervals, and profiles fail closed because the point is not the ordinary likelihood maximum and repeated-sampling inference is not calibrated.

`nobs()` returns the number of **likelihood-contributing** observations – the observed-response cells. This equals `fit$missing_data$counts$likelihood_rows` and the `nobs` attribute of `logLik()`. Under the default `miss_control(response = "drop")` every fitted row is observed, so it equals `length(fit$tmb_data$y)`; under `response = "include"` the masked rows are excluded. Original-row counts live in `fit$missing_data`, never in `nobs()`.

Value

`print()` returns its input invisibly, after displaying the fit or summary. `summary()` returns a list of class `summary.gllvmTMB_multi` containing the model header, fixed-effect estimates, covariance summaries, and convergence diagnostics. `logLik()` returns a scalar of class `logLik` with `df` and `nobs` attributes for supported likelihood fits. `nobs()` returns the number of likelihood-contributing response cells as a numeric scalar. Estimator restrictions described below apply to these accessors.

gllvmTMB_multi-vcov *Fixed-effect estimates and covariance for a multi-trait fit*

Description

`coef()` returns the fixed-effect point estimates; `vcov()` returns their covariance matrix, taken from the fit's TMB `sdreport()`.

Usage

```
## S3 method for class 'gllvmTMB_multi'
coef(object, ...)

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

Arguments

<code>object</code>	A fitted multivariate model returned by <code>gllvmTMB()</code> .
<code>...</code>	Ignored, present for S3 consistency.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
coef(fit)
vcov(fit)
```

Value

`coef()`: a named numeric vector, one entry per fixed-effect term (`object$X_fix_names`).

`vcov()`: a square numeric matrix with those same names on both margins. Rows and columns for coefficients held fixed via `Xcoef_fixed` are NA – a fixed parameter was not estimated, so it has no sampling covariance.

Point estimates need no standard errors

`coef()` works on any fit, including one made with `gllvmTMBcontrol(se = FALSE)` – point estimates are this package's supported claim and do not depend on `sdreport()`. `vcov()` does depend on it, and raises the same typed errors `confint()` does when it is missing (`gllvmTMB_confint_no_sdreport`) or non-finite (`gllvmTMB_confint_nonfinite_se`), naming `standard_errors()` as the remedy in the first case. The conditions are shared deliberately: a caller that handles one should handle the other.

What this covariance is

The fixed-effect block of the single TMB `sdreport()` the fit already carries – a Wald covariance, with exactly the caveats any Wald quantity from this package carries. It is not a resampled or profiled quantity, and no interval built from it has certified coverage. `vcov()` refuses a fit with non-unit likelihood weights: that fit targets a weighted estimating objective and has no certified sandwich covariance.

See Also

`standard_errors()` to compute the `sdreport()` after fitting; `confint()` for intervals; `summary()` for a coefficient table.

`gllvmTMB_va-methods` *Methods for variational (integration = "va") fits*

Description

Methods for the object returned by `gllvmTMB(..., control = gllvmTMBcontrol(integration = "va"))`.

Usage

```
## S3 method for class 'gllvmTMB_va'
print(x, digits = 3, ...)

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

## S3 method for class 'summary.gllvmTMB_va'
print(x, digits = 3, ...)

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

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

## S3 method for class 'gllvmTMB_va'
confint(object, parm, level = 0.95, ...)

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

## S3 method for class 'gllvmTMB_va'
coef(object, ...)

## S3 method for class 'gllvmTMB_va'
```

```

residuals(object, ...)

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

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

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

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

## S3 method for class 'gllvmTMB_va'
predict(object, ...)

```

Arguments

x, object	A fit returned by <code>gllvmTMB()</code> with <code>control = gllvmTMBcontrol(integration = "va")</code> .
digits	Decimal digits in the printed output. Default 3.
...	Currently unused.
parm	Fixed-effect coefficient indices or names for <code>confint()</code> . By default all fixed effects are returned.
level	Nominal confidence level for fixed-effect VA-Wald intervals.

Details

This is an opt-in research route. Its objective is an **ELBO** – a lower bound on the log-likelihood – not a log-likelihood, and its inverse Hessian is not automatically calibrated frequentist uncertainty. Accordingly `logLik()`, `AIC()`, and `BIC()` remain undefined. `vcov()` and `confint()` expose only fixed-effect VA-Wald uncertainty computed from the profiled Schur information matrix. The returned values are explicitly labelled uncalibrated; raw-loading Wald intervals are not available.

The ordination surface is the exception: `extract_ordination()`, `getLV()`, `getLoadings()`, and `extract_loadings()` all work for a variational fit and return latent scores and loadings as POINT ESTIMATES. `getLV(se = TRUE)` additionally returns a variational *posterior* SD – not a standard error – and only when the fit's `eval_method` resolves to "gh". See the `integration` argument of `gllvmTMB()` for the measured accuracy of these point estimates.

Value

`print()` and `print.summary()` return their argument invisibly. `summary()` returns an object of class "summary.gllvmTMB_va". `nobs()` returns an integer count of unit-by-response cells. `vcov()` returns the profiled-Schur beta covariance matrix and `confint()` its nominal Wald intervals; both carry `calibrated = FALSE` and `uncertainty_basis = "VA-Wald profiled Schur information"` attributes.

gllvmTMBcontrol	<i>Control parameters for <code>gllvmTMB()</code></i>
-----------------	---

Description

Control parameters for `gllvmTMB()`

Usage

```
gllvmTMBcontrol(
  d_B = NULL,
  d_W = NULL,
  spde_mode = c("per_trait", "shared"),
  n_init = 1L,
  optimizer = c("nlminb", "optim"),
  optArgs = list(),
  init_jitter = 0.3,
  init_strategy = c("default", "single_trait_warmup"),
  start_method = list(method = NULL, jitter.sd = 0),
  start_from = NULL,
  se = TRUE,
  verbose = FALSE,
  integration = c("laplace", "va"),
  va_H = 7L,
  va_eval_method = c("auto", "jj", "gh"),
  aghq = FALSE,
  aghq_iter_cap = 1L,
  aghq_n_adapt = 400L,
  aghq_ridge = 2,
  aghq_continuation = TRUE,
  aghq_shift_tol = 1e-04,
  aghq_grad_tol = 1e-04,
  aghq_grad_tol_rel = 1e-06,
  aghq_multistart = TRUE,
  aghq_f_tol = 1e-09,
  aghq_escalate_patience = 3L,
  aghq_rho_min = 1/64,
  warn_runaway = TRUE,
  allow_nongaussian_reml = FALSE,
  loading_ridge = NULL,
  ...
)
```

Arguments

<code>d_B, d_W</code>	Latent dimensions for the between-unit and within-unit reduced-rank components. Set to a positive integer to enable <code>latent()</code> / <code>indep()</code> covariance structures at the corresponding tier.
-----------------------	---

spde_mode	"per_trait" (default) fits one independent SPDE field per trait when a spatial() term is present; "shared" fits one shared SPDE field with trait-specific scalar loadings.
n_init	Number of random-start replicates. Reduced-rank GLLVMs are often multi-modal because the latent-factor likelihood has many equivalent local maxima; running several restarts with different starting values and keeping the fit with the lowest -logLik reduces the risk of settling on a suboptimal solution. Default 1 (single fit). Increase to 5–10 for two-level rr models.
optimizer	One of "nlopt" (default) or "optim". Use the latter together with optArgs for finicky two-level rr fits.
optArgs	A list of arguments passed to the optimiser. For optim the most useful is list(method = "BFGS").
init_jitter	Standard deviation of N(0, sigma) jitter applied to the starting parameter vector across the n_init restarts. Default 0.3.
init_strategy	One of "default" (current behaviour) or "single_trait_warmup". The warmup option fits an intercept-only univariate GLM per trait, using that trait's family, and seeds the matching log_phi_* entries before MakeADFun(). Recommended for count families (especially nbinom2) where the default initialisation can leave the optimiser walking the (ψ_t, ϕ_t) trade-off ridge. No effect for traits whose family doesn't carry a phi parameter (Gaussian, Poisson, binomial). Default "default".
start_method	Optional reduced-rank starting-value method, modelled after glmmTMB::glmmTMBControl(start_method = ...) but extended for two-level gllvmTMB models. Use list(method = "res", jitter.sd = 0.2) to seed latent() loadings and latent scores from a reduced-rank decomposition of fixed-effect residuals. Use list(method = "indep") to first fit the matching independent diagonal GLMM/GLLVM and copy its estimated fixed effects, per-trait variance starts, and random effects into the full latent covariance fit. jitter.sd adds Normal jitter to residual-start latent scores. Default list(method = NULL, jitter.sd = 0) keeps the historical starts.
start_from	Optional fitted gllvmTMB object, usually a simpler model such as one latent() tier or an independent diagonal model (indep()). Any estimated TMB parameters with shapes matching the current model are copied into the starting parameter list before optimisation. This implements the "fit simpler, then use it as starting values" workflow recommended for complex reduced-rank models.
se	Logical; if TRUE, compute TMB::sdreport() after optimisation so Wald standard errors and Hessian diagnostics are available. If FALSE, skip standard-error calculation and return the point-estimate fit with sd_report = NULL and a diagnostic note. This is useful for hard models where the point estimate is needed and uncertainty will be obtained by bootstrap or profile methods. Default TRUE.
verbose	If TRUE, prints a one-line summary per restart so the user can see which seed led to the winning fit. Default FALSE.
integration	Which method evaluates the latent-variable integral. "laplace" (default) is the Laplace approximation. For an unpenalised native-Laplace fit, logLik(), AIC(), BIC() and likelihood-ratio tests have their ordinary maximum-likelihood interpretation. AGHQ-refined fits also expose these methods, but use a different

integration objective and must not be compared across engines; with a loading ridge, the fitted point is penalised MAP and ordinary likelihood-comparison interpretations do not apply.

"va" selects an opt-in **research** route whose objective is an evidence lower bound, not a marginal likelihood. Fixed-effect `vcov()` and `confint()` use profiled-Schur VA-Wald information and are labelled `calibrated = FALSE`; `getLV()` can return a variational posterior SD, which is not a frequentist random-effect standard error. No nominal coverage claim is made. A bound must not be compared across ranks or models, so it cannot be used for model or rank selection.

It is admitted only inside the Gate-E implementation/light-fit region: `latent(..., unique = FALSE)`, the 18 scalar family/link cells in the response-family registry, `d` up to 2, up to 80 responses, at least 100 units, and the native TMB engine. A request outside that region is an **error**, not a warning. The `d` limit is retained from the historical explicit-JJ preregistered recovery gate: `d = 4` was measured and refused because, with few responses, the planted axes collapsed more often than that gate allowed. This is an implementation admission fence, not recovery evidence for the current automatic GH-H7 route. In the current 36-cell family-by-rank campaign only the Poisson-log $q = 5$ cell passed the overall point route, and that rank lies outside the public `d ≤ 2` fence; VA therefore remains experimental and `calibrated = FALSE`. It cannot be combined with `aghq`, which is an alternative evaluation of the same integral rather than an additional layer.

Offering this value advertises nothing about its accuracy.

[Experimental] The **ordination surface is available** for a "va" fit: `extract_ordination()`, `getLV()`, `getLoadings()` and `extract_loadings()` return latent scores and loadings as POINT ESTIMATES. Fixed-effect `confint()` and `vcov()` expose uncalibrated VA-Wald uncertainty; `getLV(se = TRUE)` returns a variational *posterior* SD, not a frequentist standard error, and only when `eval_method` resolves to "gh". Loading/covariance intervals remain unavailable.

Historical explicit-JJ evidence. Recovery of the ordination below was measured with the earlier JJ evaluator against planted truth at the admitted cells (`d = 2, 8` responses, `n = 150` and `400`, 50 seeds per cell), with the Laplace route run on the same simulated data as a control. This campaign predates the current automatic Gauss-Hermite default and must not be used as evidence for the accuracy of the current GH-default route:

- **Gaussian:** the loading-implied variance is recovered at 0.98–1.02 of truth, and the mean paired difference in latent-score correlation against Laplace is below $2e-07$ at both `n`.
- **Poisson:** also 0.98–1.02 of truth, with a paired latent-score difference of $\sim 4e-04$, negligible in magnitude but statistically detectable, and the loading scale sits $\sim 1\%$ below Laplace.
- **Binomial:** latent scores recover at about $r = 0.59$. That ceiling is a limit of **binary data** at these cells rather than of the historical JJ route: the Laplace default reaches about $r = 0.56$ – 0.59 on the same data, and a third estimator (Gauss-Hermite) lands in the same place. The loading SCALE, however, is biased low (0.58–0.67 of truth) and this **is** a real bias relative to Laplace, whose median brackets 1. Read a binomial ordination's axes and relative positions, not the absolute magnitude of its loadings.

The comparison is not effort-matched: the historical JJ arm ran four starts behind an agreement-and-gradient health gate, the Laplace arm a single start admitted on a positive-definite Hessian.

"va" returns an object of class "gllvmTMB_va" (see [gllvmTMB_va-methods](#)) rather than an ordinary fit, so that every method which would treat its objective as a likelihood fails loudly instead of returning a number. Because the engine runs its own multi-start and optimiser policy, the search settings of [gllvmTMBcontrol\(\)](#) (`n_init`, `optimizer`, `optArgs`, `start_from`, `init_*` and `se`) have no effect on this route. Any model structure the route cannot represent (a latent term away from the unit grouping, a constrained ordination, an offset, weights, REML, `lambda_constraint`, `Xcoef_fixed`, or a further random effect) is an **error**, never a silent omission.

There is no "eva" value. The EVA engine exists in this package and is reachable as a research route, but it is not wired to [gllvmTMB\(\)](#), and an argument value that could only ever raise an error would advertise a capability the package does not have. No general coefficient, covariance, interval, or calibration claim is made for EVA. Existing measurements are research diagnostics only and do not establish a public estimator route.

va_H	Gauss-Hermite quadrature order for the variational route, a single odd integer ≥ 3 (default 7). Ignored unless <code>integration = "va"</code>. Quadrature dominates the cost of a variational fit and the work is linear in H, so this is the main speed control on that route. On a binomial probit cell with 20 traits, $H = 7$ was indistinguishable from $H = 61$ in both the recovered loading scale and the recovered linear predictor, at two and at five latent dimensions, while running 3.4-6.7 times faster. $H = 5$ was measurably worse at five latent dimensions. Gate E therefore promoted $H = 7$; $H = 61$ remains available as a diagnostic.
va_eval_method	How the variational route evaluates the per-observation expectation. "auto" (default) selects Gauss-Hermite quadrature for every admitted scalar family/link cell and is what you want unless you are comparing methods. "gh" uses Gauss-Hermite quadrature. "jj" uses the Jaakkola-Jordan bound, which is defined only for pure binomial-logit fits and is substantially cheaper. The two differ in more than speed: on binary data the bound's recovered loading magnitudes are biased downward and do not improve as the sample grows, whereas quadrature's converge. Latent-variable <i>scores</i>, and therefore ordinations, are effectively unaffected by the choice. Ignored unless <code>integration = "va"</code>.
aghq	Adaptive Gauss-Hermite quadrature for the between-unit latent block. FALSE (default) fits by Laplace approximation. A positive integer requests that many quadrature nodes. "auto" lets the package decide, and it declines to Laplace whenever the model is ineligible or the expected gain does not justify the cost. Opt-in and experimental: no capability claim is made for quadrature-fitted models, and eligibility is narrow: the random part must be a single ordinary <code>latent()</code> block on the unit tier.
aghq_iter_cap	Optimiser iteration budget for the first adaptation pass. Default 1L. Later passes escalate when continuation is enabled.
aghq_n_adapt	Maximum number of adaptation passes. Default 400L.

aghq_ridge	Ridge penalty on the loadings, as the scale tau; Inf disables it. Default 2. Naming a numeric value explicitly also makes the Laplace + ridge control reachable without quadrature. The default does not penalise an ordinary Laplace fit unless the caller explicitly names aghq_ridge. AGHQ itself is opt-in, but once enabled it uses the default tau = 2 penalised MAP route unless Inf is requested. "auto" is an opt-in experimental route for pure single-trial Bernoulli models with one ordinary unit-tier latent() block. It uses an unpenalised 9-node multi-start AGHQ pilot, then $\tau = \min(6, \max(1, \ \Lambda_{\text{pilot}}\ _F / \sqrt{pq}))$. If the pilot or scale-aware final fit is unusable, the returned fit transparently falls back to the shipped tau = 2 route; inspect <code>fit\$aghq\$ridge_auto</code> . Evidence supports failure/runaway avoidance in this scope, not a broad loading-accuracy improvement. The measured grid used a logit link with $p = 6$, $q = 2$, and $n = 100$, 400 , or 1600 ; other links and dimensions are extrapolations. The "auto" route always uses 9-node multi-start AGHQ for both pilot and final fits; a conflicting node/start control is replaced with a warning, while an explicit <code>aghq = FALSE</code> is incompatible.
aghq_continuation	If TRUE (default), the adaptation loop may raise <code>aghq_iter_cap</code> across passes. FALSE pins the cap and disables escalation.
aghq_shift_tol, aghq_grad_tol, aghq_f_tol	Convergence tolerances for the adaptation loop: the quadrature-point shift, the gradient at the adaptation point, and the change in objective. Defaults $1e-4$, $1e-4$, $1e-9$.
aghq_grad_tol_rel	Relative gradient tolerance, tested as $\max \text{grad} / \max(1, \text{objective})$. Default $1e-6$. The gradient of a likelihood summed over many observations grows with the sample, so the absolute <code>aghq_grad_tol</code> on its own becomes unreachable at scale. The two are combined with OR, so this can only ever admit more fits as converged, never fewer; set it to 0 to test the absolute tolerance only. Read <code>fit\$aghq\$converged</code> for the verdict, not <code>fit\$opt\$convergence</code> , which on the quadrature path records the optimiser's per-pass iteration cap and so reports a limit even on a healthy fit.
aghq_multistart	If TRUE (default), the quadrature is run from two starting points: the Laplace optimum and a data-driven alternative that uses no knowledge of the truth. The fit with the better final objective is kept. FALSE uses the Laplace warm start only, reproducing the behaviour of earlier versions. The extra start costs one additional adaptation run, and exists because the Laplace optimum is sometimes itself a runaway that the quadrature would otherwise inherit.
aghq_escalate_patience	Number of accepted passes before the iteration cap is escalated. Default 3L.
aghq_rho_min	Minimum trust-region-style acceptance ratio for a pass. Default $1/64$.
warn_runaway	If TRUE (default), warn once per session when a binomial latent-variable fit triggers the package's existing runaway-loading diagnostic. Set FALSE to silence the fit-time warning; the diagnostic remains available through <code>gllvmTMB_diagnose()</code> . The same switch also governs the multinomial contrast-degeneracy warning (collapsed contrast variance, rail-correlated contrasts, or a collapsed spatial range),

which uses its own once-per-session slot so neither family's warning can suppress the other's. Ordinal fits emit no fit-time warning: that row's arms ship disarmed because no threshold met its calibration targets, so it reports statistics through `check_gllvmTMB()` only.

<code>allow_nongaussian_reml</code>	Permit REML = TRUE for non-Gaussian families. Default FALSE; the restricted likelihood is defined by a Gaussian marginalisation, so this is an escape hatch, not an endorsement.
<code>loading_ridge</code>	Integration-neutral alias for <code>aghq_ridge</code> . Use this spelling to request the same loading MAP penalty on a Laplace fit without suggesting that the penalty belongs to AGHQ. The default NULL preserves historical unpenalised Laplace fits. Supply at most one of <code>loading_ridge</code> and <code>aghq_ridge</code> ; neither may be combined with <code>estimator = "mspl"</code> .
<code>...</code>	Reserved for future use. Currently ignored with a warning.

Details

Recommended workflow for two-level latent models. Reduced-rank likelihoods are notoriously multimodal in two-level fits like `value ~ 0 + trait + latent(0+trait|ID, d_B) + latent(0+trait|obs_ID, d_W)`. The recommended workflow is:

1. Run several restarts (`n_init = 5` to `10`) and keep the one with the lowest `-logLik`.
2. If that fails to converge cleanly, switch the optimiser to `optim` with BFGS: `gllvmTMBcontrol(optimizer = "optim", optArgs = list(method = "BFGS"))`.
3. `start_method = list(method = "res", jitter.sd = 0.2)` is **soft-deprecated as of 0.6.0** and warns once per session. It still fits: it decomposes the fixed-effect residual matrix into starting values for Λ and the latent scores. Do not reach for it.

It was retired on measurement. Across 89 simulated fits (Gaussian, Poisson and negative-binomial, $d = 1$ to 3 , three and five traits), it was **never materially better** than the default start (its three best margins were 0.07 , 0.29 and 0.66 log-likelihood units) and was materially worse eight times, once by 14.6 units. At $d \geq 2$ it made no difference at all. Every failure reported convergence `== 0` and a positive-definite Hessian on both sides, so the worse fit was silent, and restarts did not detect it: in one such fit all five `n_init = 5` restarts returned the same worse optimum.

If you use it anyway, compare `fit$opt$objective` against a default-start fit and keep the lower value.

4. For Gaussian two-level models, prefer `start_method = list(method = "indep")` or manually fit a simpler model and pass it through `start_from = simpler_fit`. This is a GLMM warm start rather than a fixed-effect-only GLM residual start.

Value

A `gllvmTMBcontrol` list.

Examples

```
# gllvmTMBcontrol() is a pure-R constructor: it builds and validates a
# control list without fitting anything.
```

```

gllvmTMBcontrol()                                # historical defaults

# Multi-start to guard against multimodal reduced-rank likelihoods, with
# jitter on the starting parameter vector across restarts.
gllvmTMBcontrol(n_init = 3, init_jitter = 0.2)

# Switch the optimiser to optim + BFGS for finicky two-level rr fits.
gllvmTMBcontrol(optimizer = "optim", optArgs = list(method = "BFGS"))

# Experimental, opt-in scale-aware ridge for the evaluated Bernoulli AGHQ
# scope. The route uses a fixed 9-node multi-start rule; this is not an
# interval-calibration certificate.
gllvmTMBcontrol(aghq_ridge = "auto")
# The same fixed loading MAP penalty, named independently of integration.
gllvmTMBcontrol(loading_ridge = 2)

# Experimental scalar variational route: public auto uses GH with H = 7.
gllvmTMBcontrol(integration = "va")
# H = 61 remains a diagnostic; JJ is explicit binomial-logit only.
gllvmTMBcontrol(integration = "va", va_H = 61, va_eval_method = "gh")

```

impute_model

Predictor model for a missing covariate used inside mi()

Description

`impute_model()` wraps the model for a covariate declared missing with `mi()` in a `gllvmTMB()` formula. A bare two-sided formula in the `impute` argument – for example `impute = list(x = x ~ z)` – is sugar for a Gaussian predictor model; use `impute_model()` when an explicit predictor family is wanted.

Usage

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

Arguments

formula	Two-sided predictor-model formula. The left-hand side must be the same variable used inside <code>mi()</code> .
family	Predictor-model family. Supported values are <code>gaussian()</code> , <code>binomial(link = "logit")</code> , <code>cumulative_logit()</code> , and <code>categorical()</code> . Gaussian predictors support fixed, grouped-intercept, and phylogenetic- intercept covariate models; discrete predictors are fixed-effect only.

Details

The current interface fits one modelled missing predictor at a time. A Gaussian predictor (`gaussian()`, or a bare formula) is treated as a latent quantity integrated out by the Laplace approximation and may use fixed covariates, one grouped random intercept, or one phylogenetic structured intercept. A Gaussian predictor that lives at a coarser observed level can declare that level with `mi_group(group)` in the predictor formula. Discrete predictors are fixed-effect only: `binomial(link = "logit")` uses exact two-state summation, `cumulative_logit()` uses exact ordered K-state summation, and `categorical()` uses exact unordered K-state softmax summation. Other predictor families are rejected explicitly.

Value

A `gllvmTMB_impute_model` object for the `impute` argument of `gllvmTMB()`.

See Also

`gllvmTMB()` for the `impute =` argument, `cumulative_logit()` for an ordered categorical predictor model, and `miss_control()` for the `predictor = "model"` switch.

Examples

```
impute_model(x ~ z)
impute_model(score ~ z, family = cumulative_logit())
```

imputed

Extract fitted missing-predictor values from a gllvmTMB fit

Description

For a `gllvmTMB()` fit with a modelled missing predictor (`mi(x)` plus `missing = miss_control(predictor = "model")`), `imputed()` returns a fitted summary of each missing predictor value. For a **continuous Gaussian** predictor (`impute_model(x ~ z)`), the estimate is the conditional mode with a local Gaussian/Laplace conditional standard error from the joint Hessian. For a **binary** predictor (`impute_model(x ~ z, family = binomial())`), the estimate is the conditional probability $P(x = 1 \mid y, z)$ from the exact two-state marginalisation, `std_error` is NA (the discrete route reports a fitted distribution, not a single mode with a Hessian SE), and `uncertainty_status` is `"discrete_no_se"`.

Usage

```
imputed(object, ...)

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

Arguments

object	A gllvmTMB fit.
...	Reserved for future extractor options.
variable	Optional missing-predictor name. The default uses the only modelled missing predictor in the fit.
rows	Which units to return. "missing" (default) returns only the units whose predictor value was missing; "all" returns every unit, with observed predictor values labelled as observed.
se	Logical; include conditional standard errors when the fit contains a successful <code>TMB::sdreport()</code> result.

Details

The missing predictor lives at a declared unit or coarser grouping level, so `imputed()` returns one row per missing level value (not one per long-format model row). The Gaussian estimates are conditional modes, not conditional means, multiple- imputation draws, or posterior summaries.

Join keys and row identifiers. `level` names the unit or coarser group at which the missing predictor lives, and `level_id` gives its value. Join results back to the data using that pair. The `original_row` and `model_row` columns are **latent-level ordinals** (the 1-based index of the missing unit / group in the covariate model's latent level), *not* indices into the rows of the data frame you passed to `gllvmTMB()`. They coincide with original data rows only in the narrow wide-traits() case where each unit is a single row; off the wide path (long data, or a coarser `mi_group()` level) they index the latent level, not the data. This differs from `predict_missing()`, whose `original_row` maps to the supplied long- or wide-data row.

Value

A data frame with `variable`, `level`, `level_id`, `original_row`, `model_row`, `observed`, `estimate`, `std_error`, `source`, and `uncertainty_status`. `level_id` is character so the same column can hold unit, group, or phylogenetic-tip labels.

indep	<i>Per-trait marginal variance: <code>indep(0 + trait g)</code></i>
-------	---

Description

Canonical name for the **always-alone marginal-only** per-trait diagonal variance term, producing $\Sigma = \text{diag}(\sigma_t^2)$ with identity off-diagonals.

Usage

```
indep(formula, common = FALSE)
```

Arguments

formula	$0 + \text{trait} \mid g$ style formula (LHS is the trait factor, typically $0 + \text{trait}$; RHS is the grouping factor).
common	FALSE (default) for trait-specific marginal variances; TRUE to tie all traits to one shared variance at this grouping tier.

Details

The package's covstruct API organises around two mutually exclusive modes, distinguished by convention:

Decomposition (latent by default) Shared cross-trait covariance plus the default trait-specific Psi companion: $\Sigma = \Lambda\Lambda^T + \Psi$.

Marginal (indep standalone) Per-trait total variance with no cross-trait decomposition: $\Sigma = \text{diag}(\sigma_t^2)$.

Use `indep()` when you want to commit to the marginal-only interpretation explicitly. Ordinary `latent()` now carries the diagonal Psi companion by default when you want the cross-trait decomposition.

For a scalar marginal-only tier with one variance shared by all traits, use `common = TRUE`. This is the non-deprecated standalone replacement for the legacy scalar diagonal spelling when no `latent()` term is paired on the same grouping.

Mutual exclusion with `latent()`:

Combining `indep(0 + trait | g)` with `latent(0 + trait | g, d = K)` on the same grouping is **over-parameterised** (the model cannot decide whether trait variance lives in the shared component or the marginal component). The parser raises a `cli::cli_abort()` in this case. Combining `indep` with `unique` on the same grouping is similarly redundant and also errors.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
# Marginal-only fit (per-trait variance, identity correlation):
fit <- gllvmTMB(value ~ 0 + trait + indep(0 + trait | site),
               data = df, trait = "trait", unit = "site")

# Scalar marginal-only fit (one shared variance across traits):
fit <- gllvmTMB(value ~ 0 + trait + indep(0 + trait | site, common = TRUE),
               data = df, trait = "trait", unit = "site")

# ERROR: indep + latent on the same grouping is over-parameterised.
# gllvmTMB(value ~ 0 + trait +
#         indep(0 + trait | site) +
#         latent(0 + trait | site, d = 2), data = df)
```

Value

A formula marker; never evaluated.

See Also

[latent\(\)](#), [phylo_indep\(\)](#), [spatial_indep\(\)](#), [extract_Sigma\(\)](#).

isdm_source

Declare one iSDM source observation model

Description

Wraps an admitted observation law with the source-specific fixed-effects formula that describes recording effort or observation bias. The ecological formula remains in [gllvmTMB\(\)](#); `observation` contributes columns only on rows from this declared source. It is therefore not a second ecological process and does not turn relative intensity into abundance, occupancy, or detectability.

Usage

```
isdm_source(family, observation)
```

Arguments

family	An admitted family law, currently poisson() with its log link or binomial() with link = "cloglog".
observation	A one-sided formula for source-specific observation effects, for example <code>~ access + popdens</code> or <code>~ observer + method</code> . Do not add <code>0 +</code> merely to make source formulas identifiable: the wrapper keeps the ecological design first and automatically reference-codes any aliased source-observation columns.

Value

An internal source declaration accepted by [isdm_sources\(\)](#).

Examples

```
isdm_source(poisson(link = "log"), observation = ~ access + popdens)
```

isdm_sources

Declare the sources of a multi-source integrated model

Description

[Experimental]

Usage

```
isdm_sources(...)
```

Arguments

... Two or more named arguments; each name is a source label and each value is a bare admitted observation law or an `isdm_source()` declaration. At least one source must be a count stream: the detection arm's offset is admitted only alongside a count arm sharing the same intensity, so an all-detection declaration is refused here rather than failing later.

Details

Builds the family argument for an integrated species-distribution fit in which any number of named observation sources – an opportunistic portal stream, digitised literature records, checklists, a structured survey – share one ecological linear predictor while each source keeps its own observation law.

Each argument is named for a source and gives either a bare observation law or `isdm_source()` with its source-specific observation formula. Two laws are admitted, because both express the observation as a thinning of one shared intensity: `poisson()` (a count stream; the offset is log effort) and `binomial("cloglog")` (detection/non-detection; the offset is log support, and the complementary log-log link is what makes a detection consistent with the same intensity that generates the counts). Other links and families are refused: a logit detection model, for instance, does not share a scale with a count arm merely because both converge.

The data must carry an `isdm_source` column whose values are exactly the declared source names, and – when the declaration mixes the two laws – every trait must be observed by every declared source. Everything such a fit reports is relative intensity: presence-only data cannot identify absolute abundance, occupancy, or detectability, and none are estimated. This interface is experimental and may change.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
## `dat` is long, one row per (cell, species, source), with an isdm_source
## column naming each row's source and log_support its known effort/area.
fit <- gllvmTMB(
  value ~ 0 + trait + trait:env +
    offset(log_support) + latent(0 + trait | cell_id, d = 1),
  data = dat, trait = "trait", unit = "cell_id", family = fam
)
```

Value

A mixed-family list understood by `gllvmTMB()`, with `family_var = "isdm_source"`. (The `isdm_source_laws` attribute is informational only: internal validation rebuilds the declaration from the list's names and laws, which survive reordering; the attribute does not.)

Examples

```
fam <- isdm_sources(
  gbif = isdm_source(poisson(), observation = ~ access + popdens),
  survey = isdm_source(poisson(), observation = ~ observer + method)
)
names(fam)
```

kernel_latent

*Generic dense-kernel covariance keywords***Description**

kernel_latent() fits a named random-effect tier with a user-supplied between-unit covariance matrix K . For one named tier, unique = TRUE adds a kernel-structured diagonal Ψ companion; the default FALSE is loadings-only. Two or more named kernel_latent() tiers may share the same grouping levels, each with its own K , loading matrix, and latent field. Multi-kernel fits are loadings-only: unique = TRUE and kernel_dep() are not available in that combination. Similar kernels can be weakly separable, so use [diagnose_kernel_separability\(\)](#) before assigning a distinct interpretation to each component.

A matrix returned by [make_cross_kernel\(\)](#) is treated as a fixed supplied covariance. The model does not estimate its association structure or bridge strength. [extract_Gamma\(\)](#) returns descriptive point summaries; it does not provide calibrated intervals or establish a causal evolutionary process.

Usage

```
kernel_latent(unit, K, d = 1, name = "kernel", unique = FALSE)
```

```
kernel_indep(unit, K, name = "kernel", common = FALSE)
```

```
kernel_dep(unit, K, name = "kernel")
```

Arguments

unit	Unquoted grouping column whose levels align with rownames(K).
K	Numeric dense positive-semidefinite covariance or correlation matrix aligned to the grouping levels.
d	Integer latent rank for kernel_latent().
name	Character scalar used as the extractor level, e.g. <code>extract_Sigma(fit, level = "cross")</code> .
unique	Logical; TRUE auto-includes the kernel-structured diagonal trait-specific Ψ companion for a single dense-kernel tier. The default FALSE preserves the loadings-only subset.
common	FALSE (default) for a separate dense-kernel variance per trait (kernel_indep); TRUE ties all traits to one shared variance, the canonical spelling for the soft-deprecated kernel_scalar() .

Value

A formula marker; never evaluated as a regular R function.

Examples

```

A <- diag(5)
dat <- data.frame(
  unit = factor(rep(paste0("u", 1:5), each = 2), levels = paste0("u", 1:5)),
  obs = factor(seq_len(10)),
  y1 = rnorm(10),
  y2 = rnorm(10)
)
rownames(A) <- colnames(A) <- levels(dat$unit)
fit <- gllvmTMB(
  traits(y1, y2) ~
    1 + kernel_latent(unit, K = A, d = 1, name = "known",
                      unique = TRUE),
  data = dat,
  unit = "obs",
  cluster = "unit",
  family = gaussian()
)
extract_Sigma(fit, level = "known")

```

kernel_scalar

Dense-kernel one-shared-variance covariance: kernel_scalar()**Description**

kernel_scalar() is the dense-kernel analogue of [scalar\(\)](#) / [phylo_scalar\(\)](#): **one variance shared by every trait**, coupled between grouping levels by the supplied dense matrix K ($\Sigma = \sigma^2 \mathbf{I}_T$, $\text{Cov}(b_{gt}, b_{g't}) = \sigma^2 K_{gg'}$). It uses the same phylo-equivalent diagonal engine path as [kernel_indep\(\)](#), with the per-trait variances tied to a single shared parameter, so [extract_Sigma\(\)](#) exposes it under level = name.

Usage

```
kernel_scalar(unit, K, name = "kernel")
```

Arguments

unit	Unquoted grouping column whose levels align with rownames(K).
K	Numeric dense positive-semidefinite covariance or correlation matrix aligned to the grouping levels.
name	Character scalar used as the extractor level, e.g. <code>extract_Sigma(fit, level = "cross")</code> .

Value

A formula marker; never evaluated as a regular R function.

See Also

[kernel_indep\(\)](#), [kernel_dep\(\)](#), [kernel_latent\(\)](#), [scalar\(\)](#), [phylo_scalar\(\)](#), [extract_Sigma\(\)](#).

kernel_unique	<i>Deprecated alias:</i> kernel_unique()
---------------	--

Description

[Deprecated]

Usage

```
kernel_unique(unit, K, name = "kernel")
```

Arguments

unit	Unquoted grouping column whose levels align with rownames(K).
K	Numeric dense positive-semidefinite covariance or correlation matrix aligned to the grouping levels.
name	Character scalar used as the extractor level, e.g. <code>extract_Sigma(fit, level = "cross")</code> .

Details

Only `kernel_unique()` is deprecated; dense-kernel modelling and [kernel_latent\(\)](#), [kernel_indep\(\)](#), and [kernel_dep\(\)](#) remain current. This alias is retained for old formulas. Use [kernel_indep\(\)](#) for a standalone diagonal kernel covariance, or `kernel_latent(..., unique = TRUE)` when a single latent kernel tier should include its diagonal Ψ companion.

Value

A formula marker; never evaluated as a regular R function.

latent	<i>Latent-factor (reduced-rank) random effect:</i> <code>latent(0 + trait g, d = K)</code>
--------	--

Description

Canonical name for a reduced-rank latent-factor random effect at a grouping `g`. Formerly `rr(0 + trait | g, d = K)` – same engine, new name. The "rr" alias still works for backward compat but emits a one-shot deprecation warning per session.

Usage

```
latent(formula, d = 1, unique = TRUE, common = FALSE, lv = NULL)
```

Arguments

formula	$0 + \text{trait} \mid g$ style formula (LHS is the response factor, typically $0 + \text{trait}$; RHS is the grouping factor).
d	Integer; number of latent factors.
unique	Logical; TRUE (default) auto-includes the diagonal trait-specific Ψ companion ($\Sigma = \Lambda\Lambda^\top + \Psi$). Set FALSE for the loadings-only / rotation-invariant subset. The former argument name <code>residual</code> is retained as a soft-deprecated alias.
common	Logical; FALSE (default) estimates one diagonal Ψ variance per trait. TRUE ties the default ordinary diagonal Ψ companion to one shared variance across traits. Only applies when <code>unique = TRUE</code> .
lv	One-sided formula for predictor-informed latent-score means. Runtime support is limited to ordinary unit-tier <code>latent(..., lv = ~x)</code> , and only Gaussian and pure binomial (logit/probit/cloglog) fits are currently admitted (partial coverage, supported for the ordinary-latent case). Source-specific <code>*_latent(..., lv = ~x)</code> forms are parsed and then fail loud (not yet fittable). Where it is admitted, the fit is experimental and exploratory: extract the coefficients with <code>extract_lv_effects()</code> , whose default type = "axis_effect" output is rotation-dependent (an intrinsic latent-axis indeterminacy, not a defect – see <code>extract_lv_effects()</code> for the rotation-invariant type = "trait_effect" alternative), and whose standard errors and intervals are Wald approximations with no established repeated-sampling coverage; do not treat them as calibrated.

Details

Used inside a `gllvmTMB()` formula:

```
value ~ 0 + trait + latent(0 + trait | unit, d = 2)
```

By default, ordinary `latent()` now fits the canonical decomposition $\Sigma = \Lambda\Lambda^\top + \Psi$, with the diagonal Ψ companion included automatically:

```
value ~ 0 + trait + latent(0 + trait | unit, d = 2)
```

Set `unique = FALSE` for the loadings-only subset. For a scalar diagonal Ψ companion shared across traits, use `common = TRUE`; this replaces the older two-term spelling for ordinary intercept-only latent terms.

Value

A formula marker; never evaluated.

See Also

`indep()`, `phylo_latent()`, `diag_re`, `extract_Sigma()`, `extract_lv_effects()` for predictor-informed latent-score coefficients.

Examples

```
# Long-format stacked traits: a 2-factor latent random effect at `site`.
# Ordinary latent() carries its diagonal Psi by default
# (Sigma = Lambda Lambda^T + diag(psi)).
df <- simulate_site_trait(
  n_sites = 30, n_species = 4, n_traits = 4,
  mean_species_per_site = 4, seed = 42
)$data
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 2),
  data = df, unit = "site"
)
extract_Sigma(fit)
```

loading_ci

Confidence intervals on individual entries of the loading matrix

Description

Return per-entry CIs on either the raw reduced-rank loading matrix `Lambda_<level>` or the standardised loading $\rho[t,k] = \text{Lambda}[t,k] / \sqrt{\text{Sigma_total}[t,t]}$ of a confirmatory `gllvmTMB()` fit. A numerical Jacobian of the complete requested loading vector is combined with the TMB `sreport()` covariance. On the standardised scale this includes covariance among axes and uncertainty in fitted denominator components such as Ψ .

Usage

```
loading_ci(
  fit,
  level = c("unit", "unit_obs"),
  method = c("wald", "wald_asym", "profile"),
  conf_level = 0.95,
  sigma_d2 = 1,
  loading_scale = NULL
)
```

Arguments

<code>fit</code>	A multivariate <code>gllvmTMB()</code> fit object.
<code>level</code>	Which loading matrix to summarise: "unit" (default) or "unit_obs". Legacy aliases "B" / "W" are accepted with a one-shot deprecation warning.
<code>method</code>	CI method: "wald" (default) Symmetric Wald via the joint delta method on the requested <code>loading_scale</code> .

	"wald_asym" Asymmetric Wald via the Fisher-z transformation on $\rho_{tk} = \Lambda_{tk} / \sqrt{\Sigma_{total,tt}}$. This method is available only with <code>loading_scale = "standardized"</code>; its bounds stay in $(-1, 1)$. It captures bounded-support asymmetry but not higher-order log-likelihood curvature. "profile" Profile-likelihood inversion through <code>loading_profile()</code>. This re-fits across a grid for each free loading entry and can be used when Wald inference is blocked by a non-positive-definite Hessian. The current profile target is raw Lambda, so <code>loading_scale = "standardized"</code> is refused.
<code>conf_level</code>	Confidence level. Defaults to 0.95.
<code>sigma_d2</code>	Deprecated compatibility argument; ignored. Standardised inference now derives each denominator from model-implied total variance via <code>extract_Sigma()</code> instead of accepting one scalar residual variance.
<code>loading_scale</code>	NULL, "raw", or "standardized". NULL uses "raw" for "wald" / "profile" and "standardized" for "wald_asym". Estimates, SEs, bounds, and downstream null regions are always on this recorded scale.

Details

Per-entry CIs are **only well-defined for confirmatory fits** — i.e. fits supplied with a `lambda_constraint` that fixes enough entries to pin the rotation. Exploratory fits leave Lambda identified only up to a $d \times d$ orthogonal rotation, so the SE on any single `Lambda[i, k]` depends on the rotation convention and is not a biological quantity. This function therefore errors on exploratory fits and points the user at `confirmatory_lambda()` / `suggest_lambda_constraint()`, or at `extract_communality()` / `extract_Sigma()` for rotation-invariant summaries.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
# Build a confirmatory fit
M <- confirmatory_lambda(
  species = species_names,
  group   = species_group,
  d       = 2L,
  loads_on = list(A = 1L, B = 2L)
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 2L),
  data      = df_long,
  family    = binomial(link = "probit"),
  lambda_constraint = list(unit = M)
)
loading_ci(fit, level = "unit")
loading_ci(fit, level = "unit", loading_scale = "standardized")
```

Value

A data frame (one row per Lambda entry) with columns `trait`, `axis`, `estimate`, `se`, `lower`, `upper`, `method`, `loading_scale`, and `pinned`. `pinned = TRUE` records that the raw loading was fixed by `lambda_constraint`. Its raw SE is zero, but its standardised SE can be nonzero because total variance can still vary.

Interval calibration

The point estimates are the supported claim. The interval methods here are provided for exploration: their empirical coverage is not certified for this estimand (the worked example below is a binomial-probit fit), so treat the intervals as exploratory rather than coverage-calibrated. See `NEWS.md` for the current coverage status.

See Also

`flag_unreliable_loadings()` for a decision-aid summary; `confirmatory_lambda()` to build a confirmatory constraint matrix; `extract_communality()` for rotation-invariant alternatives.

loading_profile	<i>Profile-likelihood curve(s) for entries of the loading matrix</i>
-----------------	--

Description

For each requested entry (*i*, *k*) of `Lambda_<level>`, refit the model with that entry pinned to a grid of values and return the resulting log-likelihood curve. The output is a long data.frame with class `profile_loadings`; pass it to `plot()` to visualise the classic LR U-shape, or to `loading_ci()` (via `method = "profile"`) to invert the curve and get CIs that do NOT depend on the fit's Hessian being positive-definite.

Usage

```
loading_profile(
  fit,
  level = c("unit", "unit_obs"),
  entries = NULL,
  n_grid = 11L,
  grid_extent = 6,
  conf_level = 0.95
)
```

Arguments

<code>fit</code>	A confirmatory <code>gllvmTMB()</code> multi-trait fit.
<code>level</code>	Which loading matrix: "unit" (default) or "unit_obs". Legacy "B"/"W" accepted with deprecation.
<code>entries</code>	Optional matrix or data.frame of (<i>i</i> , <i>k</i>) index pairs specifying which Lambda entries to profile. Default NULL profiles every entry that is NOT pinned (i.e. every free entry the data informs). Pinned entries are skipped.
<code>n_grid</code>	Integer; number of grid points per entry. Default 11.
<code>grid_extent</code>	Numeric; total grid width as a multiple of a robust scale estimate of the loading (default 6 — i.e. estimate ± 3 scale units on each side). At <code>pdHess-OK</code> fits the scale is the Wald SE; otherwise a heuristic based on $ \text{Lambda} /2 + 0.5$.
<code>conf_level</code>	Confidence level for the eventual CI inversion; stored on the output for downstream consumers.

Details

This is **expensive**: each entry requires `n_grid` partial refits, so a 20-species 2-factor fit at default `n_grid = 11` is ~440 fits. For binary probit at the scale of this article's fixture, expect minutes. Use entries to restrict to a subset when iterating.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
pf <- loading_profile(fit, level = "unit")
plot(pf)                      # all entries faceted
loading_ci(fit, method = "profile") # inverts pf internally
```

Value

A data.frame of class `profile_loadings` with columns: `trait`, `axis`, `i`, `k`, `profile_value`, `objective` (the negative log-likelihood evaluated at that value), `delta_deviance` ($= 2 * (\text{objective} - \min(\text{objective}))$), `estimate`, `conf_level`. Pinned entries do not appear in the output.

See Also

[loading_ci\(\)](#) for the inverted-CI version, [plot.profile_loadings\(\)](#) for the U-shape visualisation.

make_cross_kernel	<i>Build a cross-lineage relatedness kernel</i>
-------------------	---

Description

`make_cross_kernel()` builds the block relatedness matrix for the C0 coevolution prototype. It builds a correlation-scale positive-semidefinite matrix `K_star = rbind(cbind(A_H, C_HP), cbind(t(C_HP), A_P))`, where `A_H` and `A_P` are within-lineage relatedness matrices and `C_HP` is the cross-lineage bridge induced by the association matrix `W`. The intended route is a prototype: pass `K_star` through the existing `phylo_latent(..., vcv = K_star, unique = TRUE)` engine. The generic `kernel_*()` surface and the `extract_Gamma()` coevolution gate are documented separately; this helper only builds the input kernel.

Usage

```
make_cross_kernel(A_H, A_P, W, rho = 0.5, eps = 1e-08)
```

Arguments

<code>A_H, A_P</code>	Numeric square correlation matrices for the host and partner lineages. Both must be symmetric, positive semidefinite, and have unit diagonal.
<code>W</code>	Numeric association matrix with <code>nrow(W) == nrow(A_H)</code> and <code>ncol(W) == nrow(A_P)</code> . If <code>W</code> has row or column names, they are aligned to <code>A_H</code> / <code>A_P</code> names before the kernel is built.

rho	Scalar bridge strength in $[-1, 1]$. Larger absolute values put more covariance in the off-diagonal host-partner block.
eps	Positive numerical floor used when taking symmetric square roots and scaling an all-zero or near-zero W .

Value

A numeric correlation matrix with the host block first and the partner block second. The matrix carries lightweight metadata for downstream extractors, including the fixed rho used to build the host-partner bridge.

Examples

```
A_H <- matrix(c(
  1.0, 0.3, 0.1,
  0.3, 1.0, 0.2,
  0.1, 0.2, 1.0
), 3, 3, byrow = TRUE)
A_P <- matrix(c(1.0, 0.25, 0.25, 1.0), 2, 2)
W <- matrix(c(1, 0, 0.5, 0, 1, 0.25), 3, 2, byrow = TRUE)
K_star <- make_cross_kernel(A_H, A_P, W, rho = 0.4)
min(eigen(K_star, symmetric = TRUE, only.values = TRUE)$values)
```

make_mesh

Construct an SPDE mesh for gllvmTMB

Description

Construct a triangular mesh and the finite-element quantities used by the `spatial_*()` keywords. The helper uses the public **fmesh** API; the SPDE/GMRF construction follows Lindgren, Rue, and Lindstrom (2011). Scope: mesh/FEM construction and fit ingestion are covered by focused tests; evidence across the broader spatial-family surface remains partial; directional anisotropy, spatiotemporal fields, barriers, and likelihood changes are rejected from this helper contract.

Usage

```
make_mesh(
  data,
  xy_cols,
  type = c("cutoff", "kmeans", "cutoff_search"),
  cutoff,
  n_knots,
  seed = 42,
  mesh = NULL,
  fmesher_func = fmesher::fm_rcdt_2d_inla,
  convex = NULL,
  concave = convex,
```

```
    ...  
  )
```

Arguments

data	A data frame containing the coordinate columns.
xy_cols	Character names of exactly two finite numeric coordinate columns in data.
type	Mesh construction method: "cutoff", "kmeans", or "cutoff_search".
cutoff	Minimum vertex separation for type = "cutoff".
n_knots	Number of coordinate centres for "kmeans", or target number of mesh vertices for "cutoff_search". The latter returns the closest valid mesh when the exact target is unavailable.
seed	Seed used only while selecting k-means centres. The caller's RNG state is restored before this function returns.
mesh	Optional pre-built fmesher mesh.
fmesher_func	Mesh constructor from fmesh .
convex, concave	Optional non-convex-hull controls passed to <code>fmesher::fm_nonconvex_hull()</code> .
...	Additional arguments passed to fmesher_func.

Value

A `gllvmTBMmesh` object containing `loc_xy`, `xy_cols`, `mesh`, the finite-element matrices in `spde`, mesh centres in `loc_centers`, and the observation-to-mesh projection matrix `A_st`.

References

Lindgren F, Rue H, Lindstrom J (2011). An explicit link between Gaussian fields and Gaussian Markov random fields: the SPDE approach. *Journal of the Royal Statistical Society: Series B*, 73, 423-498.

Examples

```
set.seed(1)  
df <- data.frame(X = runif(40), Y = runif(40))  
mesh <- make_mesh(df, c("X", "Y"), cutoff = 0.1)  
plot(mesh)
```

meta_V	<i>Known-V meta-analytic random effect (canonical name):</i> meta_V(V = V)
--------	--

Description

Canonical formula keyword for the known-sampling-error term in two-stage meta-regression workflows. The name makes it explicit that what is "known" is the **V** matrix (the sampling-error covariance). [meta_known_V\(\)](#) is retained as a soft-deprecated alias; both names desugar identically in the formula parser.

Usage

```
meta_V(V, type = "exact")
```

Arguments

V	Known sampling variance or covariance marker. In current exact-additive fits, pass the actual matrix via the top-level known_V = argument to gllvmTMB() . Formula-level V = V names the marker and keeps the syntax aligned with <code>drmTMB::meta_V()</code> .
type	Sampling-covariance mode. "exact" is implemented. "proportional" is not implemented and fails clearly in the formula parser rather than being treated as exact.

Details

Pass the matrix V via the top-level known_V = V argument to [gllvmTMB\(\)](#) when using the exact-additive (default) form. For block-diagonal within-study correlation, build V via [block_V\(\)](#). type = "proportional" is not implemented and is deliberately rejected rather than silently treated as exact.

Value

A formula marker; never evaluated.

See Also

[meta_known_V\(\)](#) (deprecated alias); [block_V\(\)](#); [gllvmTMB\(\)](#).

Examples

```
# Two-stage meta-regression with a known per-row sampling-variance V.
set.seed(131)
df <- expand.grid(
  site = factor(seq_len(50)),
  trait = factor(paste0("t", 1:3))
)
df$value <- rnorm(nrow(df), sd = 0.5)
```

```
df$sampling_var <- runif(nrow(df), min = 0.02, max = 0.08)
V <- diag(df$sampling_var)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 1) +
    meta_V(V = V, type = "exact"),
  data = df, trait = "trait", unit = "site", known_V = V
)
```

miss_control

Missing-data control for `gllvmTMB()`

Description

Configures how `gllvmTMB()` treats missing **responses** and missing **predictors**. Pass the result to the `missing =` argument of `gllvmTMB()`.

Usage

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

Arguments

- | | |
|-----------|--|
| response | How to treat rows whose response value is NA. "drop" (default) omits each missing response <i>cell</i> before the likelihood is built. In long format one row is one (unit, trait) cell, and the wide <code>traits()</code> route drops the same cells, so a unit keeps every trait it does have – this is cell-wise omission, not case-wise deletion. "include" keeps those rows, builds an observed-response mask (<code>is_y_observed</code>), and contributes nothing to the likelihood for the masked rows – the frequentist observed-data likelihood. For the supported routes, the observed-response likelihood contribution matches the corresponding observed rows, while cell identity is preserved for <code>predict_missing()</code> . |
| predictor | How to treat missing predictors (covariates). "fail" (the default): a missing value in the fixed-effect design matrix is an error, exactly as today. "model" treats a missing predictor declared with <code>mi(x)</code> as a latent variable integrated out by the Laplace approximation, with its covariate model supplied via the <code>impute =</code> argument of <code>gllvmTMB()</code> . The current routes fit one modelled predictor at a time: Gaussian fixed-effect, grouped-intercept, or phylogenetic-intercept covariate models, plus fixed-effect binary, ordered, and unordered discrete predictors. |
| engine | The estimation engine. The supported value is "laplace" (TMB Laplace approximation). "em" (the Gaussian-only EM special case) and "profile" are reserved names, not yet supported . |

Details

There is deliberately **no** estimator argument in `miss_control()`: estimator choice belongs to `gllvmTMB()`. The default `gllvmTMB()` fit uses ML; `gllvmTMB(REML = TRUE)` is a Gaussian-only pilot. It combines with either missing-response policy – including `miss_control(response = "include")`, where the restricted likelihood is formed over the observed rows – but not yet with `mi()` predictor models. There is **no** MI ("multiple imputation") engine here; multiple imputation is a separate workflow.

Under the default `miss_control(response = "drop", predictor = "fail", engine = "laplace")` the missing-data layer is an exact no-op: a complete- data fit is byte-identical to a fit built before this layer existed.

Value

A named list with elements `response`, `predictor`, and `engine`.

See Also

`gllvmTMB()` for the `missing =` argument; `gllvmTMBcontrol()` for optimiser / initialisation control.

Examples

```
miss_control()           # defaults: drop / fail / laplace
miss_control(response = "include") # keep missing-response rows (masked)
```

multinomial

Multinomial (baseline-category logit) family

Description

Baseline-category logit / softmax family for a single **unordered** categorical response with $K \geq 3$ categories. For a categorical trait with categories $1, \dots, K$ and reference category 1, the model is $\eta_k = \beta_{0k} + x^\top \beta_k$ for $k = 2, \dots, K$ with $\eta_1 \equiv 0$, and $\Pr(y = k) = \exp(\eta_k) / \sum_j \exp(\eta_j)$ (softmax). It recovers the $(K - 1)$ per-category intercepts and slopes as contrasts against the reference category.

Usage

```
multinomial(link = "logit", baseline = NULL)
```

Arguments

link	Character; only the baseline-category "logit" link is supported (present for API symmetry).
baseline	Optional reference category (level of the response factor) pinned at $\eta = 0$. NULL (default) uses the first factor level.

Details

This is a **response** family and must not be confused with the `categorical()` constructor, which is a missing-*predictor* imputation family. Because an unordered categorical response spans $K - 1$ latent liability dimensions rather than one, the admitted structured-term surface is bounded and every cell below was decided by a signed recovery campaign, not construction alone.

Admitted: fixed effects always; an ordinary shared `latent(0 + trait | unit, d = k)` ordination, which connects a multinomial trait to other-family traits through its $K - 1$ contrast pseudo-traits (the default `unique = TRUE` works, with the categorical contrast Ψ mapped off); the intercept-only phylogenetic-machinery tier in all three modes – `phylo_latent(species, d = k)` (reduced-rank), `phylo_dep()` (full unstructured V ; the identical `phylo_rr` parameterisation as `phylo_latent(d = K - 1)`), and `phylo_indep()` (diagonal V ; standalone `phylo_unique()` is its deprecated alias) – together with their `animal_*()` (pedigree/ A) and single-name `kernel_*()` (dense K) twins, which run the same engine; and the analogous spatial (SPDE) mode axis – `spatial_latent(0 + trait | coords, d = k)` (loadings-only), `spatial_indep()` (per-contrast independent fields), and `spatial_dep()` (verified identical to `spatial_latent(d = n_traits)`). All of the phylogenetic/relatedness cells are loadings-only: none emits a Ψ companion, and `unique = TRUE` (a free phylogenetic Ψ) is NOT admitted.

The honest evidence, not softened: on the phylogenetic/ relatedness surface, the one-categorical-draw-per-species recovery gate FAILED (rail rate 8/20 against a 6/20 threshold, identically for `phylo_latent()/animal_latent()/kernel_latent()` by proven engine identity; `phylo_dep()` rails 8/20 separately). A pre-registered replication rescue – five categorical draws per species rather than one – PASSED for the loadings-only and full- V cells (rail rate 4/20, median $\hat{\rho}$ 0.680 against a true 0.6) but has NOT been tested for `phylo_indep()`'s diagonal- V mode, whose own corrected rerun independently FAILED (small contrast variances collapse; a planted-zero check fails). The spatial mode axis and the `(1 | group)` route below both PASSED their signed recovery gates outright, with no replication rescue needed. **One categorical draw per species does not identify V ; five draws per species does.**

A generic `(1 | group)` random intercept is also admitted, but its semantics are **baseline-vs-rest**, not per-category: the engine adds one draw per group level to EVERY one of a multinomial observation's $K - 1$ baseline-contrast rows, which shifts $P(y = \text{baseline})$ versus $P(y \neq \text{baseline})$ without changing the odds between any two NON-baseline categories (within one fit, across groups). `sigma_re`'s substantive interpretation is therefore **reference-category- specific**, and re-labelling the baseline is **not** a reparameterisation of the same model: under `baseline = 1` the shared draw constrains the log-odds between the two non-baseline categories to be constant across groups, while under a different baseline the SAME engine shares a (new) draw between a different pair of categories instead – a different parametric restriction, not the same distribution relabelled. Re-fitting under a different baseline therefore changes the fitted response-scale probabilities too, not only `sigma_re`. The non-phylogenetic cluster / cluster2 diagonal tier (`indep(0 + trait | g)` via the `cluster/cluster2` arguments to `gl1vmTMB`, and its soft-deprecated standalone `unique()` alias) is admitted as per-contrast independent variances – the per-category random intercept most users want – but `common = TRUE` (the `scalar()` modifier) at that tier is NOT admitted. Both the `(1 | group)` and `cluster/cluster2` routes abort typed if every level of the grouping factor covers exactly one categorical observation: that is an observation-level random effect (OLRE) in disguise, unidentifiable because the softmax latent scale is fixed.

Refused, not merely deferred: `phylo_scalar()/animal_scalar()/kernel_scalar()/spatial_scalar()` and `common = TRUE` at the `cluster/cluster2` tier – a single shared level across the $K - 1$ contrasts has no interpretable null on the $(I + J)$ contrast geometry.

Everything else is still deferred and fails loud (R/multinomial-fence.R) rather than reaching an untested categorical path: `dep()`, explicit `unique()` / `indep()` at the unit tier, `latent()/dep()` at the cluster/cluster2 tiers, and the `unit_obs` grouping tier; augmented (intercept + slope) forms of every admitted keyword above; `unique = TRUE` on `phylo_latent()/animal_latent()/kernel_latent()` (a free phylogenetic Ψ) and `spatial_latent()`'s paired diagonal companion; standalone `spatial_unique()/deprecated bare spatial()` (a paired-companion alias mechanism, unlike the phylogenetic mode axis's `*_unique()`, which IS admitted); multi-kernel (more than one `kernel_latent()/kernel_dep()/kernel_indep()` name in one fit); `meta_V()` / `equalto()` (confirmed Gaussian-only, no route on a categorical-contrast pseudo-trait); and `mi()` predictor terms. A two-category response is exactly `binomial(link = "logit")` and is redirected there. See also [ordinal_probit\(\)](#) for *ordered* categories.

Value

A family object with class `c("multinomial", "family")`.

See Also

[ordinal_probit\(\)](#) for ordered categories; [categorical\(\)](#) for the missing-predictor imputation namesake.

Examples

```
multinomial()
```

ordinal_probit

Ordinal-probit threshold family for the multivariate engine

Description

The Wright/Falconer/Dempster-Lerner threshold model for K-category ordinal data with $K \geq 3$, fitted on the latent (probit) scale. The latent variable representation is $y^* = \eta + \varepsilon$, with $\varepsilon \sim N(0, 1)$ and the observed category $y = k$ iff $\tau_{k-1} < y^* \leq \tau_k$, using cutpoints $\tau_0 = -\infty$, $\tau_1 = 0$ (fixed for identifiability), $\tau_2, \dots, \tau_{K-1}$, $\tau_K = +\infty$. A K-category trait therefore estimates $K - 2$ free cutpoints.

Usage

```
ordinal_probit(link = "probit")
```

Arguments

link	Always "probit"; provided for API symmetry with the other family constructors.
------	--

Details

Because ε has unit variance by construction, the link-residual variance $\sigma_d^2 = 1$ *exactly* (no trigamma correction needed). This fixes the ordinal liability scale; it does not make an ordinal response directly comparable with an observed continuous response without an explicit scale definition. In the special two-component model containing only phylogenetic variance plus the unit probit residual, the liability-scale ratio is $H^2 = \sigma_{\text{phy}}^2 / (\sigma_{\text{phy}}^2 + 1)$. Any additional fitted variance component belongs in the denominator, so do not use that shortcut for richer models.

Hadfield (2015) eqn 10 shows that the $K = 2$ case reduces exactly to `binomial(link = "probit")`, so use `binomial()` for binary outcomes and `ordinal_probit()` for $K \geq 3$.

Convention: the engine follows Hadfield's notation ($\tau_1 = 0$ fixed, $K - 2$ free cutpoints reported as `cutpoint_2, ..., cutpoint_{K-1}`). This differs from `brms::cumulative()`, which reports $K - 1$ cutpoints as `Intercept[1..K-1]`.

Cross-engine note: `gllvmTMB`'s `ordinal_probit()` is a probit threshold model (unit latent variance; Hadfield 2015), whereas `GLLVM.jl`'s ordinal family is a cumulative-*logit* model. Cutpoints and loadings therefore live on different link scales (differing by a factor of roughly $\pi/\sqrt{3}$) and must not be compared across engines at machine tolerance; the engine = "julia" bridge maps probit to probit and does not route a logit ordinal response.

Value

A family object with class `c("ordinal_probit", "family")`. The cutpoints are estimated as part of the model fit; recover them with `extract_cutpoints()`.

References

- Dempster, E. R. and Lerner, I. M. (1950). Heritability of threshold characters. *Genetics* 35:212-236.
- Falconer, D. S. and Mackay, T. F. C. (1996). *Introduction to Quantitative Genetics*, 4th ed. Longman.
- Felsenstein, J. (2005). Using the quantitative genetic threshold model for inferences between and within species. *Phil. Trans. R. Soc. B* 360:1427-1434.
- Felsenstein, J. (2012). A comparative method for both discrete and continuous characters using the threshold model. *Am. Nat.* 179:145-156.
- Hadfield, J. D. (2015). Increasing the efficiency of MCMC for hierarchical phylogenetic models of categorical traits using reduced mixed models. *Methods Ecol. Evol.* 6:706-714. doi:10.1111/2041210X.12354
- Mizuno, A. *et al.* (2025). Phylogenetic comparative methods for threshold traits. *J. Evol. Biol.* 38(12):1699-1712.

See Also

`extract_cutpoints()` to recover $\tau_2, \dots, \tau_{K-1}$ after fitting.

Examples

```
ordinal_probit()
```

pedigree_to_A

*Pedigree to additive-genetic relatedness matrix A***Description**

Computes the dense numerator-relationship matrix **A** from a 3-column pedigree (id, sire, dam) using Henderson's (1976) recursive formula:

Usage

```
pedigree_to_A(pedigree)
```

Arguments

pedigree A data frame with one row per individual and columns identifying the individual, sire, and dam. **Column names are resolved BY NAME** (MCMCglmm-style) using these synonyms:

- Individual ID column: id or animal.
- Sire (father) column: sire or father.
- Dam (mother) column: dam or mother.

If none of the named synonyms is present, the function falls back to **positional** access – column 1 = id, column 2 = sire, column 3 = dam – with a soft note. Unknown parents encoded as NA or 0 (both treated as missing).

Details

- $A_{ii} = 1 + F_i$, where F_i is the inbreeding coefficient of individual i .
- $A_{ij} = \frac{1}{2}(A_{i,\text{sire}(j)} + A_{i,\text{dam}(j)})$ for j younger than i (pedigree is sorted oldest first).

Founder individuals (both parents unknown) are assumed unrelated and non-inbred: $A_{\text{founder},\text{founder}} = 1$.

Users typically don't call `pedigree_to_A()` directly – pass `pedigree = ped` to an `animal_scalar()` / `animal_latent()` / `animal_indep()` / `animal_dep()` keyword in the formula, and the keyword's parser handles the conversion internally. This function is exported as a public helper for users who want the matrix for their own diagnostics (e.g. inspection of inbreeding coefficients or pre-computation for repeated fits).

For very large pedigrees ($n > 5000$) the cubic-time cost of constructing a dense **A** can become noticeable. When a sparse precision is available, pass it directly with `Ainv =`; for comparison against established implementations, see `nadiv::makeAinv()`.

Value

Dense numeric matrix $n \times n$ with rownames / colnames equal to the individual IDs.

See Also

`animal_scalar()` and `siblings`.

pedigree_to_Ainv_sparse

Sparse pedigree inverse-A via Henderson-Quaas (MCMCglmm-free)

Description

Builds the sparse precision matrix $\mathbf{A}^{-1}$ for an animal- model pedigree using only **Matrix** – **no** MCMCglmm **dependency**. The returned $\mathbf{A}^{-1}$ is genuinely sparse (typical density $O(1/n)$), so the TMB fit is $O(n)$ rather than $O(n^3)$ – this is the canonical fast path for wild pedigrees with $n_{\text{individuals}} > 500$. The construction is the standard Henderson (1976) / Quaas (1976) sparse inverse: inbreeding coefficients F are derived from the dense tabular relatedness matrix, then $\mathbf{A}^{-1}$ is assembled directly from the per-individual Mendelian sampling variances. It reproduces `MCMCglmm::inverseA(pedigree)$Ainv` exactly. (The current inbreeding step forms the dense $n \times n$ $\mathbf{A}$, an $O(n^2)$ one-time cost.)

Usage

```
pedigree_to_Ainv_sparse(pedigree)
```

Arguments

pedigree A 3-column data frame: id (or animal), sire (or father), dam (or mother).

Details

Usage:

```
Ainv <- pedigree_to_Ainv_sparse(ped)
fit <- glmmTMB(value ~ 0 + trait +
               animal_scalar(id, Ainv = Ainv),
               data = df, unit = "id", cluster = "id")
```

The pedigree follows the conventional (**id**, **sire**, **dam**) column order, OR named columns matching id/animal, sire/father, dam/mother for by-name lookup. Unknown parents are encoded as NA or 0. Founders (both parents unknown) are treated as unrelated.

Value

A sparse dgCMatrix of dimension $n_{\text{individuals}} \times n_{\text{individuals}}$ with rownames and colnames equal to the individual IDs. Sparse storage; typical density is $O(1/n)$.

References

Henderson, C. R. (1976). A simple method for computing the inverse of a numerator relationship matrix used in prediction of breeding values. *Biometrics* 32: 69-83.

Quaas, R. L. (1976). Computing the diagonal elements and inverse of a large numerator relationship matrix. *Biometrics* 32: 949-953.

Hadfield, J. D. (2010). MCMC methods for multi-response generalised linear mixed models: the MCMCglmm R package. *Journal of Statistical Software* 33: 1-22. (The MCMCglmm::inverseA() reference implementation this builder is validated against.)

See Also

[pedigree_to_A\(\)](#) for the dense companion; [animal_scalar\(\)](#) and siblings for the keyword family.

Examples

```
ped <- data.frame(
  id   = paste0("i", 1:6),
  sire = c(NA, NA, "i1", "i1", "i3", "i3"),
  dam  = c(NA, NA, "i2", "i2", "i4", "i4")
)
Ainv <- pedigree_to_Ainv_sparse(ped)
class(Ainv) # "dgCMatrix"
dim(Ainv)   # 6 x 6
```

phylo

Phylogenetic random effect: lme4-bar mode-dispatch wrapper

Description

phylo() is a unified entry point for the package's three phylogenetic modes (phylo_indep, phylo_latent, phylo_dep). It accepts an lme4-bar formula on the first argument and dispatches to the appropriate canonical keyword based on the LHS shape and the optional mode = ... argument. The three current keywords stay first-class – phylo() is an additive alias matching the lme4 / brms / drmtmb convention.

Usage

```
phylo(
  formula,
  tree = NULL,
  vcv = NULL,
  mode = NULL,
  d = 1,
  unique = FALSE,
  A = NULL,
  Ainv = NULL
)
```

Arguments

formula	An lme4-bar formula (<lhs> <species_column>) OR, for backward-compat, a bare unquoted column name (legacy form). The bar's RHS is the species factor; the LHS determines the covariance structure (combined with mode, see Dispatch rules).
tree	An ape::phylo object. Canonical. Sparse $\mathbf{A}^{-1}$ via Hadfield & Nakagawa (2010).
vcv	A tip-only phylogenetic correlation matrix (n_species x n_species). Legacy / superseded.
mode	One of "scalar" / "indep" / "latent" / "dep". Optional when LHS is 1 (defaults silently to "scalar"). Mandatory when LHS is 0 + trait.
d	Latent rank for mode = "latent". Default 1.
unique	Logical; for mode = "latent", TRUE includes the phylo-structured diagonal Ψ_{phy} companion. The default FALSE preserves the loadings-only source-specific latent path.
A	Tip-level relatedness matrix; alias of vcv =.
Ainv	Sparse precision matrix (inverse of A).

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
library(ape)
tree <- rcoal(20); tree$tip.label <- paste0("sp", 1:20)
# Single shared phylogenetic variance (= phylo_scalar)
fit_s <- gllvmTMB(value ~ 0 + trait + phylo(1 | species, tree = tree),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species")
# Per-trait independent phylogenetic variances on shared A
fit_u <- gllvmTMB(value ~ 0 + trait +
  phylo(0 + trait | species, mode = "indep", tree = tree),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species")
# Reduced-rank cross-trait phylogenetic decomposition (= phylo_latent)
fit_l <- gllvmTMB(value ~ 0 + trait +
  phylo(0 + trait | species, mode = "latent",
    d = 2, tree = tree),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species")
```

```
# Backward-compat: legacy bare-name form
Cphy <- vcv(tree, corr = TRUE)
fit_legacy <- glvmTMB(value ~ 0 + trait + phylo(species),
                      data      = df,
                      trait     = "trait",
                      unit      = "species",
                      cluster    = "species",
                      phylo_vcv = Cphy)
```

Value

A formula marker; never evaluated.

Dispatch rules

Form	Mode default	Rewrites to
phylo(1 \ species)	"scalar"	phylo_indep(0 + trait \ species, c
phylo(0 + trait \ species, mode = "indep")	(mandatory)	phylo_indep(0 + trait \ species)
phylo(0 + trait \ species, mode = "latent", d = K)	(mandatory)	phylo_latent(species, d = K)
phylo(0 + trait \ species, mode = "dep")	(mandatory)	phylo_dep(0 + trait \ species)

When the LHS expands to a single column (1, intercept-only), mode is degenerate: it defaults silently to "scalar", and explicit mode = "scalar" is accepted (no warning). When the LHS is $0 + \text{trait}$, mode is **mandatory** – choosing between "indep" / "latent" / "dep" is a meaningful decision (per-trait diagonal vs reduced-rank decomposition vs full unstructured) and the parser refuses to silently default.

Backward compatibility

Legacy bare-name calls `phylo(species)` and `phylo(species, vcv = Cphy)` continue to work as deprecated aliases of `phylo_scalar(species)` / `phylo_scalar(species, vcv = Cphy)`. The legacy form rewrites to `propto(0 + species | trait)` internally, picking up the phylogenetic covariance from the top-level `phylo_vcv =` argument to `glvmTMB()`.

Intercept-and-slope terms

The unified `phylo()` wrapper is limited to intercept-only trait covariance. For an intercept and one or more slopes, use an explicit structured mode such as `phylo_indep()`, `phylo_latent()`, or `phylo_dep()`; unsupported LHS forms fail with a message showing the accepted syntax.

Cross-package coexistence

`drmTMB` also exposes `phylo(1 | species, tree = tree)` with the same Hadfield-Nakagawa sparse $\mathbf{A}^{-1}$ internal path. Both packages share the same calling convention; user muscle memory transfers.

See Also

[phylo_scalar\(\)](#), [phylo_indep\(\)](#), [phylo_latent\(\)](#), [phylo_dep\(\)](#).

phylo_dep	<i>Full unstructured phylogenetic trait covariance:</i> <code>phylo_dep(0 + trait species)</code>
-----------	---

Description

Canonical name for the **full unstructured cross-trait phylogenetic covariance** $\Sigma_{\text{phy}} \otimes \mathbf{A}$, with $T(T + 1)/2$ free parameters (Σ_{phy} parameterised via Cholesky for PSD-ness). Mathematically identical to `phylo_latent(species, d = T)` standalone; the keyword choice is documentary.

Usage

```
phylo_dep(formula, tree = NULL, vcv = NULL, A = NULL, Ainv = NULL)
```

Arguments

formula	<code>0 + trait species</code> style formula, an admitted intercept-and-slope form, or the Gaussian slope-only response-column form <code>0 + x1 + ... trait</code> .
tree	An <code>ape::phylo</code> object. Canonical.
vcv	A tip-only phylogenetic correlation matrix (<code>n_species</code> x <code>n_species</code>). Legacy alias of <code>A =</code> .
A	Tip-level relatedness matrix (<code>n_species</code> x <code>n_species</code>) – alias of <code>vcv =</code> , aligned with the <code>animal_*</code> family’s argument naming.
Ainv	Sparse precision matrix (inverse of <code>A</code>).

Details

$$\text{vec}(\mathbf{P}) \sim \mathcal{N}(\mathbf{0}, \Sigma_{\text{phy}} \otimes \mathbf{A}_{\text{phy}}).$$

Use `phylo_dep()` when you want an explicit full-unstructured cross-trait phylogenetic covariance fit. Use `phylo_latent(..., unique = TRUE)` for the rank-reduced paired phylogenetic decomposition. Use `phylo_indep()` for the marginal-only per-trait variance fit.

Slope-only response-column form:

In a Gaussian long-format model, `phylo_dep(0 + lat + temp | trait)` gives response-column slope deviations with a full positive-definite covariance among predictors. Fixed column intercepts remain `0 + trait` in the main formula; this term never adds a random intercept. Retrieve the predictor covariance with `extract_Sigma(fit, level = "column_slope")`. Use `phylo_indep()` for a diagonal predictor covariance. The helper `phylo_slope(lat + temp | trait)` is an alias for this full route.

Mutual exclusion with phylo_latent() / phylo_indep():

Combining phylo_dep(0 + trait | species) with phylo_latent is **over-parameterised**; combining with phylo_indep is **redundant** (phylo_dep already includes the diagonal). The parser raises cli::cli_abort() in any of these cases.

Pass the phylogeny via tree = phylo (canonical, sparse $\mathbf{A}^{-1}$) or vcv = Cphy ([Superseded], dense). See [phylo_latent\(\)](#) for the full discussion of the two paths.

Value

A formula marker; never evaluated.

See Also

[phylo_latent\(\)](#), [phylo_indep\(\)](#), [dep\(\)](#), [spatial_dep\(\)](#), [extract_Sigma\(\)](#).

Examples

```
tree <- ape::rcoal(8)
# Match the simulator species levels to the covariance matrix row order.
tree$tip.label <- as.character(seq_len(8))
sim <- simulate_site_trait(
  n_sites = 1, n_species = 8, n_traits = 3,
  mean_species_per_site = 8,
  Cphy = ape::vcv(tree, corr = TRUE),
  sigma2_phy = rep(0.3, 3), seed = 1
)
sim$data$species <- factor(sim$data$species, levels = tree$tip.label)
fit <- glvmTMB(
  value ~ 0 + trait + phylo_dep(0 + trait | species),
  data      = sim$data,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_tree = tree
)
```

phylo_indep	<i>Per-trait phylogenetic marginal variance:</i> phylo_indep(0 + trait species)
-------------	---

Description

Canonical name for **T per-trait phylogenetic variances coupled by the phylo correlation matrix A**; standalone = T univariate phylogenetic mixed models stacked.

Usage

```

phylo_indep(
  formula,
  tree = NULL,
  vcv = NULL,
  A = NULL,
  Ainv = NULL,
  common = FALSE
)

```

Arguments

formula	Intercept-only $\emptyset + \text{trait} \mid \text{species}$, the supported intercept-and-slope form described above, or the Gaussian long-format slope-only response-column form $\emptyset + x_1 + \dots \mid \text{trait}$.
tree	An <code>ape::phylo</code> object. Canonical.
vcv	A tip-only phylogenetic correlation matrix (<code>n_species</code> x <code>n_species</code>) for the ordinary forms. For the slope-only response-column form it is instead <code>n_traits</code> x <code>n_traits</code> , with row names matching the resolved RHS trait levels. Legacy alias of <code>A</code> =.
A	Tip-level relatedness matrix (<code>n_species</code> x <code>n_species</code>) for the ordinary forms; for the slope-only response-column form, the corresponding trait-level matrix. Alias of <code>vcv</code> =, aligned with the <code>animal_*</code> family's argument naming.
Ainv	Sparse precision matrix (inverse of <code>A</code>).
common	FALSE (default) for a separate phylogenetic variance per trait; TRUE ties all traits to one shared phylogenetic variance (intercept-only). <code>phylo_indep($\emptyset + \text{trait} \mid \text{species}$, <code>common</code> = TRUE)</code> is the canonical one-shared-variance spelling and fits the same model as the soft-deprecated <code>phylo_scalar(species)</code> .

Details

Each trait t gets its own variance $\sigma_{\text{phy},t}^2$ on the same phylogenetic correlation matrix $\mathbf{A}_{\text{phy}}$; trait-specific random vectors are otherwise independent.

$$\mathbf{p}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{phy},t}^2 \mathbf{A}_{\text{phy}}), \quad t = 1, \dots, T.$$

Use `phylo_indep()` for an explicit marginal-only phylogenetic fit (no cross-trait phylogenetic decomposition). Use `phylo_latent(..., unique = TRUE)` for the paired phylogenetic decomposition $\Sigma_{\text{phy}} = \mathbf{A}_{\text{phy}} \mathbf{A}_{\text{phy}}^{\top} + \Psi_{\text{phy}}$.

Mutual exclusion with `phylo_latent()`:

Combining `phylo_indep( $\emptyset + \text{trait} \mid \text{species}$ )` with `phylo_latent(species, d = K)` is **over-parameterised** and the parser raises a `cli::cli_abort()`.

Intercept-and-slope form:

A single-covariate random regression uses `phylo_indep(1 + x  $\mid$  species)` in wide syntax or `phylo_indep( $\emptyset + \text{trait} + (\emptyset + \text{trait}):x \mid \text{species}$ )` in explicit long syntax. It estimates independent trait-specific intercept-slope blocks. The intercept-slope correlation is estimated within each trait; cross-trait blocks are structurally zero.

Slope-only response-column form:

For a long-format Gaussian model, `phylo_indep(0 + lat + temp | trait)` adds response-column deviations to the fixed slopes without adding a random intercept. The phylogeny is indexed by the response-column factor on the right of `|`; `indep` makes the covariance among `lat` and `temp` slopes diagonal. With slope coefficient matrix $\mathbf{B}$ (rows are response columns and columns are predictors), let $\mathbf{b} = \text{vec}(\mathbf{B}^T)$ order coefficients by response column: `lat`, `temp` for column 1, then `lat`, `temp` for column 2, and so on. Then $\text{Cov}(\mathbf{b}) = \mathbf{A}_{\text{phy}} \otimes \text{diag}(\sigma_{\text{lat}}^2, \sigma_{\text{temp}}^2)$. (The native implementation stores the permutation-equivalent column-major vector.) Retrieve that predictor-basis covariance with `extract_Sigma(fit, level = "column_slope")`. It is not a trait-level Σ . This V1 route requires bare finite numeric predictors, exactly the resolved trait column on the RHS, and Gaussian responses; wide-format and non-Gaussian column slopes are planned work.

Pass the phylogeny via `tree = phylo` (canonical, sparse $\mathbf{A}^{-1}$) or `vcv = Cphy` ([**Superseded**], dense). See [phylo_latent\(\)](#) for the full discussion of the two paths.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
library(ape)
tree <- rcoal(20); tree$tip.label <- paste0("sp", 1:20)
fit <- glvmTMB(value ~ 0 + trait +
  phylo_indep(0 + trait | species, tree = tree),
  data      = df,
  trait     = "trait",
  unit      = "species",
  cluster   = "species")
```

Value

A formula marker; never evaluated.

References

- **Williams et al.** (2025) Phylogenetic generalised linear mixed models for multi-trait comparative analyses. *bioRxiv* 2025.12.20.695312. The marginal `phylo_indep + indep` standalone form stacked across traits matches their PGLMM Eq. 3.

See Also

[phylo_latent\(\)](#), [phylo_dep\(\)](#), [indep\(\)](#), [spatial_indep\(\)](#), [extract_Sigma\(\)](#).

phylo_latent	<i>Reduced-rank phylogenetic latent factors:</i> <code>phylo_latent(species, d = K)</code>
--------------	--

Description

Canonical name for the reduced-rank phylogenetic random effect. Formerly `phylo_rr(species, d = K)` – same engine, new name.

Usage

```

phylo_latent(
  species,
  d = 1,
  tree = NULL,
  vcv = NULL,
  A = NULL,
  Ainv = NULL,
  unique = FALSE
)

```

Arguments

species	Unquoted column name for the species factor.
d	Integer; number of phylogenetic latent factors.
tree	An <code>ape::phylo</code> object. Canonical. Use this if you have a tree.
vcv	A tip-only phylogenetic correlation matrix (<code>n_species</code> x <code>n_species</code>). Legacy alias of <code>A</code> .
A	Tip-level relatedness matrix (<code>n_species</code> x <code>n_species</code>) – alias of <code>vcv</code> , aligned with the <code>animal_*</code> family’s argument naming. Supply one of <code>tree</code> , <code>vcv</code> , or <code>A</code> / <code>Ainv</code> .
Ainv	Precision matrix (inverse of <code>A</code>). Sparse inputs are preserved for the sparse precision route.
unique	Logical; TRUE auto-includes the phylo-structured diagonal trait-specific Ψ_{phy} companion, folding the shared rank-K loadings and the diagonal companion into a single term ($\Sigma_{phy} = \Lambda\Lambda^T \otimes \mathbf{A} + \Psi_{phy} \otimes \mathbf{A}$). The default FALSE preserves the loadings-only / rotation-invariant subset.

Details**Two phylogeny inputs: tree = (canonical) and vcv = (legacy):**

Pass the phylogeny inside the keyword via one of two arguments:

- `tree = phylo` (recommended when the tree is available) – the full `ape::phylo` object. The package constructs the sparse phylogenetic precision using its Hadfield–Nakagawa implementation.
- `vcv = Cphy` – a tip-level `n_species` x `n_species` correlation matrix for analyses that begin from a supplied covariance matrix.

These inputs encode the same tip-level covariance target when they are constructed from the same tree and aligned identically. Numerical agreement still depends on labels, scaling, and fitting health; the function does not estimate or report ancestral states.

The two routes differ in one respect worth knowing. The `vcv` route adds a fixed $1e-8$ ridge to the supplied matrix before inverting it; the `tree` route builds the sparse precision analytically from branch lengths and adds no ridge. The two therefore agree with each other only to roughly $1e-5$ in log-density, and less on large or poorly conditioned trees.

Species are matched by FACTOR LEVEL, not by tree tip order:

The species column is aligned to the phylogeny through the order of its **factor levels**: `levels(data$species)` is looked up against the tip labels, and whatever order those levels happen to be in is the order the latent scores are indexed by. Tree tip order is not consulted.

If the factor's levels do not correspond to the species you intend, the model still fits and issues no warning. It simply fits a different phylogeny. Set the levels explicitly:

```
data$species <- factor(data$species, levels = tree$tip.label)
```

Every example below does this. It is not decoration.

See the [phylogenetic covariance article](#) for the benchmark.

Value

A formula marker; never evaluated.

References

Hadfield JD, Nakagawa S (2010). General quantitative genetic methods for comparative biology: phylogenies, taxonomies and multi-trait models for continuous and categorical characters. *J. Evol. Biol.* 23: 494-508. doi:10.1111/j.14209101.2009.01915.x

See Also

[phylo_scalar\(\)](#), [phylo_indep\(\)](#), [phylo_dep\(\)](#), [phylo_rr\(\)](#) (deprecated alias).

Examples

```
tree <- ape::rcoal(20)
# Match the simulator species levels to the covariance matrix row order.
tree$tip.label <- as.character(seq_len(20))
sim <- simulate_site_trait(
  n_sites = 1, n_species = 20, n_traits = 4,
  mean_species_per_site = 20,
  Cphy = ape::vcv(tree, corr = TRUE),
  sigma2_phy = rep(0.3, 4), seed = 1
)
sim$data$species <- factor(sim$data$species, levels = tree$tip.label)
fit <- gllvmTMB(
  value ~ 0 + trait + phylo_latent(species, d = 2),
  data      = sim$data,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_tree = tree
)
```

phylo_scalar	Single-shared-variance phylo_scalar(species)	phylogenetic	random	effect:
--------------	---	--------------	--------	---------

Description

Canonical name for the single-scalar (per-trait) phylogenetic random intercept. Each trait carries an independent length-`n_species` draw $\mathbf{p}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{phy}}^2 \mathbf{C}_{\text{phy}})$ with **one shared scaling** σ_{phy}^2 across all traits. Formerly `phylo(species)` – same engine, new name. Compare to `phylo_indep()` (D independent variances) and `phylo_latent()` (K-dim factor decomposition).

Usage

```
phylo_scalar(species, tree = NULL, vcv = NULL, A = NULL, Ainv = NULL)
```

Arguments

species	Unquoted column name for the species factor.
tree	An <code>ape::phylo</code> object. Canonical.
vcv	A tip-only phylogenetic correlation matrix (<code>n_species</code> x <code>n_species</code>). Legacy alias of <code>A =</code> .
A	Tip-level relatedness matrix (<code>n_species</code> x <code>n_species</code>) – alias of <code>vcv =</code> , aligned with the <code>animal_*</code> family’s argument naming.
Ainv	Sparse precision matrix (inverse of <code>A</code>).

Details

Pass the phylogeny via `tree = phylo` (canonical, sparse $\mathbf{A}^{-1}$; Hadfield & Nakagawa 2010) or `vcv = Cphy` ([**Superseded**], dense). See `phylo_latent()` for the full discussion of the two paths.

Value

A formula marker; never evaluated.

See Also

`phylo_latent()`, `phylo_indep()`, `phylo_dep()`, `phylo()` (deprecated alias).

Examples

```
tree <- ape::rcoal(8)
# Match the simulator species levels to the covariance matrix row order.
tree$tip.label <- as.character(seq_len(8))
sim <- simulate_site_trait(
  n_sites = 1, n_species = 8, n_traits = 3,
  mean_species_per_site = 8,
  Cphy = ape::vcv(tree, corr = TRUE),
```

```

    sigma2_phy = rep(0.3, 3), seed = 1
  )
  sim$data$species <- factor(sim$data$species, levels = tree$tip.label)
  fit <- gllvmTMB(
    value ~ 0 + trait + phylo_scalar(species),
    data      = sim$data,
    trait     = "trait",
    unit      = "species",
    cluster   = "species",
    phylo_tree = tree
  )

```

phylo_signal_mi

Phylogenetic-signal diagnostic for a modelled missing predictor

Description

For a `gllvmTMB()` fit with a phylogenetic missing-predictor covariate model ($\text{mi}(x)$ with `impute = list(x = x ~ ... + phylo(1 | species, tree = tree))`), `phylo_signal_mi()` reports an effective phylogenetic-signal statistic for the covariate model and flags the weak-signal case. Phylogenetic imputation helps when the signal in x is strong and degrades toward the independent (no-borrowing) model when it is weak; forcing a phylogenetic prior on a phylogenetically-unstructured trait adds noise rather than information (Penone et al. 2014 *Methods Ecol. Evol.* 5:961-970; Molina-Venegas 2024 *Methods Ecol. Evol.*; Goolsby, Bruggeman & Ane 2017 *Methods Ecol. Evol.* 8:22-27, Rphylopars).

Usage

```
phylo_signal_mi(object, variable = NULL, threshold = 0.1, warn = FALSE)
```

Arguments

<code>object</code>	A <code>gllvmTMB</code> fit with a phylogenetic missing-predictor model.
<code>variable</code>	Optional missing-predictor name; defaults to the only modelled missing predictor.
<code>threshold</code>	Effective-lambda threshold below which the signal is flagged as weak (default 0.1).
<code>warn</code>	Logical; when TRUE, warn against over-interpreting conditional modes when phylogenetic signal is weak. Default FALSE (the function returns the statistic silently).

Details

The statistic is an effective Pagel's lambda for the covariate, $\text{lambda} = \text{sd}_x^2 / (\text{sd}_x^2 + \text{sigma}_x^2)$, the fraction of species-level variance in x explained by the phylogenetic field (sd_x) relative to the i.i.d. residual (sigma_x) – the Pagel partition of the covariate model. lambda near 1 = strong signal (informative borrowing); lambda near 0 = weak signal (the covariate model is approximately

equivalent to an independent model). This is a verification / interpretation aid, not part of the fit; it does not change estimates.

Value

A list with `variable`, `lambda` (effective Pagel lambda), `sd_x`, `sigma_x`, `weak` (logical), and `threshold`.

See Also

`imputed()` for conditional modes; `gllvmTMB()` for the `impute = phylo` covariate model.

phylo_slope	<i>Phylogenetic random slope on a continuous covariate</i>
-------------	--

Description

Adds a per-species random slope on a continuous covariate `x`, with species-level slopes correlated according to the supplied phylogeny. Mathematically:

Usage

```
phylo_slope(formula)
```

Arguments

<code>formula</code>	<code>x species</code> style formula (LHS is the continuous covariate column name; RHS must be the species factor).
----------------------	---

Details

$$\eta_{it} += \beta_{\text{phy}}(i) x_{io}, \quad \beta_{\text{phy}} \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{slope}}^2 \mathbf{A}_{\text{phy}}).$$

The slope vector β_{phy} has length n_{species} (one slope per species). It is shared across traits – the same $\beta_{\text{phy}}(i)$ multiplies x_{io} for every trait t of species i . This is the "random regression" pattern from quantitative genetics, with the phylogenetic A-matrix as the slope-covariance prior.

Used inside a `gllvmTMB()` formula:

```
value ~ 0 + trait + (0 + trait):x +      # fixed-effect of x per trait
      phylo_latent(species, d = 2) +    # main phylogenetic random effect
      phylo_slope(x | species)          # phylo random slope on x
```

Shares the supplied tree / VCV with `phylo_latent()` when both terms are present, but constructs its own RHS-indexed precision and row map (sparse via `phylo_tree =`, dense via `phylo_vcv =`). This lets the slope grouping differ safely from the top-level cluster.

Scope:

This helper supports:

- **One** continuous covariate x (a single column name).
- **One** shared slope variance σ_{slope}^2 .
- **Slopes shared across traits** (same $\beta_{\text{phy}}(i)$ for every trait of species i).

A separate multi-predictor long-format alias is available when the RHS is the response-column factor: `phylo_slope(lat + temp | trait, tree = tree)` means `phylo_dep(0 + lat + temp | trait, tree = tree)`, so the two slope deviations have a full covariance. Use `||` instead of `|` for the diagonal `phylo_indep()` route. This alias is Gaussian-only, has no random intercept, and is intentionally outside the 5 x 3 trait-covariance grid. It does not change the one-predictor helper described above.

For trait-specific intercept-and-slope covariance, use the corresponding augmented `phylo_indep()`, `phylo_latent()`, or `phylo_dep()` syntax.

Value

A formula marker; never evaluated.

See Also

[phylo_latent\(\)](#), [phylo_scalar\(\)](#).

Examples

```
tree <- ape::rcoal(20)
# Match the simulator species levels to the covariance matrix row order.
tree$tip.label <- as.character(seq_len(20))
sim <- simulate_site_trait(
  n_sites = 10, n_species = 20, n_traits = 3,
  mean_species_per_site = 10,
  Cphy = ape::vcv(tree, corr = TRUE),
  sigma2_phy = rep(0.3, 3), seed = 1
)
sim$data$species <- factor(sim$data$species, levels = tree$tip.label)
sim$data$x <- rnorm(nrow(sim$data))
fit <- gllvmTMB(
  value ~ 0 + trait + (0 + trait):x + phylo_slope(x | species),
  data      = sim$data,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_tree = tree
)
```

phylo_unique	<i>Per-trait</i>	<i>independent</i>	<i>phylogenetic</i>	<i>random</i>	<i>intercepts:</i>
	phylo_unique(species)				

Description

[Deprecated]

Usage

```
phylo_unique(species, tree = NULL, vcv = NULL, A = NULL, Ainv = NULL)
```

Arguments

species	Unquoted column name for the species factor.
tree	An <code>ape::phylo</code> object. Canonical.
vcv	A tip-only phylogenetic correlation matrix (<code>n_species</code> x <code>n_species</code>). Legacy alias of <code>A</code> .
A	Tip-level relatedness matrix (<code>n_species</code> x <code>n_species</code>) – alias of <code>vcv</code> , aligned with the <code>animal_*</code> family’s argument naming.
Ainv	Sparse precision matrix (inverse of <code>A</code>).

Details

`phylo_unique()` is soft-deprecated as compatibility syntax in `gllvmTMB` 0.2.0. Use `phylo_indep()` for standalone marginal diagonal phylogenetic tiers. Paired explicit-Psi use remains accepted; use `phylo_latent(..., unique = TRUE)` when the folded latent term itself should carry this diagonal companion.

Canonical name for the **D independent** phylogenetic random intercepts. Each trait t gets its own variance $\sigma_{\text{phy},t}^2$ on the same phylogenetic correlation matrix $\mathbf{C}_{\text{phy}}$; the per-trait random vectors are otherwise independent. Mathematically:

$$\mathbf{p}_t \sim \mathcal{N}(\mathbf{0}, \sigma_{\text{phy},t}^2 \mathbf{C}_{\text{phy}}), \quad t = 1, \dots, T,$$

giving a phylogenetic counterpart of `unique()` – one independent variance per trait, all sharing the same phylogenetic precision.

Compare to `phylo_scalar()` (one shared variance), which collapses the T variances to a single σ_{phy}^2 ; and `phylo_latent()` (K factors), which induces *across-trait* correlation through a low-rank loading matrix.

Two co-fitting modes:

`phylo_unique()` has two complementary modes:

Alone (legacy) When written without a paired `phylo_latent(species, d = K)`, the engine implements the term internally as `phylo_latent(species, d = T)` with a diagonal $\mathbf{\Lambda}_{\text{phy}}$. The diagonal entries of $\mathbf{\Lambda}_{\text{phy}}$ are the per-trait phylogenetic SDs. This path stays for backward compatibility.

Paired with `phylo_latent(..., unique = FALSE)` (compatibility) When written together with `phylo_latent(species, d = K, unique = FALSE)`, the two terms co-fit as **separate covariance components**:

$$\Sigma_{\text{phy}} = \underbrace{\mathbf{\Lambda}_{\text{phy}} \mathbf{\Lambda}_{\text{phy}}^{\top}}_{\text{shared (rank K)}} + \underbrace{\mathbf{\Psi}_{\text{phy}}}_{\text{per-trait unique}}.$$

Λ_{phy} is filled by the rank-K shared latent factors; Ψ_{phy} carries the per-trait phylogenetic variances not absorbed by the K shared axes. `phylo_latent(species, d = K, unique = TRUE)` fits this same manuscript-aligned PGLLVM decomposition as a single term (Hadfield & Nakagawa 2010; Meyer & Kirkpatrick 2008; Halliwell et al. 2025). Replication (multiple sites per species) is required to break the Ψ_{phy} / Ψ_{non} confound at the species level.

Use `extract_Sigma()` with `level = "phy"` and `part = "shared", "unique",` or `"total"` to pull each component, the diagonal `psi_phy`, or their sum.

Pass the phylogeny via `tree = phylo` (canonical, sparse $\mathbf{A}^{-1}$) or `vcv = Cphy` ([**Superseded**], dense). See `phylo_latent()` for the full discussion of the two paths.

Value

A formula marker; never evaluated.

References

- **Hadfield, J. D. & Nakagawa, S.** (2010) General quantitative genetic methods for comparative biology: phylogenies, taxonomies and multi-trait models for continuous and categorical characters. *Journal of Evolutionary Biology* 23(3): 494-508. doi:10.1111/j.1420-9101.2009.01915.x
- **Meyer, K. & Kirkpatrick, M.** (2008) Perils of parsimony: properties of reduced-rank estimates of genetic covariance matrices. *Genetics* 178(4): 2223-2240.

See Also

`phylo_scalar()`, `phylo_latent()`, `phylo_indep()`, `phylo_dep()`, `extract_Sigma()`.

Examples

```
tree <- ape::rcoal(8)
# Match the simulator species levels to the covariance matrix row order.
tree$tip.label <- as.character(seq_len(8))
sim <- simulate_site_trait(
  n_sites = 1, n_species = 8, n_traits = 3,
  mean_species_per_site = 8,
  Cphy = ape::vcv(tree, corr = TRUE),
  sigma2_phy = rep(0.3, 3), seed = 1
)
sim$data$species <- factor(sim$data$species, levels = tree$tip.label)
fit <- gllvmTMB(
  value ~ 0 + trait + phylo_unique(species),
  data      = sim$data,
  trait     = "trait",
  unit      = "species",
  cluster   = "species",
  phylo_tree = tree
)
```

plot.gllvmTMB_multi *Plot a fitted multivariate gllvmTMB model*

Description

Produces a variety of ggplot2 visualisations for a stacked-trait multivariate GLLVM. Dispatches on type to one of seven panels:

Usage

```
## S3 method for class 'gllvmTMB_multi'
plot(
  x,
  type = c("correlation", "correlation_ellipse", "loadings", "integration",
    "communality", "variance", "ordination"),
  level = c("unit", "unit_obs"),
  boot = NULL,
  axes = c(1L, 2L),
  rotation = c("varimax", "none", "promax"),
  order_axes = TRUE,
  sign_anchor = c("auto", "none"),
  anchor_traits = NULL,
  standardize_loadings = FALSE,
  ...
)
```

Arguments

x	A fit returned by <code>gllvmTMB()</code> .
type	One of "correlation", "correlation_ellipse", "loadings", "integration", "communality", "variance", "ordination".
level	"unit" (between-unit) or "unit_obs" (within-unit). Legacy aliases "B" and "W" are accepted with a deprecation warning. Used by "loadings" (which level to plot; the default <code>c("unit", "unit_obs")</code> means "both available levels, faceted side-by-side"; pass a length-1 string to plot one tier) and "ordination" (single level required; omitted level defaults to "unit"). Ignored for "correlation" (which always shows both if available), "integration", and "variance". <i>Note:</i> the default <code>level = c("unit", "unit_obs")</code> is intentionally a length-2 vector, not the usual <code>match.arg</code> shortcut. The dispatcher does not call <code>match.arg(level)</code> itself; each helper inspects <code>level</code> and decides whether to plot one tier or both. If you copy one of these helpers into your own code, mirror that pattern rather than reflexively calling <code>match.arg(level)</code> (which would silently collapse the default to "unit" and drop the <code>unit_obs</code> panel).
boot	Optional bootstrap object. This can be either a <code>bootstrap_Sigma()</code> result or a list with elements <code>repeatability</code> , <code>communality_B</code> , <code>communality_W</code> , each a data frame with columns <code>trait</code> , <code>lower</code> , <code>upper</code> . A <code>bootstrap_Sigma()</code> object

	can add correlation intervals to "correlation" / "correlation_ellipse", whiskers to "integration", and c^2 boundary intervals to "communality". Default NULL skips interval overlays.
axes	Length-2 or length-3 integer vector for "ordination" when $d \geq 2$. Length 2 draws a single biplot. Length 3 draws a static pair-grid of the three axis pairs. For $d = 3$, the default $c(1, 2)$ is promoted to $c(1, 2, 3)$ so all three axes are visible. Ignored when $d = 1$.
rotation	One of "varimax", "none", or "promax" for "ordination" plots. The default "varimax" uses rotated, shared-variance-ordered, sign-anchored axes for interpretation. Use "none" to show the raw computational orientation. Rotation makes the biplot easier to label; use Sigma, correlations, communality, and uniqueness as the primary quantitative summaries.
order_axes	Logical. For rotated "ordination" plots, reorder latent axes by decreasing shared variance after rotation. Default TRUE. Ignored when rotation = "none".
sign_anchor	One of "auto" or "none". For rotated "ordination" plots, "auto" flips each axis so its anchor trait has a positive loading. Default "auto". Ignored when rotation = "none".
anchor_traits	Optional character vector of trait names used for sign anchoring in rotated "ordination" plots. Supply one trait per axis after any order_axes step. Missing axes fall back to the trait with the largest absolute loading. Ignored when rotation = "none".
standardize_loadings	Logical. For "ordination" plots, divide each trait loading by the square root of that trait's model-implied total variance before drawing arrows. This puts arrows on a correlation-like scale for mixed-scale traits. It changes the displayed arrow scale, not the model's communality or variance decomposition. Default FALSE.
...	Currently unused.

Details

- "correlation" Combined heatmap of trait correlations. Upper triangle = between-unit correlations (level = "unit"), lower triangle = within-unit correlations (level = "unit_obs"), diagonal = 1. Falls back to whichever level is present if the other tier is absent. Optional boot intervals are carried in the plot data when supplied.
- "correlation_ellipse" Ellipse matrix of trait correlations. Ellipse direction and eccentricity encode the sign and strength of the correlation. This is the Figure-3-style alternative to the tile heatmap. Optional boot intervals mark correlations whose interval does not cross zero with black borders and stars. These bootstrap intervals are recovery-grade, and their coverage properties are not yet established: read a marked cell as "the interval excludes zero at the fitted point", not as a calibrated significance test.
- "loadings" Tile heatmap of Lambda_B (and Lambda_W if present), faceted by level. Rows = traits, columns = factors. Pinned cells (from lambda_constraint) are drawn with a heavy outline.
- "integration" Dot-and-whisker plot of repeatability (ICC), between-tier communality and within-tier communality per trait, sorted by repeatability. Optional whiskers from a boot object (skipped if boot = NULL).

- "communality" Figure-3-style stacked bars of per-trait communality (c^2 , shared latent proportion) and uniqueness ($1 - c^2$) for the available latent tiers. Optional boot intervals are drawn on the c^2 boundary when supplied.
- "variance" Stacked-bar variance partition per trait, using `extract_proportions(format = "long")`. One bar per trait, stacks summing to 1.
- "ordination" Dimension-aware latent-score ordination. 1D fits ($d = 1$) get a horizontal score strip with trait loading lollipops; 2D fits get a standard biplot; 3D fits get a static pair-grid biplot for the three axis pairs. For $d > 3$, pick two or three axes via `axes`.

Value

A ggplot object with a `gllvmTMB_meta` attribute describing the plot type, source extractor, covariance level, interval status, and rotation status. Metadata also carries extractor notes when a plotted summary has an important caveat. Plots also carry a `gllvmTMB_data` attribute with the prepared plotting data; ordination stores separate scores and loadings tables.

<code>plot.gllvmTMBmesh</code>	<i>Plot a gllvmTMB SPDE mesh</i>
--------------------------------	----------------------------------

Description

Draw mesh edges and the observed locations. This is the plot method for meshes created by `make_mesh()`.

Usage

```
## S3 method for class 'gllvmTMBmesh'
plot(x, ...)

## S3 method for class 'sdmTMBmesh'
plot(x, ...)
```

Arguments

`x` A `gllvmTMBmesh` created by `make_mesh()`.
`...` Passed to `graphics::plot()`.

Details

`plot.sdmTMBmesh()` is a temporary legacy compatibility method. New meshes have class `gllvmTMBmesh` and should be plotted with `plot.gllvmTMBmesh()`.

Value

Invisibly returns `x` after drawing mesh edges, observations, and optional k-means centres.

plot.profile_derived *Plot a profile-derived LR curve*

Description

S3 method dispatched on the profile_derived parent class shared by the public derived-quantity curve object returned from `profile_phylo_signal()`. The diagonal-companion ratio and nonlinear communality, correlation, and proportion curve prototypes are internal and withheld from the public release. Mirrors `plot.profile_loadings()`: dotted chisq cutoff hline, solid grey vline at the estimate, dashed grey vlins at the inverted CI bounds (when `interval = TRUE`), blue curve with white-filled points, faceted by target with free x-scales.

Usage

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

Arguments

x	A profile_derived data.frame.
interval	Logical; draw inverted CI bounds (default TRUE).
...	Reserved for future options.

Value

A ggplot object.

plot.profile_loadings *Plot a profile_loadings object (LR U-shape per entry)*

Description

S3 method dispatched from `plot()`. Draws the delta-deviance curve per entry, the chisq cutoff as a dotted horizontal line, the MLE as a grey vertical line, and (optionally) the inverted CI bounds as dashed verticals.

Usage

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

Arguments

x	A profile_loadings data.frame from <code>loading_profile()</code> .
interval	Logical; draw inverted CI bounds (default TRUE).
...	Reserved for future options.

Details

Design borrowed from `drmTMB`'s `plot.profile.drmTMB()`.

Value

A `ggplot` object.

<code>plot_anisotropy</code>	<i>Plot the isotropic spatial range from a <code>gllvmTMB</code> model</i>
------------------------------	--

Description

`gllvmTMB` currently fits isotropic SPDE/GMRF spatial fields. Consequently, these helpers draw a circle whose radius is the fitted practical range, $\sqrt{8} / \kappa$. Equal axes describe the model assumption ($H = I$); they are not evidence that anisotropy was estimated. Scope: the fitted isotropic-range display is covered by focused tests; the broader spatial-family evidence remains partial; directional anisotropy, delta/spatiotemporal fields, and new likelihoods are rejected from this plotting contract.

Usage

```
plot_anisotropy(object, return_data = FALSE)
```

```
plot_anisotropy2(object, model = 1L)
```

Arguments

<code>object</code>	A fitted <code>gllvmTMB_multi</code> model containing an intercept-only or random-slope spatial field.
<code>return_data</code>	If <code>TRUE</code> , return the range geometry instead of a plot.
<code>model</code>	Retained for compatibility. Only <code>model = 1</code> is supported because <code>gllvmTMB</code> has one shared spatial range parameter.

Value

`plot_anisotropy()` returns a `ggplot2` object, or a data frame when `return_data = TRUE`. `plot_anisotropy2()` draws the same geometry with base graphics and invisibly returns a list containing `data`, `range`, `kappa`, `H = diag(2)`, and `anisotropy_estimated = FALSE`.

Scope

These functions do not estimate or diagnose directional anisotropy. Delta and spatiotemporal spatial models are not supported by this plotting contract and fail explicitly.

plot_correlations

*Plot pairwise trait correlations with intervals***Description**

plot_correlations() turns the tidy rows returned by [extract_correlations\(\)](#) into a horizontal forest plot, confidence-eye plot, heatmap, or ellipse matrix. Forest and confidence-eye styles keep point estimates visible as open points when interval bounds are missing, and draw interval information only for rows with finite lower and upper bounds. Matrix styles show the same tidy rows without requiring users to reconstruct or hand-index a correlation matrix. For fitted-object calls, open points can often be investigated by trying method = "bootstrap" or another interval method supported by [extract_correlations\(\)](#).

Usage

```
plot_correlations(
  x,
  tier = "all",
  pair = NULL,
  level = 0.95,
  method = c("none", "fisher-z", "profile", "wald", "bootstrap"),
  n_eff = NULL,
  nsim = 500L,
  seed = NULL,
  link_residual = c("auto", "none"),
  facet = c("level", "none"),
  sort = c("estimate", "magnitude", "trait", "level"),
  show_intervals = NULL,
  style = c("interval", "eye", "raindrop", "heatmap", "ellipse", "oval"),
  eye_level = level,
  raindrop_level = NULL,
  triangle = c("full", "lower", "upper"),
  include_diagonal = TRUE,
  label = TRUE,
  label_type = c("auto", "estimate", "ci", "estimate_ci", "none"),
  label_digits = 2,
  matrix_layout = c("by_level", "estimate_ci", "levels"),
  title = NULL,
  subtitle = NULL,
  caption = NULL
)
```

Arguments

x Either a fit returned by [gllvmTMB\(\)](#), a bootstrap_Sigma object with R_B / R_W summaries, or a data frame returned by [extract_correlations\(\)](#). Data

	frames must contain tier, trait_i, trait_j, correlation, lower, upper, and method.
tier, pair, level, method, n_eff, nsim, seed, link_residual	Passed to extract_correlations() when x is a fitted model. Ignored when x is already a data frame.
facet	One of "level" (default) or "none". Facetting by level keeps repeated trait pairs readable when several tiers are present.
sort	Row ordering: "estimate" (default), "magnitude", "trait", or "level".
show_intervals	Logical or NULL. If NULL (default), finite lower/upper bounds are drawn as horizontal intervals for style = "interval" and omitted for style = "eye". Set TRUE to overlay interval lines on confidence eyes. Rows without finite bounds remain visible as points.
style	One of "interval" (default), "eye", "raindrop", "heatmap", "ellipse", or "oval". "eye" draws a confidence eye: a pale frequentist compatibility shape reconstructed from the estimate and finite interval bounds, plus a hollow, sign-coloured estimate circle. Correlation rows use Fisher's z scale. The shape is not a posterior density. "heatmap" draws a trait-by-trait matrix with colour showing the point estimate. "ellipse"/"oval" draws correlation ellipses in matrix cells; ellipse tilt and width encode sign and strength. "raindrop" is accepted as a compatibility alias for "eye", and "oval" as an alias for "ellipse".
eye_level	Confidence level represented by the supplied interval bounds when style = "eye". Defaults to level, so fitted-object calls stay aligned with extract_correlations() .
raindrop_level	Compatibility alias for eye_level.
triangle	Matrix triangle to show for style = "heatmap" or style = "ellipse" when matrix_layout = "by_level": "full" (default), "lower", or "upper".
include_diagonal	Logical. Include muted diagonal self-correlation cells in matrix styles?
label	Logical. Print cell labels in matrix styles?
label_type	For matrix styles, one of "auto" (default), "estimate", "ci", "estimate_ci", or "none". With matrix_layout = "estimate_ci", "auto" prints point estimates in the upper triangle and interval bounds in the lower triangle. Interval labels are printed only where finite lower and upper bounds are supplied.
label_digits	Non-negative whole number of decimal places for matrix cell labels.
matrix_layout	Matrix-cell layout for heatmap and ellipse styles. "by_level" facets levels and uses triangle directly. "estimate_ci" uses the full matrix for each level: upper triangle labels point estimates and lower triangle labels supplied interval bounds. "levels" combines exactly two levels into one full matrix: upper triangle shows the first level and lower triangle shows the second level.
title, subtitle, caption	Optional scalar character labels for matrix styles. NULL keeps publication-safe defaults.

Details

Julia bridge (engine = "julia") fits currently fail with an explicit gate because this helper depends on interval-bearing [extract_correlations\(\)](#) rows. Use [plot_Sigma_table\(\)](#) or [plot_Sigma_heatmap\(\)](#) for point-only Julia bridge correlation displays.

Scope: the helper plots tidy cross-trait correlation rows from `extract_correlations()`, extracts those rows from a fit returned by `gllvmTMB()` (this is covered by the package's extractor-contract validation tests), or converts `bootstrap_Sigma()` correlation summaries to the same plotting schema (also covered by those tests). Matrix heatmaps and ellipse/oval displays, including upper-estimate/lower-CI and two-level upper/lower layouts, are point-estimate summaries with optional numeric interval labels (covered separately). The plot does not compute new intervals; it displays whatever interval method the input rows already contain. Matrix-style visual comparisons against known truth remain article code rather than part of this helper.

Value

A `ggplot2` plot object with `gllvmTMB_meta` and `gllvmTMB_data` attributes.

See Also

`extract_correlations()`, `plot_Sigma_table()`, `plot.gllvmTMB_multi()`.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  cors <- data.frame(
    tier = c("unit", "unit", "unit_obs"),
    trait_i = c("length", "length", "length"),
    trait_j = c("mass", "wing", "mass"),
    correlation = c(0.42, -0.18, 0.10),
    lower = c(0.12, -0.45, NA),
    upper = c(0.66, 0.12, NA),
    method = c("fisher-z", "fisher-z", "none")
  )
  plot_correlations(cors)
  plot_correlations(cors, style = "heatmap", matrix_layout = "estimate_ci")
}
```

`plot_loadings_confidence_eye`

Confidence Eye plot for per-species loading uncertainty

Description

A bar-and-whisker visualisation of raw or standardised loadings with point estimates and Wald (or other-method) confidence intervals from `loading_ci()`. Default rendering uses the "Confidence Eye" contract: a pale CI region (`geom_linerange`) and a hollow point estimate (`geom_point`, `shape = 21`, `fill white`) so the visual weight rests on the uncertainty rather than the point.

Usage

```
plot_loadings_confidence_eye(
  fit,
  level = c("unit", "unit_obs"),
  method = "wald",
  conf_level = 0.95,
  null_region = NULL,
  sort_by = c("trait_order", "magnitude"),
  ylim = NULL,
  loading_scale = NULL
)
```

Arguments

<code>fit</code>	A multivariate <code>gllvmTMB()</code> fit, OR a data frame already produced by <code>loading_ci()</code> / <code>flag_unreliable_loadings()</code> .
<code>level</code> , <code>method</code> , <code>conf_level</code> , <code>loading_scale</code>	Forwarded to <code>loading_ci()</code> when <code>fit</code> is a fit object. Ignored when <code>fit</code> is already a <code>loading_ci()</code> data frame, except that an explicitly supplied <code>loading_scale</code> must match the frame's recorded scale. The y-axis label reports the displayed scale.
<code>null_region</code>	Optional length-2 numeric drawn as a shaded "biologically negligible" band so the reader can tell at a glance which CIs sit outside it. If supplied, also colours points by reliability via <code>flag_unreliable_loadings()</code> . Default NULL (no band drawn).
<code>sort_by</code>	Character, one of "trait_order" (the order of the trait factor, the default) or "magnitude". Note "magnitude" applies a single global trait ordering keyed by each trait's largest absolute loading across axes; the shared discrete x-axis cannot reorder traits within individual facets.
<code>ylim</code>	Optional length-2 numeric vector clipping the y-axis via <code>coord_cartesian()</code> . Useful when one degenerate loading estimate would otherwise blow out the scale and crush the rest. Default NULL (no clipping).

Details

The closest published precedent for an uncertainty-aware per-species loading display in JSMD is the site-score uncertainty plot of Hoegh & Roberts (2020, doi:10.1002/ece3.5752), but that targets site scores rather than species loadings. To my knowledge no equivalent plot of per-species Lambda uncertainty exists in the ML/TMB JSMD lineage (`gllvm`, `sjSDM`, `ecoCopula`).

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
plot_loadings_confidence_eye(fit, level = "unit",
                             null_region = c(-0.1, 0.1))
```

Value

A ggplot object.

See Also

[loading_ci\(\)](#), [flag_unreliable_loadings\(\)](#).

plot_rotated_loadings *Plot a rotated loading matrix*

Description

`plot_rotated_loadings()` turns the tidy rows from [extract_rotated_loadings_table\(\)](#) into a publication-oriented loading matrix. It can also extract the table from a fitted [gllvmTMB\(\)](#) model. Rows are traits, columns are latent axes, and colour shows the signed loading after the requested rotation, axis ordering, sign anchoring, and loading scale.

Usage

```
plot_rotated_loadings(
  x,
  level = "unit",
  method = c("varimax", "promax", "none"),
  order_axes = TRUE,
  sign_anchor = c("auto", "none"),
  anchor_traits = NULL,
  loading_scale = c("standardized", "raw"),
  sort = c("dominant", "abs_loading", "trait"),
  show_values = NULL,
  digits = 2L,
  limits = NULL,
  facet = c("level", "none")
)
```

Arguments

<code>x</code>	A fit returned by gllvmTMB() or a data frame returned by extract_rotated_loadings_table() .
<code>level</code>	"unit" (between-unit) or "unit_obs" (within-unit), passed to extract_rotated_loadings_table() when <code>x</code> is a fitted model. Deprecated aliases "B" and "W" are still accepted with a warning.
<code>method</code>	One of "varimax", "promax", or "none"; passed to extract_rotated_loadings_table() for fitted-model calls.
<code>order_axes</code>	Logical. When TRUE, reorder rotated axes by decreasing raw shared variance before plotting. Ignored when <code>method = "none"</code> .
<code>sign_anchor</code>	One of "auto" or "none". "auto" flips each rotated axis so its anchor trait has a positive loading. Ignored when <code>method = "none"</code> .
<code>anchor_traits</code>	Optional character vector of trait names used for sign anchoring. Supply one trait per axis after ordering. Axes without a supplied anchor use the trait with the largest absolute loading.

loading_scale	One of "standardized" or "raw" for fitted-model calls. The default "standardized" puts loadings on a correlation-like scale for figures. Ignored when x is already a data frame.
sort	Trait ordering. "dominant" groups traits by the axis on which they have their largest absolute loading. "abs_loading" orders by each trait's largest absolute loading. "trait" preserves the incoming trait order.
show_values	Logical or NULL. When NULL, numeric tile labels are shown for matrices with at most 80 cells.
digits	Number of decimal places for tile labels.
limits	Optional two-number colour-scale limits. When NULL, limits are symmetric around zero and based on the plotted loadings.
facet	One of "level" or "none". Data frames containing more than one covariance level are faceted by level by default.

Details

Scope: the helper plots point-estimate rotated loading rows from [extract_rotated_loadings_table\(\)](#) for fitted latent() components (this builds on the package's rotated-loadings extraction validation). The plot does not compute or display loading uncertainty intervals and does not make rotated axes uniquely biological. Bootstrap- or simulation-aligned loading uncertainty remains a later inference slice.

Value

A ggplot2 plot object with gllvmTMB_meta and gllvmTMB_data attributes.

See Also

[extract_rotated_loadings_table\(\)](#), [rotate_loadings\(\)](#), [plot.gllvmTMB_multi\(\)](#).

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  rows <- data.frame(
    level = "unit",
    trait = rep(c("body", "wing", "bill"), times = 2),
    axis = rep(c("LV1", "LV2"), each = 3),
    loading = c(0.72, 0.61, -0.08, 0.05, -0.18, 0.66),
    abs_loading = abs(c(0.72, 0.61, -0.08, 0.05, -0.18, 0.66)),
    axis_variance = rep(c(1.1, 0.5), each = 3),
    axis_share = rep(c(0.69, 0.31), each = 3),
    rotation = "varimax",
    order_axes = TRUE,
    sign_anchor = "auto",
    anchor_trait = rep(c("body", "bill"), each = 3),
    loading_scale = "standardized"
  )
  plot_rotated_loadings(rows)
}
```

plot_Sigma_comparison *Plot Sigma-table estimates against a known truth matrix*

Description

plot_Sigma_comparison() turns [compare_Sigma_table\(\)](#) rows into a row-labelled comparison plot. The default style = "difference" shows estimate - truth for each covariance or correlation row; style = "scatter" shows estimate versus truth with a one-to-one reference line. Segments in these plots are comparison residuals, not confidence intervals.

Usage

```
plot_Sigma_comparison(
  x,
  truth,
  level = "unit",
  part = c("total", "shared", "unique"),
  measure = c("covariance", "correlation"),
  entries = c("upper", "unique", "all", "offdiag", "lower", "diag"),
  link_residual = c("auto", "none"),
  include_diagonal = FALSE,
  facet = c("level", "comparison", "none"),
  sort = c("abs_error", "error", "estimate", "truth", "trait", "level"),
  style = c("difference", "scatter")
)
```

Arguments

x	A fit returned by gllvmTMB() , an admitted engine = "julia" bridge fit, a data frame returned by extract_Sigma_table() , or a data frame already returned by compare_Sigma_table() .
truth	Square numeric covariance or correlation matrix passed to compare_Sigma_table() . May be omitted only when x already contains truth, error, abs_error, and comparison_status columns.
level, part, measure, entries, link_residual	Passed to compare_Sigma_table() when truth is supplied.
include_diagonal	Logical. Include diagonal rows if they are present? The default is FALSE because variances are usually on a different scale from pairwise covariance/correlation rows.
facet	One of "level" (default), "comparison", or "none". Use "comparison" when precomputed rows contain a comparison column, for example to compare two model specifications against the same truth.
sort	Row ordering for style = "difference": "abs_error" (default), "error", "estimate", "truth", "trait", or "level".
style	One of "difference" (default) or "scatter".

Details

Scope: the helper plots `compare_Sigma_table()` rows or builds them from a fitted model / Sigma table plus one supplied truth matrix (covered by the package's comparison-table validation tests, which build on the underlying Sigma-table extraction tests). It is a visual comparison helper only: it does not run simulations, compute intervals, or validate calibration. Article-specific simulation summaries and richer calibration plots remain future work.

Value

A ggplot2 plot object with `gllvmTMB_meta` and `gllvmTMB_data` attributes.

See Also

`compare_Sigma_table()`, `plot_Sigma_table()`.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  rows <- data.frame(
    level = "unit",
    trait_i = c("length", "length", "mass"),
    trait_j = c("mass", "wing", "wing"),
    estimate = c(0.62, -0.10, 0.28),
    lower = NA_real_,
    upper = NA_real_,
    matrix = "R",
    component = "total",
    diagonal = FALSE,
    triangle = "upper",
    scale = "correlation"
  )
  truth_R <- matrix(
    c(1, 0.60, -0.05, 0.60, 1, 0.20, -0.05, 0.20, 1),
    3,
    byrow = TRUE,
    dimnames = list(
      c("length", "mass", "wing"),
      c("length", "mass", "wing")
    )
  )
  plot_Sigma_comparison(rows, truth_R, measure = "correlation")
}
```

Description

`plot_Sigma_heatmap()` turns rows from `extract_Sigma_table()` into a matrix-style heatmap. It is designed for articles that need to show the block structure of a covariance or correlation matrix without manually extracting Sigma, calling `cov2cor()`, or rebuilding `geom_tile()` layers. Heatmap cells are point estimates only; interval columns, when present, stay available in the returned plot data but are not displayed.

Usage

```
plot_Sigma_heatmap(
  x,
  level = "unit",
  part = c("total", "shared", "unique"),
  measure = c("correlation", "covariance"),
  entries = c("all", "unique", "upper", "lower", "offdiag", "diag"),
  link_residual = c("auto", "none"),
  include_diagonal = TRUE,
  facet = c("level", "none"),
  label = TRUE,
  label_digits = 2,
  title = NULL,
  subtitle = NULL,
  caption = NULL
)
```

Arguments

<code>x</code>	A fit returned by <code>gllvmTMB()</code> , an admitted engine = "julia" bridge fit, a <code>bootstrap_Sigma</code> object, or a data frame returned by <code>extract_Sigma_table()</code> . Data frames must contain <code>level</code> , <code>trait_i</code> , <code>trait_j</code> , <code>estimate</code> , <code>matrix</code> , <code>component</code> , <code>diagonal</code> , and <code>triangle</code> ; <code>i</code> and <code>j</code> columns are used for trait ordering when present.
<code>level</code> , <code>part</code> , <code>measure</code> , <code>entries</code> , <code>link_residual</code>	Passed to <code>extract_Sigma_table()</code> when <code>x</code> is a fitted model. The plotting default <code>entries = "all"</code> shows the full matrix.
<code>include_diagonal</code>	Logical. Include diagonal rows when present?
<code>facet</code>	One of "level" (default) or "none".
<code>label</code>	Logical. Print numeric estimates inside cells?
<code>label_digits</code>	Number of decimal places for cell labels.
<code>title</code> , <code>subtitle</code> , <code>caption</code>	Optional scalar character labels. NULL keeps the helper's publication-safe defaults.

Details

Scope: the helper plots point-estimate heatmaps from `extract_Sigma_table()` rows or extracts those rows from a fit returned by `gllvmTMB()`, an admitted engine = "julia" bridge fit, or from

a `bootstrap_Sigma` object (covered by the package's heatmap validation tests, which build on the underlying extractor-contract and Julia-bridge fit tests). It does not display interval bounds or compare fitted values to known truth. Use `plot_Sigma_table()` for interval forests or confidence eyes, and `plot_Sigma_comparison()` for estimate-vs-truth displays. `vdiffr` snapshots and richer multi-model layout helpers remain future figure work.

Value

A `ggplot2` plot object with `gllvmTMB_meta` and `gllvmTMB_data` attributes.

See Also

`extract_Sigma_table()`, `plot_Sigma_table()`, `plot_Sigma_comparison()`.

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  rows <- data.frame(
    level = "unit",
    trait_i = rep(c("length", "mass"), each = 2),
    trait_j = rep(c("length", "mass"), 2),
    estimate = c(1, 0.35, 0.35, 1),
    matrix = "R",
    component = "total",
    diagonal = c(TRUE, FALSE, FALSE, TRUE),
    triangle = c("diagonal", "lower", "upper", "diagonal")
  )
  plot_Sigma_heatmap(rows)
}
```

`plot_Sigma_table`

Plot report-ready Sigma table rows

Description

`plot_Sigma_table()` turns rows from `extract_Sigma_table()` into a forest-style covariance or correlation plot. It is designed for reports and articles that need to show selected covariance entries without indexing the Sigma matrix by hand. Rows without finite interval bounds are drawn as open points. To display Sigma uncertainty, pass interval-bearing rows, for example from a bootstrap workflow; `extract_Sigma_table()` itself currently leaves interval columns as placeholders.

Usage

```
plot_Sigma_table(
  x,
  level = "unit",
  part = c("total", "shared", "unique"),
  measure = c("covariance", "correlation"),
  entries = c("upper", "unique", "all", "offdiag", "lower", "diag"),
```

```

link_residual = c("auto", "none"),
include_diagonal = FALSE,
facet = c("level", "none"),
sort = c("estimate", "magnitude", "trait", "level"),
show_intervals = NULL,
style = c("interval", "eye", "raindrop"),
eye_level = 0.95,
raindrop_level = NULL
)

```

Arguments

x	A fit returned by <code>gllvmTMB()</code> , an admitted engine = "julia" bridge fit, a <code>bootstrap_Sigma</code> object, or a data frame returned by <code>extract_Sigma_table()</code> . Data frames must contain level, trait_i, trait_j, estimate, lower, upper, matrix, component, diagonal, and triangle.
level, part, measure, entries, link_residual	Passed to <code>extract_Sigma_table()</code> when x is a fitted model. The plotting default entries = "upper" shows each pairwise covariance/correlation once; pass entries = "diag" for variances or entries = "unique" for the diagonal plus upper triangle.
include_diagonal	Logical. Include diagonal rows if they are present in x? The default is FALSE because covariance diagonals are variances and are usually on a different interpretive scale than pairwise covariances.
facet	One of "level" (default) or "none".
sort	Row ordering: "estimate" (default), "magnitude", "trait", or "level".
show_intervals	Logical or NULL. If NULL (default), finite lower/upper bounds are drawn as horizontal intervals for style = "interval" and omitted for style = "eye". Set TRUE to overlay interval lines on confidence eyes. Rows without finite bounds remain visible as points.
style	One of "interval" (default), "eye", or "raindrop". "eye" draws a confidence eye: a pale frequentist compatibility shape reconstructed from the estimate and finite interval bounds, plus a hollow, sign-coloured estimate circle. Correlation rows use Fisher's z scale and covariance rows use the displayed estimate scale. The shape is not a posterior density. "raindrop" is accepted as a compatibility alias.
eye_level	Confidence level represented by the supplied interval bounds when style = "eye". Default 0.95.
raindrop_level	Compatibility alias for eye_level.

Details

Scope: the helper plots point-estimate rows from `extract_Sigma_table()` or extracts those rows from a fit returned by `gllvmTMB()` or an admitted engine = "julia" bridge fit (covered by the package's extractor-contract validation tests, including the Julia-bridge fit path). Interval columns are displayed when present and finite, but this helper does not compute Sigma intervals. Uncertainty propagation for arbitrary Sigma entries belongs in bootstrap/profile infrastructure.

Value

A ggplot2 plot object with gllvmTMB_meta and gllvmTMB_data attributes.

See Also

[extract_Sigma_table\(\)](#), [plot_correlations\(\)](#), [plot.gllvmTMB_multi\(\)](#).

Examples

```
if (requireNamespace("ggplot2", quietly = TRUE)) {
  sigma_rows <- data.frame(
    level = "unit",
    trait_i = c("length", "length", "mass"),
    trait_j = c("mass", "wing", "wing"),
    estimate = c(0.22, -0.08, 0.15),
    lower = NA_real_,
    upper = NA_real_,
    matrix = "Sigma",
    component = "total",
    diagonal = FALSE,
    triangle = "upper"
  )
  plot_Sigma_table(sigma_rows)
}
```

predict.gllvmTMB_multi

Predict from a fitted gllvmTMB model

Description

Returns the linear predictor or inverse-link response at each observation in the training data, or at user-supplied newdata. For mixed-family fits, type = "response" uses the row's own trait/family inverse link rather than the first trait's link.

Usage

```
## S3 method for class 'gllvmTMB_multi'
predict(
  object,
  newdata = NULL,
  type = c("link", "response"),
  re_form = ~.,
  se.fit = FALSE,
  ...
)
```

Arguments

object	A fit returned by <code>gllvmTMB()</code> .
newdata	Optional new data frame. If NULL, predictions are produced for the training rows.
type	One of "link" (default) or "response". "response" includes the row's offset , so on a fit with an effort or support offset it returns an expected count <i>at that effort</i> , or a detection probability <i>at that support</i> – not a relative intensity. For a map or any effort-free comparison, set the offset variable to zero in newdata (e.g. <code>newdata\$log_support <- 0</code>) and predict from that; the offset is re-evaluated against newdata, so this is exact rather than an approximation. On a mixed-family fit the returned est column mixes scales by design (expected counts beside probabilities). The fit's family/source column is returned alongside it so each row's scale is identifiable.
re_form	Random-effect formula controlling which random effects are <i>included</i> in the predicted linear predictor. The default <code>~ .</code> includes them; <code>~ 0</code> , NA, and numeric <code>0</code> all request the fixed-effects-only / population-mean prediction. Any other value is not a supported form and warns rather than silently including the random effects. Honoured on both the training-row and the newdata path (before 0.7.1 it was read only on the newdata path, and there only as the literal <code>~ 0</code>). For newdata with sites/species not present in the training data the random effects cannot be drawn, so those rows are fixed-effects-only regardless of re_form. On newdata, <code>predict()</code> rebuilds the linear predictor in R and can re-add only some of the model's random-effect tiers (the unit-level rr/diag terms, propto, and the spatial SPDE field). A fit carrying any other active tier gets a warning naming exactly what was omitted; newdata = NULL always returns the full conditional predictor.
se.fit	Logical, default FALSE. If TRUE, add an se.fit column: a conditional, fixed-effect-only, delta-method (Wald) standard error of est. "Conditional" means the random-effect contributions to the linear predictor are held fixed at their predicted (conditional-mode) values – their own uncertainty is not propagated, only the fixed-effect coefficients' <code>sdreport()</code> covariance is. This is a smaller quantity than a full marginal SE would be; it is also not a claim about coverage, which has not been measured for this quantity. <code>type = "link"</code> returns the SE of the linear predictor directly; <code>type = "response"</code> multiplies it by the local derivative of the per-row inverse link (the standard delta-method transform), so it approximates the SE of the inverse-link fitted value, not an exact one. Currently only supported for newdata = NULL (training rows), <code>non-multinomial()</code> fits, fits without an active <code>mi()</code> missing-covariate model, and fits with unit likelihood weights. A non-unit weighted objective has no certified prediction-standard-error route.
...	Unused.

Value

A data frame with the original row identifiers plus an est column on the requested link or response scale, and (if `se.fit = TRUE`) an se.fit column carrying the conditional delta-method standard error described above.

predict_cross_covariance

Predict pair-specific cross-lineage covariance

Description

`predict_cross_covariance()` combines a component-specific `Gamma_shape` block from `extract_Gamma()` with entries of the fitted dense kernel matrix. For a fixed cross-lineage kernel built with `make_cross_kernel()`, the off-diagonal `K` entries already include the supplied fixed `rho`, so the pair-specific covariance is `Gamma_shape * K[row_level, col_level]`. This helper therefore uses the shape-scale `Gamma` and does not multiply by `extract_Gamma(scale = "effect")`.

Fixed dense kernel `latent()` tiers, covered by the package's validation tests, store their aligned `K` matrices on the fit, and this helper returns point estimates for named species/lineage pairs. It does not estimate `rho`, produce intervals, calibrate null thresholds, or combine components into a universal total `Gamma`.

Usage

```
predict_cross_covariance(
  fit,
  level,
  row_levels = NULL,
  col_levels = NULL,
  row_traits,
  col_traits
)
```

Arguments

<code>fit</code>	A fitted <code>gllvmTMB_multi</code> object.
<code>level</code>	Character scalar naming the fitted <code>kernel_*()</code> tier.
<code>row_levels, col_levels</code>	Character vectors naming row and column levels in the fitted kernel matrix. If both are omitted for a kernel built by <code>make_cross_kernel()</code> , host and partner levels from the kernel metadata are used.
<code>row_traits, col_traits</code>	Character vectors of trait names defining the rows and columns of the component-specific <code>Gamma_shape</code> block.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
predict_cross_covariance(
  fit,
```

```

level = "phy",
row_levels = c("H1", "H2"),
col_levels = c("P1", "P2"),
row_traits = c("host_size", "host_defence"),
col_traits = c("partner_size", "partner_attack")
)

```

Value

A data frame with one row per level-pair and trait-pair. Columns include component, row_level, col_level, row_trait, col_trait, kernel_value, gamma_shape, covariance, rho, and kernel_includes_rho.

predict_missing	<i>Predict the masked (missing) response cells of a gllvmTMB fit</i>
-----------------	--

Description

For a model fitted with `missing = miss_control(response = "include")` (see [miss_control\(\)](#)), [gllvmTMB\(\)](#) keeps the rows / cells whose response was missing, masks them out of the likelihood, and predicts them from the fitted model. `predict_missing()` returns those masked response cells with their model-based predictions and the original-row / cell accounting from `fit$missing_data`.

Usage

```

predict_missing(
  object,
  type = c("link", "response"),
  se = FALSE,
  se_route = c("quad", "joint", "joint_load", "sim", "boot"),
  n_sim = 2000L,
  sim_seed = NULL,
  n_boot = 200L,
  boot_seed = NULL,
  boot_dgp = c("ml", "reml"),
  ...
)

```

Arguments

object	A fit returned by gllvmTMB() .
type	One of "link" (default; the linear predictor) or "response" (the inverse-link conditional mean; for ordinal_probit() traits, the expected category instead – see Details).
se	EXPERIMENTAL, internal-only. If TRUE, appends <code>se_confidence</code> (delta-method SE of the reconstructed mean) and <code>se_prediction</code> (<code>se_confidence</code> combined with the family noise variance) columns. Gaussian fits only in this slice; see Details. Default FALSE.

se_route	EXPERIMENTAL, internal-only. One of "quad" (default; R1-quad), "joint" (R1-joint), "joint_load" (R1-joint+loadings), "sim" (R2, simulation-based), or "boot" (R3, parametric bootstrap – see Details). Ignored unless se = TRUE.
n_sim	EXPERIMENTAL, internal-only. Number of Monte Carlo replicates drawn per masked cell when se_route = "sim". Ignored for every other route. Default 2000.
sim_seed	EXPERIMENTAL, internal-only. Integer RNG seed for se_route = "sim". Default NULL, which derives a fixed seed from the fitted parameter vector so repeated calls on the same fit reproduce identical draws; the caller's global RNG state is always saved and restored, never disturbed. Ignored for every other route.
n_boot	EXPERIMENTAL, internal-only. Number of full-refit bootstrap replicates when se_route = "boot". Ignored for every other route. Default 200.
boot_seed	EXPERIMENTAL, internal-only. Integer RNG seed for se_route = "boot". Default NULL, which derives a fixed seed from the fitted parameter vector (same convention as sim_seed); the caller's global RNG state is always saved and restored. Ignored for every other route.
boot_dgp	EXPERIMENTAL, internal-only. One of "ml" (default) or "rem1" – see Details. Ignored unless se_route = "boot".
...	Unused.

Details

Missing responses are *predicted / reconstructed* as fitted values, not latent covariates. The separate `imputed()` extractor returns modelled missing **predictors** from supported `mi()` fits. The point predictions here are the fitted linear predictor (type = "link") or its inverse-link response (type = "response"). Reconstruction standard errors and prediction intervals are not currently returned by default.

se = TRUE is **EXPERIMENTAL and INTERNAL-ONLY** (Design 119 Slice 1): register status heuristic_unvalidated – no repeated-sampling coverage evidence exists for se_confidence or se_prediction, and neither is an interval claim of any kind. It is currently implemented for gaussian fits only (other families abort). Five routes exist (se_route): "quad" (default) omits the b_fix/latent-score cross-covariance and any diag_B ("unique"/Psi) or within-unit (rr_W) random-effect contribution (sec.3 R1-quad) and OVER-covers se_confidence; "joint" computes the exact joint-precision variance for the b_fix and latent-score blocks (sec.3 R1-joint) but omits the loading-uncertainty block entirely and UNDER-covers; "joint" and "quad" BRACKET nominal coverage. "joint_load" (sec.7b, R1-joint+loadings) adds that third block; it is the best-calibrated delta-method route measured and still fails the gate by ~1.2 points at 95% (sec.7c). "sim" (sec.3 R2) is a Monte Carlo route: it draws the same gradient-relevant parameter subvector "joint_load" uses from its exact joint-precision marginal normal, forms n_sim EXACT (non-linearised) draws of $\eta^* = x'b^* + \lambda_t u_i^*$, and adds one gaussian family draw per replicate for y^* . Unlike the delta-method routes it makes NO normality assumption about the predictive distribution – se_confidence / se_prediction are the empirical sd of η^* / y^* , and it ALSO returns empirical-quantile columns (q_lo_conf/q_hi_conf, q_lo_pred/q_hi_pred at nominal 95%, and their *90 companions at nominal 90%) that a normal-quantile route cannot produce. It still holds every parameter at its estimate (a plug-in simulation), so it does not address hyperparameter uncertainty. "boot" (sec.3 R3, sec.7d) is a parametric bootstrap: it simulates a complete dataset

at the fitted parameters (fresh latent scores AND a fresh family draw for every cell), masks the same cells, and REFITS the model `n_boot` times. Because every replicate is a full refit, it is the only route that propagates parameter (and dispersion) uncertainty rather than holding it fixed at the plug-in estimate; it is also by far the most expensive route (`n_boot` model fits per call) and, in this slice, only supports gaussian fits with a single ordinary loadings-only `latent()` term (see Details in the source for the derivation and the scope guard). A refit that errors or fails to converge is dropped from the pivot sample and counted in `n_boot_ok`; if fewer than half the refits survive, `se_confidence/se_prediction/the quantile` columns are NA for that fit (with a warning) rather than reporting an interval built from a thin sample.

`boot_dgp` (sec.7e, following the wave-3 coverage diagnosis that "boot" with `boot_dgp = "ml"` under-covers at the same level as "joint_load") controls what parameters generate the `n_boot` complete-data worlds. "ml" (default, byte-identical to the pre-sec.7e route) simulates from the SAME ML fit being intervalised – so at small `n` its known downward variance/loading bias is "re-imported" into the bootstrap world, making the pivots systematically narrow. "reml" fits ONE auxiliary REML model on the same (masked) data/formula first and simulates from ITS parameters instead – REML's own known bias direction (larger, less biased at small `n`, converging to ML as `n` grows) targets exactly that gap. The quantity being pivoted (the point estimate the interval is centred on, and every inner refit) stays ML either way; only the generative world changes. Ignored unless `se_route = "boot"`.

Do not surface `se_confidence / se_prediction / the quantile` columns as calibrated uncertainty in any user-facing output until the Design 119 sec.4 coverage campaign clears a route and family for export.

For `ordinal_probit()` traits, `type = "response"` is the **expected category** $E[k] = \sum_k k \cdot P(\text{category } k | \eta, \tau)$, computed from the fitted cutpoints (Hadfield 2015 convention: $\tau_1 = 0$ fixed, $\tau_2, \dots, \tau_{K-1}$ estimated; see `extract_cutpoints()`) – not a probability. It is not an elementwise `pnorm(eta)`, which is not a category quantity once $K > 2$. `type = "link"` is unchanged: the probit-scale linear predictor.

Value

A data frame with one row per masked response cell, with columns: `original_row` (the supplied long-data row, the supplied wide-data row before `traits()` stacking, or – for a `multinomial()` fit – the pre-expansion row of the user's data that the $K-1$ category-contrast pseudo-row belongs to), `model_row` (the row index into the fitted long-format data / response), the unit / cluster / trait identifier columns, `est` (the prediction on the requested scale), and, when `se = TRUE`, `se_confidence` and `se_prediction` (see Details – EXPERIMENTAL, not a calibrated interval). When `se = TRUE` and `se_route` is "sim" or "boot", also `q_lo_conf/q_hi_conf`, `q_lo_pred/q_hi_pred` (empirical 2.5%/97.5% quantiles / bootstrap interval endpoints) and `q_lo_conf90/q_hi_conf90`, `q_lo_pred90/q_hi_pred90` (the nominal 90% companions). When `se_route = "boot"`, also `n_boot_ok` (the number of the `n_boot` refits that converged and entered the pivot sample). A complete-data fit (no masked cells) returns a zero-row data frame with the same columns.

See Also

`gllvmTMB()`, `miss_control()`, `predict.gllvmTMB_multi()`.

predictive_check	<i>Fitted-model predictive checks for a multivariate gllvmTMB fit</i>
------------------	---

Description

`predictive_check()` compares the observed stacked-trait response to draws from the fitted model. The frequentist semantics are explicit: these are fitted-model predictive checks, not Bayesian posterior predictive checks.

Usage

```
predictive_check(
  object,
  type = c("rq_qq", "rootogram", "stat_grouped", "dens_overlay"),
  nsim = NULL,
  ndraws = NULL,
  seed = NULL,
  trait = NULL,
  group = NULL,
  stat = c("mean", "median", "zero_fraction"),
  residual_type = c("randomized_quantile", "simulation_rank"),
  condition_on_RE = TRUE,
  max_count = NULL
)
```

Arguments

object	A <code>gllvmTMB_multi</code> fit.
type	Diagnostic plot type. "rq_qq" plots exact randomized-quantile residuals when available; "rootogram" compares observed count frequencies with fitted-model simulated count frequencies; "stat_grouped" compares grouped summary statistics; "dens_overlay" overlays observed and simulated densities and is mainly useful for continuous responses.
nsim, ndraws	Number of fitted-model draws. <code>ndraws</code> is accepted as a <code>bayesplot</code> / <code>brms</code> -style alias; supply only one.
seed	Optional RNG seed.
trait	Optional character vector of trait names to keep.
group	Row-metadata column used by "stat_grouped". Default is "trait".
stat	Grouped statistic for "stat_grouped".
residual_type	Residual type used by "rq_qq". Defaults to exact "randomized_quantile" residuals; "simulation_rank" is available as a simulation-based fallback.
condition_on_RE	Logical. Passed to <code>simulate.gllvmTMB_multi()</code> for simulation-based checks. The default TRUE checks the fitted response distribution conditional on fitted random-effect modes.

`max_count` Optional upper count shown separately in "rootogram". Counts larger than this value are pooled into a final ">max_count" bin. Default NULL chooses a bounded automatic display range and pools larger counts so that a single extreme simulated draw cannot create a very wide rootogram.

Details

The returned object is a ggplot. Its plotted data, fit-health table from `check_gllvmTMB()`, and `fit$fit_health` snapshot are also stored in `attr(plot, "gllvmTMB_diagnostic")` so the figure can be audited without reverse-engineering ggplot layers.

Scope: fitted-model predictive plots and residual Q-Q/rootogram helpers for `gllvmTMB_multi` fits, with exact randomized-quantile residuals for Gaussian, binomial, Poisson, lognormal, Gamma, NB2, Beta, betabinomial, Student-t, zero-truncated Poisson, zero-truncated NB2, NB1, and ordinal-probit rows, plus a simulation-rank fallback. These are diagnostic displays, not formal uniformity, dispersion, interval-calibration, or Bayesian posterior-predictive tests. Exact residual support for tweedie, the delta/hurdle families, and multinomial is not implemented (see `residuals.gllvmTMB_multi()` for why); the simulation-rank fallback's own family coverage is separately limited and is not audited here.

Value

A ggplot object with diagnostic metadata attached in `attr(plot, "gllvmTMB_diagnostic")`.

Examples

```
set.seed(1)
n <- 24
df <- data.frame(
  unit = factor(rep(seq_len(n), each = 2)),
  trait = factor(rep(c("a", "b"), n)),
  value = rpois(2 * n, lambda = 2)
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | unit, d = 1),
  data = df,
  trait = "trait",
  unit = "unit",
  family = poisson()
)
predictive_check(fit, type = "rq_qq", seed = 1)
predictive_check(fit, type = "rootogram", ndraws = 20, seed = 1)
```

```
print.gllvmTMB_Sigma_phy_slope
```

Print an augmented latent-slope Sigma extraction

Description

Shared by the augmented phylo_latent (level = "phy_slope") and augmented spatial_latent (level = "spde_slope") per-LHS-column cross-trait Sigma extractions; the object's header field selects the printed keyword.

Usage

```
## S3 method for class 'gllvmTMB_Sigma_phy_slope'
print(x, digits = 3, ...)
```

Arguments

x	A gllvmTMB_Sigma_phy_slope object from extract_Sigma() with level = "phy_slope" or level = "spde_slope".
digits	Number of significant digits.
...	Ignored.

Value

x, invisibly.

profile_ci_total_variance
<i>Profile-likelihood CI for per-trait total variance</i>

Description

Genuine chi-square_1 profile intervals (via fix-and-refit on log V_t) for the per-trait total variance $V_t = (\text{Lambda Lambda}') [t, t] + \text{psi}[t]$, the diagonal of the requested covariance tier. This is the same functional [extract_Sigma\(\)](#) returns on its diagonal, and the same one [bootstrap_Sigma\(\)](#) resamples.

Usage

```
profile_ci_total_variance(
  fit,
  tier = c("unit", "unit_obs", "phy", "B", "W"),
  trait_idx = NULL,
  level = 0.95
)
```

Arguments

fit	A fit returned by gllvmTMB() .
tier	Covariance tier: "unit", "unit_obs", "phy", or the soft-deprecated legacy aliases "B" / "W". Default "unit".
trait_idx	Integer indices of traits, or NULL for all.
level	Confidence level. Default 0.95. Only 0.95 carries coverage evidence.

Value

A data frame with one row per trait and columns:

trait, tier Trait name and the canonical tier name.

estimate Point estimate of V_t .

lower, upper Two-sided profile bounds, NA when the profile could not be computed.

method Always "profile".

interval_status Claim-boundary marker: "certified-0.94" when the row falls inside the certified regime above, "route-only" for a computed but uncertified interval, and "none" for a point-only row with no interval. "certified-0.94" marks *regime membership*; it does not certify the individual interval.

What the coverage evidence does and does not cover

One regime of this function has measured frequentist coverage. Under simulation from a known Gaussian data-generating process, using an unpenalised native-Laplace fit with $n_{\text{units}} \geq 150$ and $d \leq 2$, the two-sided intervals met a **pre-registered** ≥ 0.94 **gate** (0.9467 in both cells, 20,000 replicates each, computed among converged fits only). **That gate is 0.94, not nominal 95%** – both cells sit roughly 3.3 cluster standard errors below 0.95, so do not describe these as 95% intervals. Coverage is a *marginal average* over the simulated V_t distribution and does **not** hold in the smallest- V_t ventile (0.926 at $d = 1$). The interval is not equal-tailed, so **one-sided use is invalid**. The evidence is one simulated DGP, not a claim about any real dataset.

Everything else the arguments admit is an uncertified computed route: every AGHQ fit, any loading-ridge fit, every non-Gaussian family, every tier other than "unit", $d > 2$, $d = 0$, $n_{\text{units}} < 150$, any level other than 0.95, and the off-diagonal and psi targets (the psi target was measured on the same run and **failed**). Rather than refuse those calls – they are legitimately useful for exploration – the returned interval_status column marks each row.

See Also

[extract_Sigma\(\)](#) for the point estimate, [bootstrap_Sigma\(\)](#) for the resampling route to the same estimand.

profile_cross_rho	<i>Profile fixed cross-lineage rho values</i>
-------------------	---

Description

profile_cross_rho() formalises a fixed-kernel sensitivity workflow for cross-lineage kernels. It rebuilds K_{star} with [make_cross_kernel\(\)](#) over a user-supplied rho grid, calls a caller-supplied `refit(K, rho, ...)` function for each grid value, and returns a tidy likelihood table.

This is a fixed-kernel profile workflow for comparing defended rho values. Note its limits: rho is not a TMB parameter, this helper does not estimate rho, and it does not produce confidence intervals or null calibration.

Usage

```
profile_cross_rho(
  A_H,
  A_P,
  W,
  rho,
  refit,
  metrics = NULL,
  eps = 1e-08,
  keep_fits = FALSE,
  ...
)
```

Arguments

A_H, A_P, W	Inputs passed to make_cross_kernel() .
rho	Numeric vector of fixed rho values to evaluate.
refit	Function called as <code>refit(K = K_rho, rho = rho_i, ...)</code> . It should return a fitted object with a <code>logLik()</code> method.
metrics	Optional function called as <code>metrics(fit = fit_i, K = K_rho, rho = rho_i)</code> . It may return a named list or one-row data frame of additional scalar summaries, such as a Gamma correlation or a component norm.
eps	Positive numerical floor passed to make_cross_kernel() .
keep_fits	Logical; if TRUE, fitted objects are attached as a "fits" attribute. Default FALSE keeps the returned object light.
...	Additional arguments passed to <code>refit</code> .

Value

A data frame of class `gllvmTMB_cross_rho_profile` with columns `rho`, `logLik`, `relative_logLik`, `delta_deviance`, `is_best`, `convergence`, `pd_hessian`, `status`, and `error`, plus any metric columns returned by `metrics`.

Examples

```
A_H <- diag(2)
A_P <- diag(2)
rownames(A_H) <- colnames(A_H) <- c("H1", "H2")
rownames(A_P) <- colnames(A_P) <- c("P1", "P2")
W <- matrix(c(1, 0.2, 0.2, 1), 2, 2)
dimnames(W) <- list(rownames(A_H), rownames(A_P))
dat <- data.frame(y = c(1, 2, 3, 4), x = c(0, 1, 0, 1))
profile_cross_rho(
  A_H, A_P, W,
  rho = c(0, 0.25),
  refit = function(K, rho) stats::lm(y ~ x, data = dat)
)
```

profile_cross_rho_ci *Profile sensitivity interval for the cross-lineage rho*

Description

Turns a `profile_cross_rho()` grid into a profile-likelihood sensitivity interval for the fixed cross-lineage correlation rho. The interval is the set of rho values whose deviance excess $2 * (\logLik_{\max} - \logLik)$ stays below `qchisq(level, df = 1)`; the bounds are located by linear interpolation of the profiled delta_deviance curve between the two grid points that bracket each crossing. Use a rho grid dense enough to bracket the crossings (a coarse 3-4 point sensitivity grid is usually too sparse for a calibrated interval). When the profiled curve does not rise above the threshold on one side within the supplied grid, that bound is reported as the grid edge and flagged unbounded (`lower_bounded / upper_bounded FALSE`) – widen the grid.

This is screening-grade profile/sensitivity interval evidence on a *fixed*-rho refit grid, not in-engine rho estimation. Direct validation of this route remains partial, and its coverage has **not** been calibrated, so do not report it as a validated confidence interval.

Usage

```
profile_cross_rho_ci(profile, level = 0.95)
```

Arguments

profile	A <code>gllvmTMB_cross_rho_profile</code> data frame from <code>profile_cross_rho()</code> (needs the rho and delta_deviance columns).
level	Confidence level in (0, 1); default 0.95.

Value

A list with estimate (the grid rho with the highest logLik), lower, upper, level, lower_bounded, upper_bounded, and the deviance threshold. Bounds are clamped to the rho domain [-1, 1].

See Also

`profile_cross_rho()`, `make_cross_kernel()`

profile_phylo_signal *Profile-likelihood curve(s) for per-trait phylogenetic signal*

Description

For each requested trait, sweep over a grid of candidate values of the 2-component phylogenetic signal $H_t^2 = \sigma_{\text{phy},t}^2 / (\sigma_{\text{phy},t}^2 + \sigma_{\text{non},t}^2)$, refit the model under each constraint, and return the LR curve. Currently supports only the simple 2-component case (phylo_diag + species-level diag); 3-component decompositions error with a note.

Usage

```
profile_phylo_signal(
  fit,
  trait_idx = NULL,
  n_grid = 21L,
  grid_extent = 4,
  conf_level = 0.95
)
```

Arguments

fit	A fit returned by gllvmTMB() .
trait_idx	Integer vector of trait indices, or NULL for all.
n_grid	Integer; number of grid points per trait. Default 21.
grid_extent	Numeric; total grid width as a multiple of a robust scale (default 4).
conf_level	Confidence level. Default 0.95.

Details

Parallel CURVE endpoint to [profile_ci_phylo_signal\(\)](#).

Value

A data.frame of class profile_phylo_signal, profile_derived, data.frame.

See Also

[profile_ci_phylo_signal\(\)](#), [plot.profile_derived\(\)](#).

profile_targets	<i>List profile-ready confidence-interval targets for a fitted model</i>
-----------------	--

Description

`profile_targets()` is an advanced inventory for users who need to diagnose or script profile confidence intervals. Most users should start with [confint.gllvmTMB_multi\(\)](#) or a report-ready extractor such as [extract_correlations\(\)](#). This helper tells you which parm labels can be profiled directly.

Usage

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

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>ready_only</code>	If TRUE, return only rows with <code>profile_ready = TRUE</code> . Default FALSE returns the full inventory.

Details

The returned tidy data frame has one row per profile target and records whether the target is profile-ready and, if not, why. Direct targets correspond to a single TMB parameter element (for example `b_fix[1]`, `sigma_eps`, or `sd_B[2]`) and can be passed to `confint(fit, parm = ..., method = "profile")`. Nonlinear derived targets are excluded because their former penalty-profile routes are withheld.

Scope: the controlled vocabulary and direct-profile routing is covered for fitted models with retained TMB objects. Broader target-explicit profile calibration remains future work.

The output uses a controlled vocabulary for direct target classes, transformations, and readiness notes.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
fit <- gllvmTMB(value ~ 0 + trait + latent(0 + trait | site, d = 1),
               data = sim$data,
               trait = "trait",
               unit = "site")
targets <- profile_targets(fit)
head(targets)
profile_targets(fit, ready_only = TRUE)
```

Value

A data frame with one row per target and the following columns:

`parm` User-facing target label, e.g. `b_fix[1]`, `sigma_eps`, or `sd_B[2]`.

`target_class` One of `fixed_effect`, `variance`, `dispersion`, `loading_packed`, `threshold`, or `scaling`.

`tmb_parameter` TMB-side parameter name (matches `names(fit$opt$par)`).

`index` 1-based index within the TMB parameter vector. NA for scalars.

`estimate` Point estimate on the natural scale (e.g. `sigma_eps = exp(log_sigma_eps)`).

`link_estimate` Point estimate on the optimisation scale (the TMB internal scale).

`scale` Either "natural" (after applying the transformation) or "link" (for raw packed entries).

`transformation` One of `linear_predictor`, `exp`, `logit`, `logit_p_tweedie`, `lambda_packed`, or `ordinal_threshold`.

`target_type` Always "direct" for currently returned rows.

`profile_ready` TRUE iff the target can be passed directly to `confint(fit, parm = ..., method = "profile")`.

`profile_note` One of `ready`, `tmb_object_required`, `missing_tmb_parameter`, or `latent_rotation_ambiguous`.

See Also

`confint.gllvmTMB_multi()` for the routing of `method = c("wald", "profile")`, `tmbprofile_wrapper()` for direct-parameter profiling. For nonlinear summaries, consult the relevant extractor; a method token does not imply that a profile interval is currently available.

residuals.gllvmTMB_multi

Diagnostic residuals for a multivariate gllvmTMB fit

Description

Returns row-wise residual diagnostics for a fitted `gllvmTMB_multi` model. `type = "randomized_quantile"` uses exact family CDFs for Gaussian, binomial, Poisson, lognormal, Gamma, NB2, Beta, betabinomial, Student-t, zero-truncated Poisson, zero-truncated NB2, NB1, and ordinal-probit rows. `type = "simulation_rank"` uses fitted-model simulations and is available as a fallback for checking the same row contract when exact family-CDF plumbing is not implemented for a family; its own family coverage (what `simulate.gllvmTMB_multi()` can draw from) is tracked separately and is not extended here.

Usage

```
## S3 method for class 'gllvmTMB_multi'
residuals(
  object,
  type = c("randomized_quantile", "simulation_rank"),
  scale = c("normal", "uniform"),
  nsim = NULL,
  ndraws = NULL,
  seed = NULL,
  trait = NULL,
  condition_on_RE = TRUE,
  ...
)
```

Arguments

<code>object</code>	A <code>gllvmTMB_multi</code> fit.
<code>type</code>	"randomized_quantile" for exact family-CDF randomized quantile residuals where implemented, or "simulation_rank" for simulation-rank residuals from fitted-model draws.
<code>scale</code>	"normal" returns normal-quantile residuals; "uniform" returns the randomized PIT value.
<code>nsim, ndraws</code>	Number of fitted-model draws for <code>type = "simulation_rank"</code> . Ignored by exact randomized-quantile residuals.
<code>seed</code>	Optional RNG seed.

trait	Optional character vector of trait names to keep.
condition_on_RE	Logical. Passed to <code>simulate.gllvmTMB_multi()</code> for simulation-rank residuals.
...	Currently unused.

Details

Rows are retained even when a residual cannot be computed. Inspect the `status` column before treating residuals as complete.

Scope: exact family-CDF randomized-quantile residuals for Gaussian, binomial, Poisson, lognormal, Gamma, NB2, Beta, betabinomial, Student-t, zero-truncated Poisson, zero-truncated NB2, NB1, and ordinal-probit rows, plus simulation-rank residuals from fitted-model draws. Tweedie, the delta/hurdle families (`delta_lognormal`, `delta_gamma`), and multinomial are deliberately not implemented: tweedie has no closed-form CDF without a new dependency, the delta/hurdle families mix a point mass at zero with a continuous part and need an explicit design decision for splitting the point mass, and multinomial's categories are unordered so a randomized-quantile residual is undefined without inventing an ordering. Unsupported families are retained with row status rather than promoted to exact residual claims. Formal residual tests (beyond the recovery checks in `tests/testthat/test-exact-rq-residuals-families.R`) remain later validation work.

The returned data frame also carries `attr(x, "gllvmTMB_diagnostic")` with `check_gllvmTMB()` output and the fitted object's `fit_health` snapshot.

Value

A data frame with row metadata (`.row`, `trait`, `family_id`, `family`, `link_id`), observed, randomized PIT value `u`, `residual`, `status`, `scale`, and method metadata. The attribute `method` records the residual engine.

Examples

```
set.seed(2)
n <- 24
df <- data.frame(
  unit = factor(rep(seq_len(n), each = 2)),
  trait = factor(rep(c("a", "b"), n)),
  value = rpois(2 * n, lambda = 2)
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | unit, d = 1),
  data = df,
  trait = "trait",
  unit = "unit",
  family = poisson()
)
residuals(fit, type = "randomized_quantile", seed = 1)
```

ridge_path

*Sweep the loading-ridge penalty and report a stability path***Description**

ridge_path() refits the same model across a grid of loading-ridge penalty scales (tau, i.e. gllvmTMBcontrol(loading_ride = tau)) and reports, for every trait, how the largest loading and its communality move as the penalty weakens (tau growing toward Inf, i.e. plain maximum likelihood with no ridge).

Usage

```
ridge_path(
  formula,
  data,
  family,
  tau = c(0.5, 1, 2, 4, 8, Inf),
  unit = "site",
  trait = "trait",
  weights = NULL,
  missing = miss_control(),
  control = gllvmTMBcontrol(),
  ...
)
```

Arguments

formula, data, family, unit, trait, weights, missing	Passed to gllvmTMB() at every grid point, exactly as you would call it directly. unit defaults to "site", matching gllvmTMB() 's own default.
tau	Numeric vector of ridge scales to sweep. Order does not matter: the function sorts internally, both for refitting and for the printed classification. Every entry must be a positive number; Inf is allowed and requests plain ML. Default c(0.5, 1, 2, 4, 8, Inf).
control	A gllvmTMBcontrol() object providing every OTHER setting (n_init, se, warn_runaway, ...). Its own loading_ridge / aghq_ridge value is overridden at every grid point and does not need to be set. The estimator is always "ml": loading_ridge cannot be combined with estimator = "mspl" (see gllvmTMB()).
...	Passed to gllvmTMB() at every grid point.

Details

This is a SENSITIVITY diagnostic, never an identification certificate. A ridge that stabilises a fit does not prove the underlying model is identified; it only prevents the optimiser from drifting to an unbounded point. The path lets you tell two situations apart that a single fit cannot: a trait whose loading and communality settle down as the penalty weakens is behaving like an ordinary, data-determined estimate; a trait that keeps moving toward the boundary as the penalty weakens is one whose finite estimate under any single tau is being determined by the penalty, not by the data.

Every grid point is a COLD refit: each τ starts from the model's own default initial values, not from the previous grid point's solution. This trades some speed for comparability across τ – a warm-started path could stay near one local optimum across the whole grid and never reveal that a later τ has a different, better optimum, which would understate exactly the instability this function exists to detect.

`print()` classifies each trait using a SIGNED statistic – a shrinking or flat loading is always "interior", regardless of magnitude; only a substantial POSITIVE move counts as "penalty-determined". Two comparisons are possible, chosen automatically per trait:

A converged $\tau = \text{Inf}$ point is available The largest finite- τ loading is compared directly to the Inf loading (the plain signed relative change). A blow-up there – exactly what the paragraph above calls "moving toward boundary" – is compared against `boundary_rel_thresh` (default 0.10, i.e. a >10% increase from the last finite τ to Inf). This comparison is never skipped when a converged Inf point exists: it is the diagnostic the previous paragraph promises.

No converged $\tau = \text{Inf}$ point is available The last two finite- τ grid points are compared using the log-log slope (elasticity) $d(\log \text{ loading}) / d(\log \tau)$, not the raw relative change. The raw relative change decays like $1/\tau$ for a fixed absolute grid spacing even when a trait is genuinely still moving – a $\tau = c(2000, 2100)$ pair (a 1.05x ratio) gives a still-growing trait only a 5% raw change, an artifact of the grid geometry rather than the trait's behaviour. The elasticity is invariant to that spacing (for loading $\sim \tau^k$ it recovers k exactly at any two grid points) and is compared against `elasticity_thresh` (default 0.10).

Both are heuristics, not formal tests; see `print.gllvmTMB_ridge_path`'s arguments to adjust either threshold.

$\tau = \text{Inf}$ requests a plain maximum-likelihood refit with no ridge at all. That refit can legitimately fail to converge, or converge to an extreme loading, for exactly the traits this function exists to flag – that failure or extremity is itself part of the diagnostic evidence, not a bug to work around. `fit_error` records the message when a grid point's refit errors outright, and the row's other columns are NA for that point; the classifier then falls back to the finite- τ elasticity comparison, since a failed Inf refit contributes no loading to compare.

Value

A data frame of class `c("gllvmTMB_ridge_path", "data.frame")` with one row per $\tau \times$ trait combination:

`tau` The ridge scale for this row's refit.

`trait` Trait name.

`max_loading` The largest `|loading|` for this trait across the model's reduced-rank blocks at this τ .

`communality` This trait's between-unit communality at this τ (from `extract_communality()`, `level = "unit"`); NA if unavailable.

`logLik_at_map` The UNPENALISED log-likelihood evaluated at the penalised (MAP) point – not a maximum for any finite τ , so it is not comparable across τ as a model-selection criterion. At $\tau = \text{Inf}$ this is the ordinary maximum log-likelihood.

`convergence` The optimiser's convergence code (0 = converged); NA if the refit errored.

`fit_error` The error message if this grid point's refit failed outright; NA otherwise.

See Also

[screen_gllvmTMB\(\)](#) for the pre-fit response screen this complements; the loading ridge itself is documented at [gllvmTMBcontrol\(\)](#)'s `loading_ridge` argument.

Examples

```
n <- 40
df <- data.frame(
  unit = factor(seq_len(n)),
  a = rbinom(n, 1, 0.5),
  b = rbinom(n, 1, 0.5)
)
path <- ridge_path(
  traits(a, b) ~ 1 + latent(1 | unit, d = 1),
  data = df,
  family = binomial(),
  unit = "unit",
  tau = c(1, 2, 4, Inf)
)
path
```

rotate_loadings

Rotate the loadings of a fitted multivariate model

Description

Applies a varimax or promax rotation to the loading matrix Λ from a fit returned by [gllvmTMB\(\)](#). Use `level = "unit"` for the between-unit reduced-rank component and `level = "unit_obs"` for the within-unit component. The latent scores are rotated by the complementary transform so the linear predictor, fitted log-likelihood, and implied covariance are unchanged.

Usage

```
rotate_loadings(
  fit,
  level = "unit",
  method = c("varimax", "promax", "none"),
  order_axes = TRUE,
  sign_anchor = c("auto", "none"),
  anchor_traits = NULL
)
```

Arguments

<code>fit</code>	A fitted multivariate model returned by gllvmTMB() .
<code>level</code>	"unit" (between-unit) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.

method	One of "varimax", "promax", or "none".
order_axes	Logical. When TRUE (default for rotated output), reorder rotated axes by decreasing shared variance $\text{colSums}(\Lambda^2)$. Ignored when method = "none".
sign_anchor	One of "auto" or "none". "auto" (default for rotated output) flips each rotated axis so its anchor trait has a positive loading. Ignored when method = "none".
anchor_traits	Optional character vector of trait names used for sign anchoring. Supply one trait per axis after ordering. Axes without a supplied anchor use the trait with the largest absolute loading.

Details

Rotation is for interpretation of the loading columns, especially figures. It does not make the axes uniquely "right"; it chooses a readable orientation among mathematically equivalent orientations. For quantitative interpretation, start from the model-implied Σ , correlations, communality, and uniqueness, then use rotated axes for labels and plots.

A defensible plotting workflow is: fit the model, check convergence, reconstruct covariance summaries, then rotate loadings for visual interpretation. Rotate `level = "unit"` and `level = "unit_obs"` separately because they represent different covariance structures. `method = "varimax"` is the usual first choice for a simple loading pattern; use `method = "promax"` only when correlated latent axes are intended.

The rotation is applied to Λ on the *left* and to the latent scores on the *right* using the inverse transform, so $\Lambda_{\text{rot}} \mathbf{z}_{\text{rot}} = \Lambda \mathbf{z}$ is unchanged. For varimax, T is orthogonal so the rotation preserves the implied covariance $\Lambda \Lambda^T$; for promax, T is oblique and the columns of Λ_{rot} are no longer orthogonal.

Value

A list with rotated Λ ($n_{\text{traits}} \times d$), rotated scores (with rows = units or within-unit observations, columns = factors), and the rotation matrix T such that $\Lambda_{\text{rotated}} = \Lambda T$. The list also includes `axis_variance`, `axis_order`, `axis_sign`, and `anchor_traits` metadata after any ordering and sign anchoring.

Examples

```
set.seed(1)
sim <- simulate_site_trait(
  n_sites = 60, n_species = 12, n_traits = 4,
  Lambda_B = matrix(c(1.0, 0.7, -0.3, 0.5,
                      0.3, -0.5, 0.8, 0.2),
                    nrow = 4, ncol = 2),
  psi_B = c(0.3, 0.3, 0.3, 0.3),
  seed = 1
)
fit <- gllvmTMB(value ~ 0 + trait +
               latent(0 + trait | site, d = 2),
               data = sim$data,
               trait = "trait",
               unit = "site")
raw <- extract_ordination(fit, "unit")
```

```
rot <- rotate_loadings(fit, level = "unit", method = "varimax")
# raw$loadings - lower-triangular (hard to read)
# rot$Lambda - varimax-rotated (typically simpler structure)
```

sanity_multi

Print a quick convergence and parameter sanity report

Description

Use `sanity_multi()` as the fast first screen after fitting. It prints, and returns invisibly, pass / warn status flags covering optimiser convergence, max gradient component, Hessian definiteness, parameter standard errors, and identifiability of the `latent()` loadings. For report tables use `check_gllvmTMB()`; for a broader human-readable summary use `gllvmTMB_diagnose()`.

Usage

```
sanity_multi(object, gradient_thresh = 0.01, se_thresh = 100)
```

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>gradient_thresh</code>	Maximum allowed absolute gradient component. Default 0.01.
<code>se_thresh</code>	Threshold above which a parameter SE is flagged as suspiciously large. Default 100.

Details

Scope: fast numerical and loading-shape checks for fitted models. A PASS here does not prove interval calibration or latent-rank identifiability; use target-explicit known-DGP simulation studies for those heavier questions.

Value

Invisibly a list of diagnostic results.

scalar	<i>One shared trait variance: scalar(0 + trait g)</i>
--------	---

Description

Canonical no-prefix name for the **one-shared-variance** trait covariance: a single variance σ^2 shared by every trait with zero cross-trait covariance, giving $\Sigma_T = \sigma^2 \mathbf{I}_T$. It is the most parsimonious mode in the covariance grid.

Usage

```
scalar(formula)
```

Arguments

formula	0 + trait g style formula (LHS is the trait factor, typically 0 + trait; RHS is the grouping factor).
---------	---

Details

scalar() desugars byte-identically to indep(0 + trait | g, common = TRUE) – the two spellings fit the same model, tying all trait variances to one shared parameter. Use scalar() when you want to commit explicitly to one shared trait-scale variance; use indep() for a separate variance per trait, or dep() / latent() for cross-trait covariance.

The shared variance still couples grouping levels through the source (identity for the no-prefix row). The source-specific phylo_scalar(), animal_scalar(), and spatial_scalar() carry the same one-shared-variance meaning on their respective relationship operators.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
# One shared trait variance across all traits (identity cross-trait):
fit <- gllvmTMB(value ~ 0 + trait + scalar(0 + trait | site),
  data = df, trait = "trait", unit = "site")
```

```
# Equivalent longhand:
fit <- gllvmTMB(value ~ 0 + trait + indep(0 + trait | site, common = TRUE),
  data = df, trait = "trait", unit = "site")
```

Value

A formula marker; never evaluated.

See Also

[indep\(\)](#), [dep\(\)](#), [latent\(\)](#), [phylo_scalar\(\)](#), [extract_Sigma\(\)](#).

screen_control	<i>Control pre-fit response screening</i>
----------------	---

Description

`screen_control()` stores threshold settings used by `screen_gllvmTMB()`. The defaults are deliberately conservative and advisory: they flag data conditions to inspect before fitting, but they do not remove traits, choose a latent rank, or validate a model.

Usage

```
screen_control(
  module = "binomial",
  rare_warn_n = 10,
  rare_strong_n = 5,
  prevalence_warn = c(0.05, 0.95),
  prevalence_strong = c(0.02, 0.98),
  phi_warn = 0.9,
  phi_strong = 0.95,
  discordant_warn_n = 10,
  discordant_strong_n = 5,
  hamming_rate_warn = 0.01,
  separation = c("none", "fixed"),
  separation_tolerance = 1e-04
)
```

Arguments

<code>module</code>	Screening module. The first implemented module is "binomial".
<code>rare_warn_n</code> , <code>rare_strong_n</code>	Minority success/failure counts below these values trigger increasingly strong warnings.
<code>prevalence_warn</code> , <code>prevalence_strong</code>	Length-two numeric vectors giving lower and upper prevalence values for advisory imbalance notes.
<code>phi_warn</code> , <code>phi_strong</code>	Pairwise absolute-phi thresholds.
<code>discordant_warn_n</code> , <code>discordant_strong_n</code>	Pairwise discordant-count thresholds for near-duplicate binary traits.
<code>hamming_rate_warn</code>	Pairwise normalized Hamming-distance threshold.
<code>separation</code>	Optional fixed-design separation screen. The default "none" preserves the existing advisory screen and does not require <code>detectseparation</code> ; "fixed" returns one certificate per maximal coefficient-connected fixed-effect block.
<code>separation_tolerance</code>	Solver tolerance passed to <code>detectseparation</code> .

Value

A gllvmTMB_screen_control object.

screen_gllvmTMB	<i>Screen candidate responses before fitting a gllvmTMB model</i>
-----------------	---

Description

screen_gllvmTMB() is a formula-aware pre-fit screen. It summarises response and formula conditions before fitting a stacked-trait GLLVM. The screen is advisory: it does not fit the model, remove traits, choose a latent rank, prove identifiability, or guarantee convergence. Binary/binomial screening is implemented. Non-binary modules, optional comparator checks, and high-dimensional benchmarks are not currently part of this function.

Usage

```
screen_gllvmTMB(
  formula,
  data,
  family,
  unit = NULL,
  trait = "trait",
  weights = NULL,
  missing = miss_control(),
  control = screen_control(),
  Xcoef_fixed = NULL,
  known_groups = NULL
)
```

Arguments

formula	A gllvmTMB() formula. Wide data can use <code>traits()</code> on the left-hand side; long data should use a response column on the left-hand side plus <code>trait =</code> .
data	A data frame.
family	A response family. Version 1 screens single-family <code>binomial()</code> fits with <code>logit</code> , <code>probit</code> , or <code>cloglog</code> links.
unit	Optional unit/grouping column. If omitted, the first covariance-structure grouping column is used when available.
trait	Trait column for long data. Wide <code>traits()</code> calls create this column internally.
weights	Optional weights vector, with the same binomial trial-count semantics as <code>gllvmTMB()</code> for long-format binomial fits.
missing	Missing-data control; defaults to <code>miss_control()</code> .
control	A <code>screen_control()</code> object.

Xcoef_fixed	Optional named structural-zero fixed-effect constraints, normalized against the expanded fixed-effect columns as in <code>gllvmTMB()</code> . This argument is used by the opt-in fixed-design separation screen.
known_groups	Optional named list of character vectors of trait names, each with at least two distinct names (duplicated names within one group are an error). Each element declares a set of traits the user believes forms an exact one-hot (simplex) block or one or more exact pairwise containment (nesting) relations – for example the dummy columns of one categorical review-type or geographic-scope variable, or a broad-realm indicator together with one or more narrower nested realms. Each declared group is checked against the observed complete rows with an exact deterministic test: row sums for a one-hot block, or, for nesting, EVERY pairwise containment among the declared members (so it does not matter which order – or how many mutually incomparable narrower members – the group happens to be written in). When the relations found form a single total chain, the certificate reports that chain; when they form a partial order instead (for example one broad member containing two mutually incomparable narrower ones), every relation found is named and the group still fails as nesting. A group whose members are all constant on the complete rows is reported separately (not as nesting: a containment test is vacuously true against a constant and would not be genuine evidence); a member that is constant while other members of the same group are not is excluded from every pairwise comparison for the same reason. The one-hot test is not whole-group only: when the WHOLE declared group does not sum to 1 (for example a genuine one-hot block declared together with one unrelated trait), a bounded exhaustive search checks every subset of size 2 or more, smallest first, and reports each MINIMAL one-hot subset found as its own certificate – $A + B + C = 1$ is reported once even inside a larger declared group, never also as a redundant superset. This search is attempted only when the declared group has at most 12 members ($2^{12} - 13 = 4083$ subset checks, microseconds even for many rows; declared groups are small by construction – one categorical variable’s dummy columns or one nesting relation). A group larger than that bound is never silently reported PASS: the whole-group one-hot and pairwise nesting checks still run, but an explicit <code>known_group_subset_not_attempted</code> row records that the subset search itself was skipped for size, so a "no subset found" result can be told apart from "not looked for". A subset that sums to 1 only because every one of its members is constant is not reported (the same guard as the whole-group case). Each result is its own row in the <code>response_dependencies</code> table, independent of the automatic affine-rank screen below – except that a declared one-hot block, including a certified subset, also counts toward that screen’s unresolved affine-dependency count (a nesting relation never does: it is an inequality, not an exact affine relation among the columns). Screening is Bernoulli-only, matching the automatic screen.

Details

The first implemented module covers binomial traits. It distinguishes Bernoulli responses from multi-trial binomial responses, reports the relevant denominators, and flags constants, sparse minority outcomes, duplicate or near-duplicate binary traits, rank-deficient fixed-effect designs, and

grouping/rank conditions that should be inspected before interpretation. In systematic maps, a trait may be a content item or indicator. In a binary JSDM, a trait may be a species presence-absence response. Rare or constant species are flagged for inspection and possible sensitivity analysis, not automatically removed.

The screen also checks the response side for exact higher-order dependencies that the pairwise duplicate/complement check cannot see – most commonly a one-hot/simplex block, where several trait columns are really the dummy coding of one categorical variable and sum to exactly 1 on every complete row (a review-type, geographic-scope, or temporal-scope set is a typical source). It works by an affine-rank check on the augmented response matrix $M = \text{cbind}(1, Y)$ over complete Bernoulli rows (rows where every screened trait is observed), using base R's `qr()` with tolerance $\sqrt{.Machine\$double.eps} * \max(\dim(M))$ (the usual dimension-scaled machine-precision rule, so the check tightens automatically on a larger response matrix rather than using one fixed constant regardless of size). The `$design` table's `response_affine_rank` row reports the matrix's rank (value) and its column count (threshold, = `ncol(M)`); a rank deficiency – threshold – value – means at least one exact affine dependency exists among the traits. The `$response_dependencies` table then adds, where cheaply recoverable from the matrix's null space, a human-readable certificate naming the traits and the relation (for example "A + B + C = 1"). This is a best-effort report, not exhaustive minimal-subset discovery: with more than one simultaneous exact dependency the automatic search can fail to decompose the null space into individually clean certificates, in which case the remaining deficiency is reported without a certificate (type = "unresolved"), never silently dropped. Declaring known structure via `known_groups` sidesteps that limitation with an exact, deterministic, per-group check. An exact one-hot block is not itself a bug to silently filter – it is one categorical variable, and belongs in `multinomial()` or a deliberate reference-level coding, not in several unconstrained binary traits.

With `screen_control(separation = "fixed")`, the screen uses the optional `detectseparation` package to classify the observed fixed design. This includes finite known offsets, whose presence is recorded in the returned separation table. The offset does not change the fixed-design recession directions used by the certificate. This is diagnostic support only: `gllvmTMB()` currently rejects nonzero offsets for binomial responses.

This certificate covers fixed-effect geometry only. It does not establish finiteness of latent loadings or covariance parameters, validate the full marginal GLLVM likelihood, or select a penalized estimator.

Value

A `gllvmTMB_screen` object. Use `screen_table()` to extract report-ready tables.

References

- Albert A, Anderson JA (1984). On the existence of maximum likelihood estimates in logistic regression models. *Biometrika* 71:1–10. doi:10.1093/biomet/71.1.1.
- Peduzzi P, Concato J, Kemper E, Holford TR, Feinstein AR (1996). A simulation study of the number of events per variable in logistic regression analysis. *Journal of Clinical Epidemiology* 49:1373–1379. doi:10.1016/S0895-4356(96)00236-3.
- Vittinghoff E, McCulloch CE (2007). Relaxing the rule of ten events per variable in logistic and Cox regression. *American Journal of Epidemiology* 165:710–718. doi:10.1093/aje/kwk052.

Chalmers RP (2012). mirt: A multidimensional item response theory package for the R environment. *Journal of Statistical Software* 48(6):1–29. doi:10.18637/jss.v048.i06.

Kosmidis I, Schumacher D, Schwendinger F (2026). detectseparation: Detect and Check for Separation and Infinite Maximum Likelihood Estimates. R package version 0.4.0. doi:10.32614/CRAN.package.detectseparation.

Examples

```
df <- data.frame(
  study = factor(seq_len(12)),
  a = c(rep(1, 10), 0, 0),
  b = c(rep(1, 10), 0, 0),
  c = rep(c(0, 1), 6)
)
# The wrapper keeps the rendered example focused on the screen tables;
# inspect warnings in interactive work.
scr <- suppressWarnings(screen_gllvmTMB(
  traits(a, b, c) ~ 1 + latent(1 | study, d = 2),
  data = df,
  unit = "study",
  family = binomial()
))
screen_table(scr, "traits")
screen_table(scr, "recommendations")
```

screen_table	<i>Extract tables from a pre-fit screen</i>
--------------	---

Description

Extract tables from a pre-fit screen

Usage

```
screen_table(
  x,
  table = c("summary", "traits", "pairs", "response_dependencies", "units", "design",
    "separation", "recommendations", "settings")
)
```

Arguments

x	A gllvmTMB_screen object.
table	Table to extract.

Value

A data frame.

```
simulate.gllvmTMB_multi
```

Simulate new responses from a fitted gllvmTMB model

Description

Draws `nsim` new response vectors from a fitted model. By default (`condition_on_RE = FALSE`) the random effects are **redrawn** from the fitted covariance and the response is drawn from the fitted family — the unconditional simulation appropriate for a parametric bootstrap. Redraw is not implemented for every tier; a fit using an unhandled tier falls back to conditional simulation with a warning, and intervals derived from it are too narrow. Set `condition_on_RE = TRUE` for the older conditional behaviour, which reuses the fitted random-effect modes and only adds residual noise.

Usage

```
## S3 method for class 'gllvmTMB_multi'
simulate(
  object,
  nsim = 1,
  seed = NULL,
  newdata = NULL,
  condition_on_RE = FALSE,
  ...
)
```

Arguments

<code>object</code>	A fit returned by <code>gllvmTMB()</code> .
<code>nsim</code>	Number of replicate response vectors to draw. Default 1.
<code>seed</code>	Optional RNG seed.
<code>newdata</code>	Optional new data frame; if supplied, predictions are computed at <code>newdata</code> and noise is drawn around them. The <code>newdata</code> must contain enough columns to rebuild the fixed-effects design and any random-effect grouping that was active.
<code>condition_on_RE</code>	Logical (default <code>FALSE</code>). When <code>FALSE</code> (the default), random effects are redrawn from the fitted covariance — the unconditional simulation appropriate for parametric bootstrap. Redraw is currently implemented for the <code>rr_B</code>, <code>diag_B</code>, <code>rr_W</code>, <code>diag_W</code>, <code>propto</code>, <code>lv_B</code>, <code>phylo_rr</code>, and <code>diag_species</code> tiers. Not every tier is covered. A fit using any other active tier — notably the SPDE spatial tier (<code>spde</code>) and the diagonal phylogenetic tier (<code>phylo_diag</code>) — falls back to conditional simulation and emits a one-shot warning naming the unhandled tiers. Because conditional simulation reuses the fitted random-effect modes rather than redrawing them, it understates between-unit variability: intervals derived from it (for example via <code>bootstrap_Sigma()</code>) are too narrow and should not be read as calibrated. Treat the warning as a signal that simulate-based uncertainty is not trustworthy for that fit.

When TRUE, the existing fitted RE modes are reused (the older glmmTMB-style conditional simulation that only adds Gaussian noise on top of `fit$report$eta`). Forced to TRUE when newdata is supplied (RE modes for unseen levels cannot be redrawn).

... Currently unused.

Value

A matrix of dimension `n_obs` x `nsim` (or `nrow(newdata)` x `nsim` when newdata is supplied). **Per-row family coverage (#1079):** rows are drawn from their true fitted distribution for gaussian, binomial (at the row's actual trial count, not forced to Bernoulli), poisson, lognormal, Gamma, nbinom2, nbinom1, Beta, betabinomial, student-t, zero-truncated poisson, zero-truncated nbinom2, delta_lognormal, delta_gamma, ordinal_probit, and multinomial. The one exception is tweedie, which has no exact draw implemented; its rows (and any other unrecognised family) are returned as `NA_real_`, together with a warning naming the affected `family_id` values — never a Gaussian-on-link-scale substitute, and never silent. The newdata path (predictions on unseen data) still falls back to a Gaussian-on-link-scale draw for every family, with its own warning; family-aware newdata simulation is not yet implemented.

<code>simulate_site_trait</code>	<i>Simulate a functional-biogeography GLLVM dataset (<code>sites</code> × <code>species</code> × <code>traits</code>)</i>
----------------------------------	---

Description

Simulates a three-dimensional dataset from the Nakagawa et al. (in prep) functional-biogeography model:

$$y_{sit} = \alpha_t + x'_s \beta_t + r_{st} + u_{st} + e_{sit} + p_{it} + q_{it}$$

Usage

```
simulate_site_trait(
  n_sites = 50,
  n_species = 20,
  n_traits = 3,
  mean_species_per_site = 5,
  n_predictors = 2,
  alpha = NULL,
  beta = NULL,
  sigma2_eps = 0.5,
  Lambda_B = NULL,
  Lambda_W = NULL,
  psi_B = NULL,
  psi_W = NULL,
  sigma2_phy = NULL,
  sigma2_sp = NULL,
  Cphy = NULL,
```

```

    spatial_range = NULL,
    sigma2_spa = NULL,
    coords = NULL,
    seed = NULL
)

```

Arguments

n_sites	Integer; number of sites S .
n_species	Integer; number of species I in the regional pool.
n_traits	Integer; number of traits T .
mean_species_per_site	Average number of species observed per site (Poisson with this mean, truncated at $n_{species}$). Default 5.
n_predictors	Integer; number of site-level predictors. Default 2.
alpha	Optional length- n_{traits} vector of trait intercepts; random if NULL.
beta	Optional $n_{traits} \times n_{predictors}$ matrix of trait-specific slopes; random if NULL.
sigma2_eps	Residual variance (psi_W only term in fixed-effects-only simulator). Default 0.5.
Lambda_B, Lambda_W	Optional $n_{traits} \times d_B / n_{traits} \times d_W$ loading matrices for between-site / within-site reduced-rank components. Set to NULL (default) to omit.
psi_B, psi_W	Optional length- n_{traits} vectors of trait-specific variances at the global / local level. (Lowercase psi matches the factor-analysis convention – $\mathbf{\Psi}_B = \text{diag}(\text{psi}_B)$, $\mathbf{\Psi}_W = \text{diag}(\text{psi}_W)$; see decisions.md 2026-05-14 notation reversal.)
sigma2_phy, sigma2_sp	Optional length- n_{traits} vectors of phylogenetic and non-phylogenetic species variance. Cphy is required if sigma2_phy is supplied.
Cphy	Optional $n_{species} \times n_{species}$ phylogenetic correlation matrix. If supplied, used to draw p_{it} .
spatial_range, sigma2_spa	Optional spatial range and per-trait variance for an exponential spatial residual r_{st} . If both supplied and coords is NULL, sites are placed uniformly in $[0, 1]^2$.
coords	Optional $n_{sites} \times 2$ matrix of site coordinates (used when spatial_range is supplied).
seed	Optional RNG seed.

Details

This simulator is domain-specific: it produces a (site, species, trait) cube, which is the canonical functional-biogeography layout. The package as a whole works for any (unit, trait) stacked-trait GLLVM (unit may be site, individual, species, paper, ...); this particular simulator

targets the site $\times$ species $\times$ trait special case used in the methods paper. For simulations from a generic (unit, observation, trait) design without phylogenetic or spatial structure, use `simulate_unit_trait()`.

Each component can be turned on or off via the corresponding variance / loading argument. The default settings produce a fixed-effects-only dataset suitable for Stage-1 regression tests.

Value

A list with components:

data Long-format data frame with one row per (site, species, trait) observation: columns site, species, trait, value, site_species, predictors env_1, ..., env_n_predictors, plus lon and lat if coords were generated.

truth Named list of true parameter values (alpha, beta, Lambda_B, Lambda_W, psi_B, psi_W, sigma2_phy, sigma2_sp, sigma2_spa, spatial_range, sigma2_eps).

Cphy The phylogenetic correlation matrix used (or NULL).

coords Site coordinates used (or NULL).

See Also

`simulate_unit_trait()` for the generic (unit, observation, trait) simulator without phylogenetic or spatial machinery.

Examples

```
set.seed(1)
sim <- simulate_site_trait(n_sites = 30, n_species = 8, n_traits = 3,
                          mean_species_per_site = 4)
head(sim$data)
sim$truth$alpha
```

<code>simulate_unit_trait</code>	<i>Simulate a generic stacked-trait GLLVM dataset (units $\times$ observations $\times$ traits)</i>
----------------------------------	---

Description

Simulates a long-format dataset from the generic (unit, observation, trait) stacked-trait GLLVM that the package supports. A unit is any grouping over which a between-unit covariance structure applies (individual, site, species, paper, ...) and each unit carries `n_obs_per_unit` within-unit observations:

$$y_{uot} = \alpha_t + b_{ut} + w_{uot} + e_{uot}$$

where b_{ut} is the between-unit signal (reduced-rank Lambda_B plus diagonal psi_B), w_{uot} the within-unit observation-level signal (reduced-rank Lambda_W plus diagonal psi_W), and e_{uot} a row-level residual.

Usage

```
simulate_unit_trait(
  n_units = 50L,
  n_obs_per_unit = 3L,
  n_traits = 5L,
  alpha = NULL,
  Lambda_B = NULL,
  Lambda_W = NULL,
  psi_B = NULL,
  psi_W = NULL,
  sigma2_eps = 0.5,
  seed = NULL,
  ...
)
```

Arguments

<code>n_units</code>	Integer; number of units U .
<code>n_obs_per_unit</code>	Integer; number of within-unit observations O per unit.
<code>n_traits</code>	Integer; number of traits T .
<code>alpha</code>	Optional length- <code>n_traits</code> vector of trait intercepts; random if NULL.
<code>Lambda_B</code>	Optional <code>n_traits</code> x <code>d_B</code> between-unit loading matrix for the reduced-rank between-unit component. Set to NULL (default) to omit.
<code>Lambda_W</code>	Optional <code>n_traits</code> x <code>d_W</code> within-unit loading matrix for the reduced-rank observation-level component (matching a default <code>latent()</code> term, or an explicit <code>latent(unique = TRUE)</code> term, on the within-unit grouping). Set to NULL (default) to omit.
<code>psi_B</code>	Optional length- <code>n_traits</code> vector of trait-specific between-unit variances. (Lowercase <code>psi</code> matches the factor-analysis convention – <code>\boldsymbol{\Psi}_B = \text{diag}(\text{psi}_B)</code> ; see decisions.md 2026-05-14 notation reversal.)
<code>psi_W</code>	Optional length- <code>n_traits</code> vector of trait-specific within-unit (observation-level) variances – <code>\boldsymbol{\Psi}_W = \text{diag}(\text{psi}_W)</code> .
<code>sigma2_eps</code>	Row-level residual variance. Default 0.5.
<code>seed</code>	Optional RNG seed.
<code>...</code>	Reserved for future extensions; currently ignored.

Details

This is the generic sibling of [simulate_site_trait\(\)](#). Use this function when the design is an abstract (unit, observation, trait) layout with no phylogenetic or spatial structure. Use [simulate_site_trait\(\)](#) when you need the domain-specific functional-biogeography (site, species, trait) cube with phylogenetic species correlation and spatial site coordinates baked into the DGP.

Each component can be turned on or off via the corresponding variance / loading argument. The all-NULL default settings produce a fixed-effects-only dataset (trait intercepts plus row-level residual) suitable for fast regression fixtures.

Value

A list with components:

data Long-format data frame with one row per (unit, observation, trait): columns unit, observation, trait, value, and unit_observation – the row id used for a per-observation diagonal within-unit grouping (one level per unit-observation cell).

truth Named list of true parameter values: alpha, Lambda_B, Lambda_W, psi_B, psi_W, sigma2_eps.

See Also

[simulate_site_trait\(\)](#) for the domain-specific (site, species, trait) functional-biogeography simulator with phylogenetic and spatial machinery.

Examples

```
set.seed(1)
## Fixed-effects-only default (all structure NULL):
sim0 <- simulate_unit_trait(n_units = 20, n_obs_per_unit = 3, n_traits = 4)
head(sim0$data)
sim0$truth$alpha

## With a between-unit reduced-rank loading and diagonal within-unit term:
Lambda_B <- matrix(c(0.9, 0.7, -0.4, 0.2,
                    0.1, -0.2, 0.6, 0.8), nrow = 4, ncol = 2)
sim <- simulate_unit_trait(n_units = 40, n_obs_per_unit = 3, n_traits = 4,
                          Lambda_B = Lambda_B, psi_W = 0.3, seed = 1)
str(sim$truth)
```

slope_sd_ci

Wald confidence intervals on augmented random-slope standard deviations

Description

Per-trait confidence intervals on the standard deviation of an augmented random-slope term. All intervals are delta-method Wald intervals built on the log-SD scale so the bounds stay positive by construction. Three underlying routes are covered, dispatched automatically from the fit's structure:

- The ordinary augmented **diagonal** (Psi) route (theta_diag_B_slope, from latent(0 + trait + (0 + trait):x | unit, d = K) or unique(...)). theta_diag_B_slope is a genuine univariate log-SD (sd = exp(theta)), so the interval is read directly off sdreport()'s cov.fixed – no Jacobian.
- The **phylogenetic Cholesky** augmented-slope route (theta_dep_chol, from phylo_dep(1 + x1 + ... + xs | species) / phylo_indep(1 + x | species)). The marginal slope SD is a nonlinear function of several packed Cholesky entries ($\text{Var}(\text{slope}_t) = \text{L21}_t^2 + \exp(\text{diag_slope}_t)^2$ for a single slope), so its standard error comes from an ADREPORT()ed quantity in the C++ template (sd_b), letting sdreport() run the multivariate delta method against the authoritative packing.

- The **loadings-only** augmented random-slope route (theta_rr_B_slope with no diagonal companion, e.g. latent(..., unique = FALSE)). The marginal slope variance is a quadratic form in several loading entries (Lambda_B_slope %*% t(Lambda_B_slope)), again handled via an ADREPORT()ed quantity (sd_rr_B_slope).

Usage

```
slope_sd_ci(fit, level = 0.95, scale = c("sd", "variance"))
```

Arguments

fit	A multi-trait gllvmTMB() fit with at least one augmented random-slope structure covered above.
level	Confidence level. Defaults to 0.95.
scale	"sd" (default) returns the standard deviation scale; "variance" returns the variance scale (estimate^2, bounds exp(2 * (theta +/- z * se(theta)))). Both scales are computed from the identical theta / se_theta, so they agree by construction. total_sd is always reported on the SD scale regardless of scale; total_lower / total_upper follow scale like lower / upper.

Value

A data frame of class gllvmTMB_slope_ci, one row per slope SD (trait x slope term), with columns:

trait Trait factor level.

term The slope covariate name.

estimate, lower, upper On the requested scale. lower / upper are NA whenever status != "ok".

component "unique_psi" when the diagonal route's fit also carries a shared loadings block, so estimate is only part of the marginal slope SD; "total" otherwise, meaning estimate already is the total marginal slope SD.

total_sd Point estimate of the TOTAL marginal slope SD, on the SD scale regardless of scale. Equal to estimate when component == "total".

total_lower, total_upper Interval on total_sd, on the requested scale. NA whenever total_status != "ok" or the supporting ADREPORT is unavailable.

total_status Kill-switch status for the total_sd interval; see below. "unavailable" when no supporting ADREPORT exists (only possible for the diagonal route on a stale cached fit).

theta, se_theta The log-SD estimate and its standard error, for auditability. For the diagonal route these are the raw fixed-effect parameter and its cov.fixed SE (unchanged from slice 1); for the phylogenetic and loadings-only routes these are derived from an ADREPORTed natural-scale quantity via the delta method $se(\log(X)) = se(X) / X$.

method "wald_log_scale" for the diagonal route (direct parameter read), "wald_log_scale_adreport" for the phylogenetic and loadings-only routes (ADREPORT delta method).

interval_status Always "wald_uncalibrated".

status "ok", "no_pd_hessian", "se_nonfinite", "se_blowup", or "near_zero_relative" – see Kill-switch guard below.

scale The requested scale, "sd" or "variance".

The returned object also carries a hard-coded attribute `calibrated = FALSE` (never an argument; only a future coverage certificate can flip this in the source), and the requested level.

Scope boundary – what this interval IS and IS NOT

Every interval returned by this function is a **transformed Wald interval read off** `sdreport()`, the same inferential instrument as `loading_ci()` and `confint_inspect()`: it uses the asymptotic normal approximation and propagates it through a monotone log/exp transform. It is **not a calibrated coverage statement** – no repeated-sampling coverage campaign has been run for any of these estimands, and `interval_status = "wald_uncalibrated"` marks every row as such. The package's validation-debt ledger records this as tested-but-not- coverage-certified (see the ledger for the current status).

When the fit's random-slope term carries BOTH a diagonal Psi companion and a shared loadings component (`theta_rr_B_slope`, the default alongside `theta_diag_B_slope` for ordinary `latent()`), estimate for that route is the per-trait UNIQUE (Psi) component of slope variance only – it excludes the shared loadings contribution to the marginal slope variance, and can understate the total marginal slope SD substantially (measured up to ~45% on a recovery fixture). This restriction travels with the **data**, not only the print method: the component column reads "unique_psi" in that case ("total" otherwise – including for the phylogenetic and loadings-only routes, where the reported quantity already IS the total marginal slope SD). `total_sd` always reports the total marginal slope SD as a point estimate. `total_lower` / `total_upper` report a genuine interval on that total wherever the C++ template `ADREPORT()`s the combined expression directly (both routes as of this version); `total_status` names the same kill-switch statuses as `status` when that support is present, or "unavailable" when it is not (e.g. a fit refit under an older package version whose `ADREPORT` does not yet exist). A `cli::cli_warn()` fires on the call itself whenever the shared loadings block is present alongside the diagonal route, not only when the result is printed.

Kill-switch guard – point estimates always, intervals never from a fit that cannot support one

Applied identically to every route. `lower/upper` (and, where supported, `total_lower/total_upper`) are set to NA (with status explaining why and a `cli::cli_warn()`) whenever any of:

- `fit$sd_report$pdHess` is FALSE (`status = "no_pd_hessian"`, applies to every row);
- the log-SD estimate or its standard error is non-finite (`status = "se_nonfinite"`) – for the `ADREPORT`-based routes this also covers a non-positive natural-scale estimate, which cannot be logged;
- `se_theta > 10` on the log scale – essentially information-free (an SE of 10 on the log scale means the interval would span e^{40}); `status = "se_blowup"`;
- `sd_hat` is at most 1% of the largest `sd_hat` among this fit's OTHER slope coordinates on the SAME route (only evaluated when there is more than one to compare against); `status = "near_zero_relative"`, mirroring the package's relative-to-siblings convention for a collapsed variance component (`psi_rel_thresh` / `near_zero_psi_*`, `R/diagnose.R`).

The point estimate (estimate, theta) is always returned, following the house line that a non-PD Hessian disqualifies standard errors but not point estimates (`R/bootstrap-sigma.R`, `R/cv-internal.R`).

See Also

`loading_ci()` for the equivalent interval on Lambda entries; `extract_Sigma()` with `level = "unit_slope"` or `level = "phy"` for the full augmented covariance (point estimates only).

Examples

```
set.seed(1)
n_ind <- 30L
n_traits <- 2L
trait_levels <- paste0("t", seq_len(n_traits))
individuals <- paste0("id", seq_len(n_ind))
df <- expand.grid(
  individual = factor(individuals, levels = individuals),
  rep = 1:5,
  trait = factor(trait_levels, levels = trait_levels)
)
df$x <- stats::rnorm(nrow(df))
alpha <- c(0.2, -0.1)
beta <- c(0.3, -0.2)
psi_noise <- matrix(stats::rnorm(4L * n_ind, sd = 0.25), nrow = 4L)
eta <- numeric(nrow(df))
for (o in seq_len(nrow(df))) {
  tt <- as.integer(df$trait[o])
  ii <- as.integer(df$individual[o])
  base <- 2L * (tt - 1L)
  eta[o] <- alpha[tt] + beta[tt] * df$x[o] +
    psi_noise[base + 1L, ii] + psi_noise[base + 2L, ii] * df$x[o]
}
df$value <- eta + stats::rnorm(nrow(df), sd = 0.3)

options(gllvmTMB.quiet_grammar_notes = TRUE)
fit <- gllvmTMB(
  value ~ 0 + trait + (0 + trait):x +
    unique(0 + trait + (0 + trait):x | individual),
  data = df,
  trait = "trait",
  unit = "individual",
  control = gllvmTMBcontrol(se = TRUE, optimizer = "optim",
    optArgs = list(method = "BFGS"))
)
slope_sd_ci(fit)
slope_sd_ci(fit, level = 0.90, scale = "variance")
```

Description

`spatial()` is a unified entry point for the package's three spatial modes (`spatial_indep`, `spatial_latent`, `spatial_dep`). It accepts an lme4-bar formula on the first argument and dispatches to the appropriate canonical keyword based on the LHS shape and the optional `mode = ...` argument. The three current keywords stay first-class – `spatial()` is an additive alias matching the lme4 / brms / drmTMB convention.

Usage

```
spatial(
  formula,
  mesh = NULL,
  coords = NULL,
  mode = NULL,
  d = 1,
  unique = FALSE
)
```

Arguments

formula	An lme4-bar formula. The bar's RHS is the spatial coordinate placeholder (typically <code>coords</code> or <code>site</code>); the LHS determines the covariance structure (combined with <code>mode</code> , see Dispatch rules). For backward compatibility, <code>0 + trait</code> <code>coords</code> and <code>coords</code> <code>trait</code> (no <code>mode</code>) also work as deprecated aliases for the per-trait diagonal spatial field.
mesh	A <code>fmesher</code> mesh constructed via <code>make_mesh()</code> . Canonical. Required for the new dispatch path; optional for the legacy bare-formula form (where the mesh is passed at the top level of <code>gllvmTMB()</code>).
coords	Optional, retained as a hint of which coordinate columns correspond to the spatial grouping. The mesh actually used is whatever is passed to the <code>mesh</code> argument of <code>gllvmTMB()</code> or this function.
mode	One of "scalar" / "indep" / "latent" / "dep". Optional when LHS is 1 (defaults silently to "scalar"). Mandatory when LHS is <code>0 + trait</code> .
d	Latent rank for <code>mode = "latent"</code> . Default 1.
unique	Logical; forwarded to <code>spatial_latent()</code> when <code>mode = "latent"</code> . The default <code>FALSE</code> keeps the low-rank-only path.

Details

This is the spatial parallel of the `phylo()` mode-dispatch wrapper: same dispatch logic, different engine slots (SPDE Matern instead of Hadfield A^{-1}).

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
library(fmesher)
# Single shared spatial-field variance (= spatial_indep(..., common = TRUE))
fit_s <- gllvmTMB(value ~ 0 + trait + spatial(1 | site, mesh = mesh),
```

```

      data = df,
      trait = "trait",
      unit = "site")
# Per-trait independent fields on a shared mesh
fit_u <- gllvmTMB(value ~ 0 + trait +
  spatial(0 + trait | coords, mode = "indep",
    mesh = mesh),
  data = df,
  trait = "trait",
  unit = "site")
# Reduced-rank shared spatial latent factors (= spatial_latent)
fit_l <- gllvmTMB(value ~ 0 + trait +
  spatial(0 + trait | coords, mode = "latent",
    d = 2, mesh = mesh),
  data = df,
  trait = "trait",
  unit = "site")
# Backward-compat: legacy bare-formula form
fit_legacy <- gllvmTMB(value ~ 0 + trait + spatial(0 + trait | coords),
  data = df,
  trait = "trait",
  unit = "site",
  mesh = mesh)

```

Value

A formula marker; never evaluated.

Dispatch rules

Form	Mode default	Rewrites to
<code>spatial(1 \ site)</code>	"scalar"	<code>spatial_indep(0 + trait \ coords,</code>
<code>spatial(0 + trait \ coords, mode = "indep")</code>	(mandatory)	<code>spatial_indep(0 + trait \ coords)</code>
<code>spatial(0 + trait \ coords, mode = "latent", d = K)</code>	(mandatory)	<code>spatial_latent(0 + trait \ coords</code>
<code>spatial(0 + trait \ coords, mode = "dep")</code>	(mandatory)	<code>spatial_dep(0 + trait \ coords)</code>

When the LHS expands to a single column (1, intercept-only), mode is degenerate: it defaults silently to "scalar", and explicit mode = "scalar" is accepted (no warning). When the LHS is `0 + trait`, mode is **mandatory** – choosing between "indep" / "latent" / "dep" is a meaningful decision (per-trait marginal vs reduced-rank decomposition vs full unstructured) and the parser refuses to silently default.

Backward compatibility

Legacy bare-formula calls `spatial(0 + trait | coords)` and `spatial(coords | trait)` (no mode argument) continue to work as deprecated aliases for the per-trait diagonal spatial field (with an additional orientation-flip lifecycle warning for the pre-0.1.4 orientation). The legacy form rewrites to `spde()` internally, picking up the SPDE mesh from the top-level `mesh =` argument to `gllvmTMB()`.

Intercept-and-slope terms

The unified `spatial()` wrapper is limited to intercept-only trait covariance. For an intercept and slope, use an explicit structured mode such as `spatial_indep()`, `spatial_latent()`, or `spatial_dep()`; unsupported LHS forms fail with a message showing the accepted syntax.

Cross-package coexistence

`drmTMB` also exposes `spatial(1 | site, mesh = mesh)` with the same SPDE Matern internal path. Both packages share the same calling convention; user muscle memory transfers.

See Also

[spatial_indep\(\)](#), [spatial_latent\(\)](#), [spatial_dep\(\)](#), [phylo\(\)](#) (phylogenetic parallel), [spde\(\)](#) (deeper deprecated alias).

<code>spatial_dep</code>	<i>Full unstructured spatial trait covariance:</i> <code>spatial_dep(0 + trait coords)</code>
--------------------------	---

Description

Canonical name for the **full unstructured cross-trait spatial covariance** $\Sigma_{\text{spa}} \otimes \mathbf{Q}^{-1}$, with $T(T+1)/2$ free parameters (Σ_{spa} parameterised via Cholesky). Mathematically identical to `spatial_latent(0 + trait | coords, d = T)` standalone; the keyword choice is documentary.

Usage

```
spatial_dep(formula, coords = NULL, mesh = NULL)
```

Arguments

<code>formula</code>	<code>0 + trait coords</code> style formula (LHS is the trait factor <code>0 + trait</code> ; RHS is the <code>coords</code> placeholder symbol that points at the make_mesh() coordinate columns).
<code>coords</code>	Character; the column-name pair of spatial coordinates in data (e.g. <code>c("lon", "lat")</code>). Resolved by the parser when supplied as keyword argument; <code>NULL</code> when the orientation expresses the coordinates via the formula RHS.
<code>mesh</code>	An <code>fmesher</code> mesh object built via make_mesh() . It may be supplied here or through the top-level <code>mesh =</code> argument to glvmTMB() . The engine does not construct a mesh automatically from <code>coords</code> .

Details

Use `spatial_dep()` when you want an explicit full-unstructured cross-trait spatial covariance fit. Use `spatial_latent()` for the rank-reduced K-factor model. Use `spatial_indep()` for the marginal-only per-trait spatial fit.

Value

A formula marker; never evaluated.

Formula orientation

Same convention as the rest of the `spatial_*` keywords: the canonical orientation is `0 + trait | coords` (LHS = trait factor, RHS = the coords placeholder). `spatial_dep` is born with this orientation; the legacy `coords | trait` form is not accepted.

Mutual exclusion with `spatial_latent()` / `spatial_indep()`:

Combining `spatial_dep(0 + trait | coords)` with `spatial_latent` is **over-parameterised**; combining with `spatial_indep` is **redundant**. The parser raises `cli::cli_abort()` in any of these cases.

See Also

[spatial_latent\(\)](#), [spatial_indep\(\)](#), [dep\(\)](#), [phylo_dep\(\)](#), [extract_Sigma\(\)](#).

Examples

```
sim <- simulate_site_trait(
  n_sites = 30, n_species = 6, mean_species_per_site = 5,
  spatial_range = 0.4, sigma2_spa = rep(0.3, 6), seed = 1
)
mesh <- make_mesh(sim$data, c("lon", "lat"), cutoff = 0.1)
fit <- gllvmTMB(
  value ~ 0 + trait +
    spatial_dep(0 + trait | site, mesh = mesh),
  data = sim$data,
  trait = "trait",
  unit = "site"
)
```

spatial_indep	<i>Per-trait spatial marginal field:</i>	<code>spatial_indep(0 + trait coords)</code>
---------------	--	--

Description

Canonical name for **T per-trait spatial fields coupled by the SPDE precision matrix Q**; standalone = T univariate spatial fits stacked.

Usage

```
spatial_indep(formula, coords = NULL, mesh = NULL, common = FALSE)
```

Arguments

formula	$0 + \text{trait} \mid \text{coords}$ style formula (LHS is the trait factor $0 + \text{trait}$; RHS is the coords placeholder symbol that points at the <code>make_mesh()</code> coordinate columns).
coords	Character; the column-name pair of spatial coordinates in data (e.g. <code>c("lon", "lat")</code>). Resolved by the parser when supplied as keyword argument; NULL when the orientation expresses the coordinates via the formula RHS.
mesh	An <code>fmesher</code> mesh object built via <code>make_mesh()</code> . It may be supplied here or through the top-level <code>mesh =</code> argument to <code>gllvmTMB()</code> . The engine does not construct a mesh automatically from <code>coords</code> .
common	FALSE (default) for a separate spatial-field variance per trait; TRUE ties all traits to one shared spatial variance (intercept-only). <code>spatial_indep(0 + trait coords, common = TRUE)</code> is the canonical one-shared-variance spelling and fits the same model as the soft-deprecated <code>spatial_scalar()</code> .

Details

Each trait t gets its own variance τ_t^2 on a Matern $\nu = 1$ GMRF, with a shared range parameter κ .

Use `spatial_indep()` for an explicit marginal-only spatial fit (no cross-trait spatial decomposition). Use `spatial_latent()` for K shared spatial fields driving all T traits via a T x K loading matrix.

Value

A formula marker; never evaluated.

Formula orientation

Same convention as the rest of the `spatial_*` keywords: the canonical orientation is $0 + \text{trait} \mid \text{coords}$ (LHS = trait factor, RHS = the coords placeholder). Spatial keywords adopted this orientation at `gllvmTMB` 0.1.4; `spatial_indep` is born with it (no legacy `coords | trait` orientation is accepted).

Mutual exclusion with `spatial_latent()`:

Combining `spatial_indep(0 + trait | coords)` with `spatial_latent(0 + trait | coords, d = K)` is **over-parameterised** and the parser raises a `cli::cli_abort()`.

See Also

`spatial_latent()`, `indep()`, `phylo_indep()`, `extract_Sigma()`.

Examples

```
sim <- simulate_site_trait(
  n_sites = 20, n_species = 4, mean_species_per_site = 4,
  spatial_range = 0.4, sigma2_spa = rep(0.3, 4), seed = 1
)
mesh <- make_mesh(sim$data, c("lon", "lat"), cutoff = 0.1)
fit <- gllvmTMB(
  value ~ 0 + trait +
```

```

        spatial_indep(0 + trait | site, mesh = mesh),
data = sim$data,
trait = "trait",
unit = "site"
)

```

spatial_latent	<i>Reduced-rank spatial latent factors:</i> spatial_latent(0 + trait coords, d = K)
----------------	---

Description

Canonical name for the reduced-rank spatial random effect: K shared SPDE fields drive all T traits via a $T \times K$ loading matrix Λ_{spa} . The spatial analogue of [phylo_latent\(\)](#) and the third cell of the spatial column of the API grid (alongside [spatial_indep\(\)](#) and [spatial_dep\(\)](#)).

Usage

```
spatial_latent(formula, d = 1, unique = FALSE, coords = NULL, mesh = NULL)
```

Arguments

formula	$0 + \text{trait} \mid \text{coords}$ style formula (LHS is the trait factor $0 + \text{trait}$; RHS is the coords placeholder symbol).
d	Integer; number of spatial latent factors (rank K). Defaults to 1.
unique	Logical; include a per-trait unique spatial diagonal companion. FALSE preserves the old low-rank-only spatial_latent() path; TRUE fits the total spatial covariance $\Lambda\Lambda^\top + \Psi$.
coords	Character; the column-name pair of spatial coordinates in data (e.g. c("lon", "lat")). Resolved by the parser when supplied as keyword argument; NULL when the orientation expresses the coordinates via the formula RHS.
mesh	An fmesher mesh object built via make_mesh() . It may be supplied here or through the top-level mesh = argument to gllvmTMB() . The engine does not construct a mesh automatically from coords.

Details

Internally this rewrites to `spde(form, .spatial_latent = TRUE, d = K)` and toggles the TMB template's `spde_lv_k` switch to K . With `unique = TRUE`, the same SPDE tier also keeps the per-trait `omega_spde` fields active, giving $\Sigma_{\text{spa}} = \Lambda_{\text{spa}}\Lambda_{\text{spa}}^\top + \Psi_{\text{spa}}$. The default `unique = FALSE` preserves the older low-rank-only path. The earlier two-term companion spelling is accepted as compatibility syntax for the same total-covariance decomposition.

The C++ kernel then reads a packed lower-triangular Λ_{spde} and K shared spatial fields `omega_spde_lv` (each prior $N(\mathbf{0}, \mathbf{Q}^{-1})$ where $\mathbf{Q}$ is the SPDE precision built from the mesh), and accumulates $\eta_o \leftarrow \sum_k \Lambda_{\text{spa},tk} \omega_k(\mathbf{s}_o)$ per observation. Per-field τ is absorbed into Λ_{spa} for

identifiability, mirroring the `phylo_latent()` convention. Like all rr loadings, Λ_{spa} is identified only up to rotation; pin via `lambda_constraint = list(spde = ...)` or post-hoc rotate via `rotate_loadings()`.

Value

A formula marker; never evaluated.

Formula orientation

The canonical orientation is `0 + trait | coords` (parallel to `latent()` / `indep()` and glmmTMB's spatial keywords). The earlier orientation `coords | trait` is accepted as a deprecated alias and emits a one-shot lifecycle::deprecate_warn() per session (introduced at gllvmTMB 0.1.4).

See Also

`spatial_indep()`, `spatial_dep()`, `phylo_latent()`.

Examples

```
sim <- simulate_site_trait(
  n_sites = 30, n_species = 6, mean_species_per_site = 5,
  spatial_range = 0.4, sigma2_spa = rep(0.3, 6), seed = 1
)
mesh <- make_mesh(sim$data, c("lon", "lat"), cutoff = 0.1)
fit <- gllvmTMB(
  value ~ 0 + trait +
    spatial_latent(0 + trait | site, d = 2, mesh = mesh),
  data = sim$data,
  trait = "trait",
  unit = "site"
)
```

spatial_scalar

Deprecated alias for one shared spatial-field variance

Description

[Deprecated]

Usage

```
spatial_scalar(formula, coords = NULL, mesh = NULL)
```

Arguments

formula	$\emptyset + \text{trait} \mid \text{coords}$ style formula (LHS is the trait factor $\emptyset + \text{trait}$; RHS is the coords placeholder symbol).
coords	Character; the column-name pair of spatial coordinates in data (e.g. <code>c("lon", "lat")</code>). Resolved by the parser when supplied as keyword argument; NULL when the orientation expresses the coordinates via the formula RHS.
mesh	An fmesher mesh object built via <code>make_mesh()</code> . It may be supplied here or through the top-level <code>mesh =</code> argument to <code>gllvmTMB()</code> . The engine does not construct a mesh automatically from coords.

Details

`spatial_scalar()` is soft-deprecated compatibility syntax in `gllvmTMB` 0.2.0. Write `spatial_indep( $\emptyset + \text{trait} \mid \text{coords}$ , common = TRUE)` in new code. The two spellings fit the same SPDE / GMRF Matérn model with **one shared variance** τ^2 across all traits and a shared range parameter κ . Each trait carries its own independent field draw on the mesh, but every trait's field has the same marginal variance and the same correlation length. Implementation: the same SPDE engine as `spatial_indep()`, with the per-trait `log_tau_spde` parameters tied via TMB's map mechanism so they collapse to a single estimable scalar.

Use `spatial_indep(..., common = TRUE)` when domain knowledge (or parsimony) says all traits should share the same amount of spatial structure. Omit `common = TRUE` for separate trait-specific spatial variances, or use `spatial_latent()` for a reduced-rank cross-trait decomposition.

Value

A formula marker; never evaluated.

Formula orientation

The canonical orientation is $\emptyset + \text{trait} \mid \text{coords}$ (parallel to `latent()` / `indep()` and `glmmTMB`'s spatial keywords). The earlier orientation `coords | trait` is accepted as a deprecated alias and emits a one-shot `lifecycle::deprecate_warn()` per session (introduced at `gllvmTMB` 0.1.4).

See Also

`spatial_indep()`, `spatial_latent()`, `spde()` (deprecated alias).

Examples

```
sim <- simulate_site_trait(
  n_sites = 20, n_species = 4, mean_species_per_site = 4,
  spatial_range = 0.4, sigma2_spa = rep(0.3, 4), seed = 1
)
mesh <- make_mesh(sim$data, c("lon", "lat"), cutoff = 0.1)
fit <- gllvmTMB(
  value ~  $\emptyset + \text{trait} +$ 
    spatial_indep( $\emptyset + \text{trait} \mid \text{site}$ , mesh = mesh, common = TRUE),
  data = sim$data,
  trait = "trait",
```

```
    unit = "site"
  )
```

spatial_unique	<i>Per-trait independent spatial random fields:</i> <code>spatial_unique(0 + trait coords)</code>
----------------	---

Description

[Deprecated]

Usage

```
spatial_unique(formula, coords = NULL, mesh = NULL)
```

Arguments

formula	<code>0 + trait coords</code> style formula (LHS is the trait factor <code>0 + trait</code> ; RHS is the <code>coords</code> placeholder symbol that points at the <code>make_mesh()</code> coordinate columns).
coords	Character; the column-name pair of spatial coordinates in data (e.g. <code>c("lon", "lat")</code>). Resolved by the parser when supplied as keyword argument; <code>NULL</code> when the orientation expresses the coordinates via the formula RHS.
mesh	An <code>fmesher</code> mesh object built via <code>make_mesh()</code> . It may be supplied here or through the top-level <code>mesh =</code> argument to <code>gllvmTMB()</code> . The engine does not construct a mesh automatically from <code>coords</code> .

Details

`spatial_unique()` is soft-deprecated as compatibility syntax in `gllvmTMB` 0.2.0. Use `spatial_indep()` for standalone marginal diagonal spatial fields. Paired explicit-Psi use remains accepted; new decomposition code can use `spatial_latent()` with `unique = TRUE`.

Canonical name for the SPDE / GMRF Matern spatial random field with **one independent field per trait** (per-trait variance τ_t^2 , shared range parameter κ). This is the "unique-rank" cell of the spatial column of the API grid.

Formerly `spde(0 + trait | coords)` and `spatial(0 + trait | coords)` – same engine, new name. See `spde()` for the technical details on the Matern $\nu = 1$ kernel.

Compare to `spatial_indep()` with `common = TRUE` (one shared variance across traits) and `spatial_latent()` (K-dim spatial latent factors with a loading matrix).

Value

A formula marker; never evaluated.

Formula orientation

The canonical orientation is $0 + \text{trait} \mid \text{coords}$ (LHS = trait factor, RHS = the coords placeholder), parallel to `latent()`, `unique()`, and `glmmTMB`'s `gau(0 + trait | pos) / exp(0 + trait | pos)`. The earlier orientation `coords | trait` is accepted as a deprecated alias: it emits a one-shot lifecycle::deprecate_warn() per session (introduced at `gllvmTMB` 0.1.4) and is internally normalised to the canonical orientation, so existing fits remain byte-identical.

See Also

`spatial_indep()`, `spatial_latent()`, `spde()` (deprecated alias).

Examples

```
sim <- simulate_site_trait(
  n_sites = 20, n_species = 4, mean_species_per_site = 4,
  spatial_range = 0.4, sigma2_spa = rep(0.3, 4), seed = 1
)
mesh <- make_mesh(sim$data, c("lon", "lat"), cutoff = 0.1)
fit <- gllvmTMB(
  value ~ 0 + trait +
    spatial_indep(0 + trait | site, mesh = mesh),
  data = sim$data,
  trait = "trait",
  unit = "site"
)
```

standard_errors

Compute standard errors after fitting

Description

`gllvmTMB()` computes a TMB `TMB::sdreport()` at fitting time unless it is told not to (`control = gllvmTMBcontrol(se = FALSE)`). Skipping it is substantially faster, but it used to be a one-way door: the only way to get standard errors afterwards was to fit the model again. `standard_errors()` removes that door by computing the same `sdreport()` on demand from the fitted object.

Usage

```
standard_errors(fit)
```

Arguments

`fit` A fitted multivariate model returned by `gllvmTMB()`.

Details

Fits made with non-unit likelihood weights use a weighted objective rather than an ordinary likelihood. `standard_errors()` refuses those fits because the package does not yet provide sandwich-calibrated uncertainty for them; point estimates remain available from the fitted model.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
## Fit fast, decide later whether standard errors are needed.
fit <- gllvmTMB(
  traits(y1, y2) ~ 1 + latent(1 | site, d = 1),
  data = dat, family = gaussian(),
  control = gllvmTMBcontrol(se = FALSE)
)

fit <- standard_errors(fit)
summary(fit)
```

Value

The same fit, with its `sd_report` field populated (and `sdreport_error` cleared). Standard-error consumers such as `summary.gllvmTMB_multi()`, `getLV()`, `getREsd()`, and `confint(method = "wald")` work on the returned object.

R semantics matter here: the fit is **returned**, not modified in place. Assign the result – `fit <- standard_errors(fit)` – or the standard errors are discarded.

If the fit already has an `sd_report`, it is returned unchanged; the calculation is not repeated.

When this helps

The saving is real but bounded: it is worth using when a workflow fits many models and needs standard errors for only some of them (model selection, simulation loops, a grid of candidate structures). For a single fit whose summary you intend to read, the default `se = TRUE` is simpler and costs nothing extra.

Same-session only

A TMB ADFun object holds external pointers into compiled memory. Those pointers do not survive `saveRDS()` / `readRDS()` or a new R session, so `standard_errors()` can only be called on a fit made in the *current* session. This is a pre-existing property of every function in this package that reuses `fit$tmobj` (profiling, bootstrapping, `getLV()`), not a new restriction – but this function names it with a clear error rather than failing obscurely. If you need standard errors from a saved fit, refit it with `se = TRUE`.

What these standard errors are

Exactly the ones the fit would have produced at fitting time – the same TMB `sdreport()` call, on the same converged parameter vector, to the bit. Deferring the calculation does not change it, and does not change any honesty caveat that already applies to Wald standard errors from this package.

See Also

[gllvmTMBcontrol\(\)](#) for the `se` argument that defers the calculation; [getREsd\(\)](#) and [getLV\(\)](#) for accessors that read the resulting `sd_report`.

suggest_lambda_constraint

Suggest a lambda_constraint matrix for a reduced-rank fit

Description

Produces a default constraint matrix `M` for the reduced-rank loadings `Lambda` in a fit returned by [gllvmTMB\(\)](#) or in a formula-data pair that will be passed to [gllvmTMB\(\)](#). The matrix is returned in the format expected by the `lambda_constraint` argument: NA in free entries, numeric values in pinned entries. This function supplies an identification convention – an orientation/scale scaffold that resolves rotational ambiguity – and does not discover biological factor structure; a low or pinned loading is a constraint choice or a result to interpret, not by itself grounds to exclude a trait.

Usage

```
suggest_lambda_constraint(
  fit_or_formula,
  data = NULL,
  level = "unit",
  convention = c("lower_triangular", "pin_top_one", "none", "varimax_threshold",
    "wald_retention", "profile_retention"),
  trait = "trait",
  unit = "site",
  threshold = 0.3,
  retention_prob = 0.9,
  sigma_d2 = 1,
  site = NULL
)
```

Arguments

<code>fit_or_formula</code>	Either a fitted multivariate model returned by gllvmTMB() or a formula. If a formula, data must also be supplied.
<code>data</code>	A data frame. Required when <code>fit_or_formula</code> is a formula; ignored when it is a fit.
<code>level</code>	Which loading matrix to constrain: "unit" (between-unit, default) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.
<code>convention</code>	One of: "lower_triangular" (default) Pin every upper-triangular entry to 0, i.e. $M[i, j] = 0$ for $j > i$. Removes the rotational ambiguity completely. Pins $K(K-1)/2$ entries.

	"pin_top_one" Single-anchor convention: $M[1, 1] = 1$, rest NA. Sets the scale of factor 1; does NOT remove rotational ambiguity.
	"none" All-NA matrix – no pins. Useful if you plan to rotate the fitted loadings after fitting instead.
trait, unit	Name of the trait and unit (site) columns. Forwarded when <code>fit_or_formula</code> is a formula and the data does not already use defaults.
threshold	Saliency threshold on the standardised loading $\rho_{tk} = \Lambda_{tk} / \sqrt{\Sigma_{total,tt}}$. Used by "varimax_threshold" and "wald_retention". Comrey-Lee 1992 convention: 0.30 = "fair" saliency (default). Must be a finite scalar in $[\emptyset, 1)$. Raise to 0.40 for "good" / 0.50 for "very good".
retention_prob	Retention probability for "wald_retention" and "profile_retention": • "wald_retention": pin entry if $\Pr(\rho > \text{threshold}) < \text{retention_prob}$. • "profile_retention": pin entry if a profile LRT fails to reject H_0: $\Lambda = \emptyset$ at significance level $1 - \text{retention_prob}$. Must be a finite scalar in $(\emptyset, 1)$. Default 0.90 follows the applied-EFA bootstrap convention; the stricter BSEM 0.95 often over-prunes at moderate sample sizes and produces non-PD refits.
sigma_d2	Deprecated compatibility argument; ignored by the standardised-loading conventions. Their denominators now come from the model-implied total variance returned by <code>extract_Sigma()</code> .
site	Deprecated alias for unit. Emits a one-shot warning and maps to unit.

Details

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
sug <- suggest_lambda_constraint(
  value ~ 0 + trait + latent(0 + trait | site, d = 2),
  data = my_data
)
fit <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 2),
  data           = my_data,
  trait          = "trait",
  unit           = "site",
  lambda_constraint = list(unit = sug$constraint)
)
```

Value

A list with components:

constraint A $T \times K$ matrix with NA in free entries and 0 (or 1, for "pin_top_one") in pinned entries. Has trait names as rownames and "f1", ..., "fK" as colnames.

convention The chosen convention.

d The number of factors K.

n_pins Number of pinned entries.

note A short explanation of what was pinned and why.

usage_hint An example call as a string showing how to use the returned matrix with `gllvmTMB()`.

suggest_lambda_constraints

Compare several lambda_constraint suggestions

Description

Runs `suggest_lambda_constraint()` under several conventions and returns a compact comparison table. This is the convenience layer for exploratory loading workflows where the analyst wants to see the cheap point-threshold suggestion beside uncertainty-aware Wald or profile-retention suggestions before choosing the matrix to pass to `lambda_constraint`.

Usage

```
suggest_lambda_constraints(
  fit_or_formula,
  data = NULL,
  level = "unit",
  methods = c("varimax_threshold", "wald_retention"),
  trait = "trait",
  unit = "site",
  threshold = 0.3,
  retention_prob = 0.9,
  sigma_d2 = 1,
  site = NULL
)
```

Arguments

<code>fit_or_formula</code>	Either a fitted multivariate model returned by <code>gllvmTMB()</code> or a formula. If a formula, data must also be supplied.
<code>data</code>	A data frame. Required when <code>fit_or_formula</code> is a formula; ignored when it is a fit.
<code>level</code>	Which loading matrix to constrain: "unit" (between-unit, default) or "unit_obs" (within-unit). Deprecated aliases "B" and "W" are still accepted with a warning.
<code>methods</code>	Character vector of conventions to run. Accepted values are the same as the convention argument of <code>suggest_lambda_constraint()</code> .
<code>trait, unit</code>	Name of the trait and unit (site) columns. Forwarded when <code>fit_or_formula</code> is a formula and the data does not already use defaults.

threshold	Saliency threshold on the standardised loading $\rho_{tk} = \Lambda_{tk} / \sqrt{\Sigma_{total,tt}}$. Used by "varimax_threshold" and "wald_retention". Comrey-Lee 1992 convention: 0.30 = "fair" saliency (default). Must be a finite scalar in $[\emptyset, 1)$. Raise to 0.40 for "good" / 0.50 for "very good".
retention_prob	Retention probability for "wald_retention" and "profile_retention": • "wald_retention": pin entry if $\Pr(\rho > \text{threshold}) < \text{retention_prob}$. • "profile_retention": pin entry if a profile LRT fails to reject H_0: $\Lambda = \emptyset$ at significance level $1 - \text{retention_prob}$. Must be a finite scalar in $(\emptyset, 1)$. Default 0.90 follows the applied-EFA bootstrap convention; the stricter BSEM 0.95 often over-prunes at moderate sample sizes and produces non-PD refits.
sigma_d2	Deprecated compatibility argument; ignored by the standardised-loading conventions. Their denominators now come from the model-implied total variance returned by extract_Sigma() .
site	Deprecated alias for unit. Emits a one-shot warning and maps to unit.

Details

The default compares varimax_threshold and wald_retention. Add "profile_retention" to methods when you want the likelihood-ratio version; it is slower because it refits once per testable loading entry.

Template for an existing fitted model or data set; replace the named inputs with your own objects before running this code.

```
cmp <- suggest_lambda_constraints(
  fit,
  methods = c("varimax_threshold", "wald_retention"),
  threshold = 0.30,
  retention_prob = 0.90
)
cmp$summary
fit_con <- gllvmTMB(
  value ~ 0 + trait + latent(0 + trait | site, d = 2),
  data = df,
  family = binomial(),
  lambda_constraint = list(unit = cmp$recommended$constraint)
)

# Expensive: one likelihood-ratio refit per testable loading.
cmp_profile <- suggest_lambda_constraints(
  fit,
  methods = "profile_retention"
)
```

Value

A list of class `gllvmTMB_lambda_constraint_suggestions` with:

summary A data frame with one row per method and columns for decision basis, cost, number of pinned/free entries, and the helper note.

suggestions A named list of the full `suggest_lambda_constraint()` return objects, one per method.

recommended_method The highest-evidence method among the requested methods. By default this is "wald_retention"; if "profile_retention" is requested, it is recommended because it uses likelihood-ratio refits rather than the baseline Hessian.

recommended The full suggestion object for recommended_method.

tidy.gllvmTMB_multi	<i>Tidy a fitted gllvmTMB model</i>
---------------------	-------------------------------------

Description

Returns a tibble (or data.frame) of either the fixed-effect coefficient table, the random-effects variance / covariance terms, or the ordinal threshold cutpoints. Mirrors the `tidy.sdmTMB()` API but augmented for the additional covstructs and the gllvmTMB-native `ordinal_probit()` family.

Usage

```
## S3 method for class 'gllvmTMB_multi'
tidy(
  x,
  effects = c("fixed", "ran_pars", "cutpoint"),
  conf.int = FALSE,
  conf.level = 0.95,
  ...
)
```

Arguments

<code>x</code>	A fit returned by <code>gllvmTMB()</code> .
<code>effects</code>	One of "fixed" (default), "ran_pars", or "cutpoint". The "cutpoint" class returns the ordinal-probit cutpoints (one row per (trait, threshold) pair); it is empty for fits with no <code>ordinal_probit()</code> traits. (Earlier releases lumped the cutpoints into "ran_pars" as a categorisation hack — see <i>NEWS</i> .)
<code>conf.int</code>	Whether to add <code>conf.low</code> / <code>conf.high</code> columns.
<code>conf.level</code>	Confidence level for the CI.
<code>...</code>	Currently unused.

Value

A data.frame. `effect = "fixed"` rows include a link column reporting each trait's link function ("identity", "probit", "log", "logit", ...). `effect = "cutpoint"` rows carry the ordinal-probit thresholds. For a non-unit weighted objective, fixed-effect point estimates remain available but `std.error` is NA, an `inference_status` column explains the boundary, and `conf.int = TRUE` fails because no sandwich interval is certified.

traits

Wide-format trait marker for the `gllvmTMB()` formula LHS

Description

Formula-LHS marker that lets `gllvmTMB()` accept a wide data frame (one row per unit, one column per trait) without making the user pivot first to one row per (unit, trait) observation.

Usage

```
traits(...)
```

Arguments

... Column-selection expression(s) passed verbatim to `tidyr::pivot_longer(cols = ...)`. Bare names or any tidyselect verb (`all_of()`, `starts_with()`, `matches()`, etc.) are accepted.

Details

The package teaches **two shapes**, long or wide data-frame:

- **long:** `gllvmTMB(value ~ ..., data = df_long, ...)` – one row per (unit, trait) observation.
- **wide data frame:** `gllvmTMB(traits(t1, t2, ...) ~ ..., data = df_wide, ...)` – one row per unit, one column per trait, with compact formula syntax.

The soft-deprecated `gllvmTMB_wide(Y, ...)` wrapper remains exported for legacy matrix-wrapper workflows, but new examples should use `traits(...)` through `gllvmTMB()`.

Both taught shapes reach the same stacked-trait model after internal stacking; the user picks whichever shape matches their data on disk.

Because the LHS already names the response traits, the RHS can use a compact wide shorthand. `1` expands to the trait-specific intercepts `0 + trait`; ordinary predictors such as `env_temp` expand to `(0 + trait):env_temp`; and `latent(1 | individual)` expands to the long covariance syntax `latent(0 + trait | individual)`. Ordinary `latent()` includes its Ψ companion by default. The same `1 | group` shorthand is recognised for `indep()`, `dep()`, bar-style `phylo_indep()` / `phylo_dep()`, and the `spatial_*`() keywords. Species-axis phylogenetic keywords such as `phylo_latent(species, d = K)` already name their phylogenetic axis and pass through unchanged. Ordinary random-intercept terms such as `(1 | batch)` also pass through unchanged.

```
gllvmTMB(
  traits(sleep, mass, lifespan, brain) ~ 1 + env_temp +
    latent(1 | individual, d = 2),
  data  = wide_df,
  unit  = "individual",
  family = gaussian()
)
```

Internally `traits()` is implemented as a `tidyr::pivot_longer()` pre-pass: the wide data is pivoted to long format with `trait` as a factor column (levels in the order the user supplied to `traits()`) and `.y_wide_` as the response column; the LHS of the formula is rewritten from `traits(...)` to `.y_wide_`; and the compact RHS is expanded to the trait-stacked long syntax before dispatch. The explicit long RHS remains accepted, so existing calls that already write `0 + trait` and `latent(0 + trait | group)` keep working.

Tidysselect verbs are supported because `traits()` forwards its arguments to `tidyr::pivot_longer(cols = ...)`: `traits(all_of(cols))`, `traits(starts_with("sp"))`, `traits(matches("^y[0-9]+$"))`, `traits(any_of(c("a", "b")))`, and bare names all work.

Cells with NA responses are, by default, dropped via `pivot_longer(values_drop_na = TRUE)` – the canonical complete-case behaviour. Passing `missing = miss_control(response = "include")` to `gllvmTMB()` instead **keeps** every `(unit, trait)` cell and masks the NA cells out of the likelihood (the observed-response mask), preserving per-cell identity and original-row accounting in `fit$missing_data`. Users who want strict listwise drop should pre-filter the wide data before calling.

Mixed-family fits (`family = list(...)` keyed by `trait`) use the same family handling after internal stacking; `traits()` does not intercept the family argument. Per-row weight vectors of length `nrow(data)` are also replicated across traits automatically. For per-cell weight matrices, pivot to long format and pass a weights column aligned with `(unit, trait)` rows. The legacy matrix wrapper `gllvmTMB_wide()` still accepts matrix weights for migration code.

Value

A formula marker; never evaluated as a function call. The parser recognises `traits(...)` on the LHS of a `gllvmTMB()` formula and dispatches to the wide-format pivot pre-pass.

See Also

`gllvmTMB()` for model fitting. The legacy matrix wrapper `gllvmTMB_wide(Y, ...)` is soft-deprecated in 0.2.0; wide-data examples now use the `traits(...)` LHS through `gllvmTMB()`.

Examples

```
# Wide format: one column per trait; traits() stacks them internally.
# The LHS *is* the trait spec, so no `trait =` argument is needed.
set.seed(1)
df_wide <- data.frame(
  unit = factor(paste0("u", 1:40)),
  t1 = rnorm(40), t2 = rnorm(40), t3 = rnorm(40)
)
fit <- gllvmTMB(
  traits(t1, t2, t3) ~ 1 + latent(1 | unit, d = 2),
  data = df_wide, unit = "unit"
)
extract_Sigma(fit)
```

Index

add_utm_columns, 5
AIC(), 99, 101
animal_dep, 6
animal_dep(), 10, 13, 84, 128
animal_indep, 7
animal_indep(), 9, 10, 13, 84, 128
animal_latent, 8
animal_latent(), 6, 10, 11, 13, 84, 128
animal_scalar, 9
animal_scalar(), 6, 7, 9, 11, 13, 128, 130
animal_slope, 11
animal_slope(), 10, 80, 85
animal_unique, 12

BIC(), 99, 101
binomial(), 18, 38, 110
block_V, 14
block_V(), 122
bootstrap_Sigma, 15, 48
bootstrap_Sigma(), 31–33, 44, 50, 69, 71–73, 169, 170, 188

categorical, 18, 125
categorical(), 106, 107, 126
check_auto_residual, 19
check_auto_residual(), 91
check_gllvmTMB, 19
check_gllvmTMB(), 15, 42, 89, 91, 92, 105, 168, 176, 181
coef.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
coef.gllvmTMB_multi
 (gllvmTMB_multi-vcov), 97
coef.gllvmTMB_va (gllvmTMB_va-methods), 98
compare_dep_vs_two_psi, 23
compare_dep_vs_two_psi(), 27, 92
compare_indep_vs_two_psi, 26
compare_indep_vs_two_psi(), 25, 26, 92
compare_loadings, 28
compare_Sigma_table, 28
compare_Sigma_table(), 156, 157
confint(), 84, 87, 97–99, 102
confint.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
confint.gllvmTMB_multi, 30, 48, 64
confint.gllvmTMB_multi(), 34, 36, 45, 173, 175
confint.gllvmTMB_va
 (gllvmTMB_va-methods), 98
confint_inspect, 34
confint_inspect(), 195
confirmatory_lambda, 36
confirmatory_lambda(), 83, 117, 118
cumulative_logit, 38
cumulative_logit(), 18, 106, 107

dep, 38
dep(), 84, 134, 182, 200
deviance.gllvmTMB_multi, 40
deviance.gllvmTMB_va
 (gllvmTMB_va-methods), 98
df.residual.gllvmTMB_va
 (gllvmTMB_va-methods), 98
diag_re, 115
diagnose_kernel_separability, 40
diagnose_kernel_separability(), 112
diagnostic_table, 41

extract_coevolution_modules, 43
extract_communality, 44, 48, 64
extract_communality(), 16, 33, 61–63, 71, 92, 117, 118, 178
extract_correlations, 46, 64
extract_correlations(), 19, 31, 33, 45, 50, 61, 72, 73, 150–152, 173
extract_cross_correlations, 49
extract_cross_correlations(), 16
extract_cutpoints, 51
extract_cutpoints(), 127, 166

- extract_Gamma, [53](#)
- extract_Gamma(), [43](#), [112](#), [163](#)
- extract_ICC_site(), [16](#), [45](#), [62](#), [63](#)
- extract_loadings, [54](#)
- extract_loadings(), [99](#), [102](#)
- extract_lv_effects, [55](#)
- extract_lv_effects(), [76](#), [115](#)
- extract_Omega, [56](#)
- extract_Omega(), [19](#), [61](#), [68](#), [71](#)
- extract_ordination, [57](#)
- extract_ordination(), [55](#), [56](#), [66](#), [99](#), [102](#)
- extract_phylo_signal, [59](#), [64](#)
- extract_phylo_signal(), [57](#), [62](#), [63](#)
- extract_proportions, [61](#)
- extract_proportions(), [57](#), [61](#), [71](#)
- extract_repeatability, [63](#)
- extract_repeatability(), [31](#), [33](#), [45](#), [61](#)
- extract_residual_cor
 - (extract_residual_cov), [65](#)
- extract_residual_cov, [65](#)
- extract_residual_split(), [57](#), [62](#), [63](#)
- extract_rotated_loadings_table, [65](#)
- extract_rotated_loadings_table(), [154](#), [155](#)
- extract_Sigma, [48](#), [67](#)
- extract_Sigma(), [16](#), [19](#), [26](#), [27](#), [33](#), [40](#), [43](#), [45](#), [50](#), [53](#), [57](#), [61](#), [62](#), [65](#), [66](#), [72](#), [73](#), [88](#), [92](#), [110](#), [113–115](#), [117](#), [134](#), [136](#), [144](#), [169](#), [170](#), [182](#), [196](#), [200](#), [201](#), [209](#), [211](#)
- extract_Sigma_table, [71](#)
- extract_Sigma_table(), [28](#), [29](#), [156](#), [158–161](#)
- fitted.gllvmTMB_julia
 - (gllvmTMB_julia-methods), [93](#)
- fitted.gllvmTMB_multi, [73](#)
- fitted.gllvmTMB_va
 - (gllvmTMB_va-methods), [98](#)
- flag_unreliable_loadings, [74](#)
- flag_unreliable_loadings(), [118](#), [153](#), [154](#)
- fmesh::fm_nonconvex_hull(), [121](#)
- get_crs, [75](#)
- get_crs(), [5](#)
- getLoadings(), [54](#), [55](#), [99](#), [102](#)
- getLV(), [99](#), [102](#), [207](#), [208](#)
- getREsd(), [207](#), [208](#)
- getResidualCor(), [65](#)
- getResidualCov(), [65](#)
- gllvm_julia_fit, [76](#)
- gllvm_julia_fit(), [93](#), [94](#)
- gllvm_julia_setup, [78](#)
- gllvm_julia_setup(), [77](#)
- gllvmTMB, [46](#), [63](#), [79](#), [125](#)
- gllvmTMB(), [10](#), [14](#), [15](#), [18–20](#), [24](#), [26](#), [28–30](#), [33](#), [34](#), [37](#), [38](#), [40](#), [44](#), [46](#), [52](#), [54](#), [55](#), [57–59](#), [62](#), [63](#), [65](#), [66](#), [68](#), [72](#), [73](#), [77](#), [80](#), [81](#), [89](#), [92](#), [95–97](#), [99](#), [100](#), [103](#), [106–108](#), [110](#), [111](#), [122–124](#), [132](#), [140](#), [141](#), [145](#), [150](#), [152](#), [154](#), [156](#), [158](#), [160](#), [162](#), [164](#), [166](#), [169](#), [173](#), [174](#), [177](#), [179](#), [181](#), [184–186](#), [188](#), [197–199](#), [201](#), [202](#), [204–206](#), [208](#), [210](#), [212–214](#)
- gllvmTMB_check_consistency, [89](#)
- gllvmTMB_diagnose, [91](#)
- gllvmTMB_diagnose(), [15](#), [19](#), [69](#), [88](#), [89](#), [91](#), [104](#), [181](#)
- gllvmTMB_julia-methods, [93](#)
- gllvmTMB_multi-methods, [95](#)
- gllvmTMB_multi-vcov, [87](#), [97](#)
- gllvmTMB_va-methods, [98](#), [103](#)
- gllvmTMB_wide(), [82](#), [88](#), [214](#)
- gllvmTMBcontrol, [100](#)
- gllvmTMBcontrol(), [103](#), [124](#), [177](#), [208](#)
- graphics::plot(), [147](#)
- impute_model, [106](#)
- impute_model(), [18](#), [38](#), [84](#)
- imputed, [107](#)
- imputed(), [141](#), [165](#)
- indep, [108](#)
- indep(), [40](#), [84](#), [115](#), [136](#), [182](#), [201](#), [203](#), [204](#)
- isdmsource, [110](#)
- isdmsource(), [111](#)
- isdmsources, [110](#)
- isdmsources(), [110](#)
- kernel_dep(kernel_latent), [112](#)
- kernel_dep(), [85](#), [114](#)
- kernel_indep(kernel_latent), [112](#)
- kernel_indep(), [85](#), [113](#), [114](#)
- kernel_latent, [112](#)
- kernel_latent(), [85](#), [114](#)
- kernel_scalar, [113](#)
- kernel_scalar(), [112](#)

- kernel_unique, 114
- latent, 114
- latent(), 40, 84, 110, 182, 203, 204, 206
- loading_ci, 116
- loading_ci(), 31, 33, 74, 75, 118, 119, 152–154, 195, 196
- loading_profile, 118
- loading_profile(), 31, 33, 117
- logLik(), 96, 99, 101
- logLik.gllvmTMB_julia
(gllvmTMB_julia-methods), 93
- logLik.gllvmTMB_multi
(gllvmTMB_multi-methods), 95
- logLik.gllvmTMB_multi(), 40
- logLik.gllvmTMB_va
(gllvmTMB_va-methods), 98
- make_cross_kernel, 119
- make_cross_kernel(), 43, 53, 112, 163, 170–172
- make_mesh, 120
- make_mesh(), 82, 147, 197, 199, 201, 202, 204, 205
- meta_known_V(), 85, 122
- meta_V, 122
- meta_V(), 10, 14, 80, 85
- mi (impute_model), 106
- mi_group (impute_model), 106
- miss_control, 123
- miss_control(), 83, 107, 164, 166, 184
- multinomial, 124
- multinomial(), 166, 186
- nobs.gllvmTMB_multi
(gllvmTMB_multi-methods), 95
- nobs.gllvmTMB_va (gllvmTMB_va-methods), 98
- ordinal_probit, 126, 126
- ordinal_probit(), 38, 51–53, 81, 88, 126, 164, 166
- pedigree_to_A, 128
- pedigree_to_A(), 130
- pedigree_to_Ainv_sparse, 129
- phylo, 130
- phylo(), 139, 197, 199
- phylo_dep, 133
- phylo_dep(), 6, 40, 84, 133, 136, 138, 139, 144, 200
- phylo_indep, 134
- phylo_indep(), 7, 13, 84, 110, 133, 134, 138, 139, 143, 144, 201
- phylo_latent, 136
- phylo_latent(), 9, 84, 115, 133, 134, 136, 139, 141–144, 202, 203
- phylo_rr(), 138
- phylo_scalar, 139
- phylo_scalar(), 10, 113, 114, 133, 138, 142–144, 182
- phylo_signal_mi, 140
- phylo_slope, 141
- phylo_slope(), 11, 80, 85
- phylo_unique, 142
- phylo_unique(), 13
- plot(), 118
- plot.gllvmTMB_multi, 145
- plot.gllvmTMB_multi(), 66, 152, 155, 161
- plot.gllvmTMBmesh, 147
- plot.profile_derived, 148
- plot.profile_derived(), 173
- plot.profile_loadings, 148
- plot.profile_loadings(), 119, 148
- plot.sdmTMBmesh (plot.gllvmTMBmesh), 147
- plot_anisotropy, 149
- plot_anisotropy2 (plot_anisotropy), 149
- plot_correlations, 150
- plot_correlations(), 161
- plot_loadings_confidence_eye, 152
- plot_rotated_loadings, 154
- plot_Sigma_comparison, 156
- plot_Sigma_comparison(), 29, 159
- plot_Sigma_heatmap, 157
- plot_Sigma_heatmap(), 151
- plot_Sigma_table, 159
- plot_Sigma_table(), 29, 151, 152, 157, 159
- poisson(), 110
- predict.gllvmTMB_julia
(gllvmTMB_julia-methods), 93
- predict.gllvmTMB_multi, 161
- predict.gllvmTMB_multi(), 73, 166
- predict.gllvmTMB_va
(gllvmTMB_va-methods), 98
- predict_cross_covariance, 163
- predict_missing, 164
- predict_missing(), 80, 84, 85, 108, 123

predictive_check, 167
 predictive_check(), 41, 42
 print.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
 print.gllvmTMB_multi
 (gllvmTMB_multi-methods), 95
 print.gllvmTMB_Sigma_phy_slope, 168
 print.gllvmTMB_va
 (gllvmTMB_va-methods), 98
 print.summary.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
 print.summary.gllvmTMB_multi
 (gllvmTMB_multi-methods), 95
 print.summary.gllvmTMB_va
 (gllvmTMB_va-methods), 98
 profile_ci_phylo_signal(), 31, 173
 profile_ci_total_variance, 169
 profile_ci_total_variance(), 15
 profile_cross_rho, 170
 profile_cross_rho(), 53, 172
 profile_cross_rho_ci, 172
 profile_phylo_signal, 172
 profile_phylo_signal(), 148
 profile_targets, 173
 profile_targets(), 34–36

 qr(), 186

 residuals.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
 residuals.gllvmTMB_multi, 175
 residuals.gllvmTMB_multi(), 41, 168
 residuals.gllvmTMB_va
 (gllvmTMB_va-methods), 98
 ridge_path, 177
 rotate_loadings, 179
 rotate_loadings(), 56, 65, 66, 155, 203

 sanity_multi, 181
 sanity_multi(), 90–92
 scalar, 182
 scalar(), 113, 114
 screen_control, 183
 screen_control(), 184
 screen_gllvmTMB, 184
 screen_gllvmTMB(), 88, 179, 183
 screen_table, 187
 screen_table(), 186
 sf::st_transform(), 5

 simulate.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
 simulate.gllvmTMB_multi, 188
 simulate.gllvmTMB_multi(), 17, 167, 175, 176
 simulate_site_trait, 189
 simulate_site_trait(), 88, 192, 193
 simulate_unit_trait, 191
 simulate_unit_trait(), 191
 slope_sd_ci, 193
 spatial, 196
 spatial_dep, 199
 spatial_dep(), 40, 85, 134, 199, 202, 203
 spatial_indep, 200
 spatial_indep(), 85, 110, 136, 199, 200, 202–206
 spatial_latent, 202
 spatial_latent(), 85, 197, 199–201, 204–206
 spatial_scalar, 203
 spatial_unique, 205
 spde(), 199, 204–206
 standard_errors, 206
 standard_errors(), 98
 stats::confint(), 32
 suggest_lambda_constraint, 208
 suggest_lambda_constraint(), 37, 83, 92, 117, 210, 212
 suggest_lambda_constraints, 210
 summary(), 98
 summary.gllvmTMB_julia
 (gllvmTMB_julia-methods), 93
 summary.gllvmTMB_multi
 (gllvmTMB_multi-methods), 95
 summary.gllvmTMB_multi(), 207
 summary.gllvmTMB_va
 (gllvmTMB_va-methods), 98

 tidy.gllvmTMB_multi, 212
 TMB::checkConsistency(), 91
 TMB::sdreport(), 108, 206
 TMB::tmbprofile(), 32, 34
 tmbprofile_wrapper(), 33, 36, 175
 traits, 213
 traits(), 79, 80, 82, 85, 88, 184

 unique(), 143, 206

 vcov(), 99, 102

`vcov.gllvmTMB_multi`
 (`gllvmTMB_multi-vcov`), [97](#)
`vcov.gllvmTMB_va` (`gllvmTMB_va-methods`),
 [98](#)

`weights.gllvmTMB_va`
 (`gllvmTMB_va-methods`), [98](#)