Package 'pigauto'

July 30, 2026

Title Fill in Missing Species Traits Using a Phylogenetic Tree

Version 0.10.0

Description Imputes missing species trait data for comparative analyses by combining three sources of information: phylogenetic similarity (closely related species share similar traits), cross-trait correlations (observed traits inform missing ones), and optional environmental covariates (climate, habitat, geography). Handles continuous measurements, counts, binary variables, ordered categories, unordered categories, bounded proportions, zero-inflated counts, and compositional multi-proportion data in a single call. The method blends a phylogenetic baseline with a graph neural network correction; a per-trait gate calibrated on held-out data ensures the network only contributes when it improves on the baseline. Provides conformal prediction intervals for continuous, count, and ordinal traits and an experimental analysis-aware multiple-imputation workflow for one missing continuous covariate in Gaussian linear, binomial-logit, and Gaussian random-intercept models, with Rubin pooling limited to fixed effects. Stochastic graph-network and posterior-tree completions are prediction diagnostics rather than validated inferential imputations. Tested up to 10,000 species.
Bundled datasets include 300-species and 9,993-species bird-trait subsets with matching example phylogenetic trees.
Rubin (1987, ISBN:978-0-471-08705-2); Vovk et al. (2005, ISBN:978-0-387-25061-8); Nakagawa and de Villemereuil (2019) <doi:10.1093/sysbio/syy089>.

License MIT + file LICENSE

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

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

Encoding UTF-8

LazyData true

LazyDataCompression xz

Depends R (>= 4.1)

Imports torch, ape, ggplot2, Matrix, rlang, stats, utils, graphics, grDevices, withr,

Suggests testthat, jsonlite, knitr, rmarkdown, pkgdown, RSpectra, Rphylopars, jomo, MCMCglmm, glmmTMB, lme4, nlme, phangorn, phylolm, rgbif, smcfcs, terra, phytools

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

NeedsCompilation no

Author Shinichi Nakagawa [aut, cre, cph]

Maintainer Shinichi Nakagawa <itchyshin@gmail.com>

Repository CRAN

Date/Publication 2026-07-30 17:10:22 UTC

Contents

`avonet300` *AVONET morphological and ecological trait data for 300 bird species*

Description

A data frame with 300 rows and 8 columns: 4 continuous morphometric traits and 3 ecological traits (2 categorical, 1 ordinal) for demonstrating mixed-type imputation. Species names are in the `Species_Key` column. Set `rownames(df) <- df$Species_Key; df$Species_Key <- NULL` before passing to `preprocess_traits`.

Usage

```
avonet300
```

Format

A data frame with 300 rows and 8 variables:

Species_Key Character. Species name in BirdTree format (spaces replaced by underscores).

Mass Numeric. Body mass (grams).

Beak.Length_Culmen Numeric. Beak length from culmen (mm).

Tarsus.Length Numeric. Tarsus length (mm).

Wing.Length Numeric. Wing length (mm).

Trophic.Level Factor. Dietary category: Carnivore, Herbivore, Omnivore, or Scavenger.

Primary.Lifestyle Factor. Lifestyle category: Aerial, Aquatic, Generalist, Insessorial, or Terrestrial.

Migration Ordered factor. Migration strategy: Resident < Partial < Full.

Source

Tobias et al. (2022) AVONET: morphological, ecological and geographical data for all birds. *Ecology Letters*, 25, 581-597. Species labels are aligned to the bundled example phylogenies.

avonet_full *Full AVONET morphological and ecological trait data for 9,993 bird species*

Description

The full-scale counterpart to avonet300: every bird species for which AVONET3 and the bundled example phylogeny agree on both a species label and a complete set of continuous morphometric measurements. The schema is identical to avonet300 (same trait columns, same factor encodings, same Species_Key column) so any code that runs on avonet300 runs on avonet_full with no modification.

Usage

```
avonet_full
```

Format

A data frame with 9,993 rows and 8 variables. Columns match avonet300: Species_Key, Mass, Beak.Length_Culmen, Tarsus.Length, Wing.Length, Trophic.Level, Primary.Lifestyle, Migration.

Details

Unlike the 300-species subset, native AVONET missingness is PRESERVED in the two ecological columns: 4 NAs in Trophic.Level and 20 NAs in Migration. The four continuous columns are complete by construction.

Use avonet_full + tree_full to exercise pigauto at real-world scale (the AVONET missingness sweep benchmark uses this dataset). For quick examples or unit tests, prefer the 300-species avonet300 subset – it is ~30x smaller and loads instantly.

See avonet300 for the full column schema (same traits and encodings).

Source

Tobias et al. (2022) AVONET: morphological, ecological and geographical data for all birds. *Ecology Letters*, 25, 581-597. Species labels are aligned to the bundled example phylogeny.

See Also

avonet300, tree_full

`build_phylo_graph` *Build a phylogenetic graph representation from a tree*

Description

Computes a symmetric-normalised Gaussian kernel adjacency matrix and Laplacian spectral node features from the cophenetic distance matrix. Results are optionally cached to an `.rds` file.

Usage

```
build_phylo_graph(tree, k_eigen = "auto", sigma_mult = 0.5, cache_path = NULL)
```

Arguments

`tree`	object of class `"phylo"`.
`k_eigen`	integer or `"auto"`. Number of Laplacian eigenvectors to use as node features. When `"auto"` (default), scales with tree size: `min(max(ceiling(n/20), 4), 32)`, giving 4 for very small trees, 8 for 100-160 tips, 15 for 300 tips, and 32 for 640+ tips.
`sigma_mult`	numeric. Bandwidth multiplier (call it s): $\sigma = \text{median}(D) \times s$ (default `0.5`).
`cache_path`	character or `NULL`. Path to an `.rds` cache file. If the file exists and dimensions match, it is loaded instead of recomputing. `NULL` disables caching (default).

Details

The graph is built in three steps: (1) cophenetic distances between all pairs of tips; (2) Gaussian kernel $A_{ij} = \exp(-d_{ij}^2/(2\sigma^2))$ with $\sigma = \text{median}(D) \times s$ (where s is the user-supplied `sigma_mult`); (3) symmetric normalisation $\tilde{A} = D^{-1/2}AD^{-1/2}$ with self-loops added before normalisation.

Spectral node features are the `k_eigen` smallest non-zero eigenvectors of the unnormalised Laplacian, which encode the broad cluster structure of the phylogeny.

Value

A list with:

adj Numeric matrix (n x n). Symmetric-normalised adjacency.

coords Numeric matrix (n x k_eigen). Spectral node features.

n Integer. Number of tips.

sigma Numeric. Bandwidth used.

D Numeric matrix (n x n). Cophenetic (patristic) distances between tips, row/column-ordered by `tree$tip.label`. Returned so downstream functions can reuse it instead of calling `ape::cophenetic.phylo()` again.

D_sq Numeric matrix (n x n). Element-wise squared cophenetic distances (`D * D`). Used by `GraphTransformerBlock` for the learnable per-head Gaussian bandwidth (B2 rate-aware attention). Not freed during `impute()` training — only `D` is freed.

R_phy Numeric matrix (n x n). Phylogenetic correlation matrix `cov2cor(ape::vcv(tree))` (diagonal = 1). Used by `fit_baseline` for BM conditional imputation.

Scaling

For trees with more than 7500 tips, `build_phylo_graph()` uses **RSpectra**'s sparse Lanczos eigensolver (`eigs_sym`) instead of the dense `base::eigen()`. This reduces the spectral step from $O(n^3)$ (hours at $n = 10{,}000$) to $O(n \cdot k \cdot \text{iters})$ (seconds to a minute). The threshold is conservative because the dense eigensolver is competitive on small and mid-size trees, and on pathological ultrametric simulations (`ape::rcoal`) the Laplacian spectrum has tight degenerate clusters that slow down Lanczos until the tree is fairly large. On realistic asymmetric phylogenies with variable branch lengths (e.g. the AVONET BirdTree Stage2 Hackett tree) the sparse path wins much earlier, and at $n = 9{,}993$ it delivers a ~10x speedup over dense. The dense path is kept as the default for smaller trees and as a fallback when **RSpectra** is not installed, or if Lanczos fails to converge. The cophenetic distance matrix is computed once per call and reused for both the adjacency and the spectral features; it is also returned in the result so that downstream consumers (notably `fit_baseline`) can skip recomputation.

Examples

```
set.seed(1)
tree <- ape::rtree(30)
g <- build_phylo_graph(tree, k_eigen = 4)
dim(g$adj)    # 30 x 30
dim(g$coords) # 30 x 4
dim(g$D)      # 30 x 30
```

`calibration_df` *Compute calibration data for probability predictions*

Description

Bins predicted probabilities and computes observed frequencies within each bin. Useful for calibration plots of binary classifiers.

Usage

```
calibration_df(truth, prob, n_bins = 10L)
```

Arguments

`truth`	numeric vector (0/1 or TRUE/FALSE).
`prob`	numeric vector of predicted probabilities.
`n_bins`	integer, number of bins (default 10).

Value

A data.frame with columns `bin_mid`, `obs_freq`, `mean_pred`, `n`.

Examples

```
calibration_df(
  truth = c(0, 0, 1, 1),
  prob = c(0.1, 0.3, 0.7, 0.9),
  n_bins = 2
)
```

`compare_methods` *Compare BM baseline and pigauto methods across replicates*

Description

Runs a full comparison of the BM baseline and pigauto (with attention + calibration) over multiple random seeds. Returns a tidy data.frame suitable for plotting or downstream analysis.

Usage

```
compare_methods(
  data,
  tree,
  splits = NULL,
  seeds = NULL,
  epochs = 500L,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	pigauto_data object.
`tree`	phylo object.
`splits`	pre-computed splits (applied to all reps) or `NULL` to create fresh splits per seed.
`seeds`	optional integer vector of random seeds for replication. The default `NULL` performs one run using the current RNG stream.
`epochs`	number of training epochs.
`verbose`	logical.
`...`	additional arguments passed to `fit_pigauto`.

Details

For each seed the function:

1. Creates train/val/test splits.
2. Fits the phylogenetic BM baseline.
3. Fits pigauto (with attention and calibration enabled by default).
4. Evaluates both methods on the test split.
5. Collects results into a single data.frame.

Value

A data.frame with columns: method, trait, type, metric, value, rep.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
cmp <- compare_methods(preprocess_traits(traits, tree), tree, seeds = 1L,
                       epochs = 5L, verbose = FALSE)
aggregate(value ~ method + trait + metric, data = cmp, FUN = mean)
```

---

confusion_matrix *Compute a confusion matrix for categorical or binary predictions*

Description

Compute a confusion matrix for categorical or binary predictions

Usage

```
confusion_matrix(truth, predicted, levels = NULL)
```

Arguments

truth	factor or character vector of true classes.
predicted	factor or character vector of predicted classes.
levels	character vector of class levels (default: union of truth and predicted levels).

Value

A list with:

table Confusion matrix (rows = truth, columns = predicted).

accuracy Overall accuracy.

per_class Data.frame with per-class precision, recall, F1.

Examples

```
confusion_matrix(
  truth = c("a", "a", "b", "b"),
  predicted = c("a", "b", "b", "b")
)
```

`cross_validate` *k-fold cross-validation for pigauto trait imputation*

Description

Performs stratified k-fold cross-validation by rotating which cells serve as the test set. Returns per-fold and aggregated metrics.

Usage

```
cross_validate(
  data,
  tree,
  k = 5L,
  seeds = NULL,
  epochs = 500L,
  verbose = TRUE,
  ...
)
```

Arguments

`data`	pigauto_data object (output of `preprocess_traits`).
`tree`	phylo object.
`k`	integer. Number of folds (default 5).
`seeds`	optional integer vector. Seeds for replicate runs. The default `NULL` performs one run using the current RNG stream.
`epochs`	integer. Training epochs per fold (default 500).
`verbose`	logical. Print progress (default TRUE).
`...`	Additional arguments passed to `fit_pigauto` (e.g. `hidden_dim`, `k_eigen`, `use_attention`).

Value

A list of class `"pigauto_cv"` with:

results Data.frame with columns: fold, rep, trait, type, metric, value.

summary Data.frame with mean and sd across folds/reps for each trait + metric.

conformal_coverage Data.frame of coverage per trait across folds (if available).

k Number of folds.

n_reps Number of replicates.

Examples

```
data(avonet300, tree300, package = "pigauto")
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
cv <- cross_validate(preprocess_traits(traits, tree), tree, k = 3L,
                     seeds = 1L, epochs = 5L, verbose = FALSE)
print(cv)
summary(cv)
```

`ctmax_sim` *Simulated multi-observation-per-species CTmax data*

Description

A simulated dataset mimicking a thermal tolerance study where each species has multiple measurements of critical thermal maximum (CTmax) taken at different acclimation temperatures. The data-generating process is

Usage

```
ctmax_sim
```

Format

A data frame with 1,464 rows and 3 variables:

species Character. Species name matching `tree300` tips.

acclim_temp Numeric. Acclimation temperature (degrees C). This is an observation-level covariate that varies within species.

CTmax Numeric. Critical thermal maximum (degrees C). Contains NA values for unobserved species and MCAR missingness.

Details

$$CTmax_{ij} = 38 + phylo_i + 0.50 \times acclim\_temp_{ij} + \epsilon_{ij}$$

where $phylo_i$ follows Brownian motion on `tree300`, the within-species acclimation response ratio is 0.50, and $\epsilon_{ij} \sim N(0, 1.5)$. Thirty percent of species are entirely unobserved (all CTmax values are NA), and an additional 15\ observations are missing at random.

This dataset demonstrates pigauto's multi-observation and observation-level covariate support. Use with `tree300` and set `species_col = "species"` in `impute`.

Source

Simulated. See `data-raw/make_ctmax_sim.R` for the generation script.

See Also

`tree300`, `impute`

`evaluate` *Evaluate a fitted pigauto model on its test set*

Description

Computes type-specific performance metrics for each trait. By default the model is evaluated on the test split stored in the fit object, but an alternative `data` / `splits` can be supplied.

Usage

```
evaluate(fit, data = NULL, splits = NULL)
```

Arguments

`fit`	pigauto_fit object.
`data`	pigauto_data object (default: `NULL`, uses the training data stored in the fit via `predict()`).
`splits`	splits object (default: `NULL`, uses `fit$splits`).

Details

Metrics by trait type:

continuous RMSE, Pearson r, MAE

count RMSE, MAE, Pearson r

binary Accuracy, Brier score

categorical Accuracy (overall)

ordinal RMSE (on integer scale), Spearman rho

When conformal scores are present in the fit, conformal coverage at the 95\ ordinal traits.

When the fit includes a baseline, baseline metrics are appended with `method = "baseline"` for direct comparison.

Value

A `data.frame` with columns: `method`, `trait`, `type`, `metric`, `value`, `n_test`.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
pd <- preprocess_traits(traits, tree)
splits <- make_missing_splits(pd$X_scaled, trait_map = pd$trait_map)
fit <- fit_pigauto(pd, tree, splits = splits, epochs = 5L,
                   verbose = FALSE)
eval_df <- evaluate(fit, data = pd)
eval_df[eval_df$metric == "rmse", ]
```

`evaluate_imputation` *Evaluate imputation performance against known values*

Description

Computes type-specific metrics for each trait on the validation and test splits. When a `trait_map` is supplied, metrics are dispatched per trait type; otherwise the function falls back to continuous-only metrics (RMSE, Pearson r, 95% coverage).

Usage

```
evaluate_imputation(pred, truth, splits, pred_se = NULL, trait_map = NULL)
```

Arguments

`pred`	predicted values: either a numeric matrix in latent scale (same dimensions as `truth`), or a `"pigauto_pred"` object from `predict.pigauto_fit`.
`truth`	numeric matrix of true values in latent scale (from `pigauto_data$X_scaled`).
`splits`	list (output of `make_missing_splits`).
`pred_se`	numeric matrix of prediction SEs (same scale as `pred`). Used for 95\ when `pred` is a `pigauto_pred` (uses `pred$se`).
`trait_map`	list of trait descriptors (from `pigauto_data`). If `NULL` and `pred` is not a `pigauto_pred`, the v0.1 all-continuous evaluation is used.

Details

Metrics by trait type:

continuous RMSE, Pearson r, 95\ supplied)

proportion RMSE, Pearson r, 95\ supplied)

count RMSE, MAE, Pearson r

ordinal RMSE, Spearman rho

binary Accuracy, Brier score

categorical Accuracy

zi_count RMSE, MAE, Pearson r, zero-accuracy, Brier score (on gate)

For binary and categorical traits the function accepts either a `pigauto_pred` object (preferred, gives access to probabilities) or raw matrices (latent scale).

Value

A `data.frame` with columns `split`, `trait`, `type`, `n`, and type-specific metric columns.

Examples

```
truth <- matrix(c(1, 2, 3, 4), nrow = 2L)
splits <- list(test_idx = 1L, val_idx = 2L, n = 2L, p = 2L)
evaluate_imputation(truth, truth, splits)
```

---

fit_baseline *Fit the phylogenetic baseline*

Description

Dispatches to pigauto's phylogenetic baseline machinery and returns imputed latent-scale means and standard errors for every species.

Usage

```
fit_baseline(
  data,
  tree,
  splits = NULL,
  model = "BM",
  graph = NULL,
  multi_obs_aggregation = c("hard", "soft"),
  lambda_mode = c("fixed_1", "estimate", "cv", "bayes"),
  em_iterations = 0L,
  em_tol = 0.001,
  em_offdiag = FALSE
)
```

Arguments

`data`	object of class `"pigauto_data"`.
`tree`	object of class `"phylo"`.
`splits`	list (output of `make_missing_splits`) or `NULL`.
`model`	character. Evolutionary model: `"BM"` (default) or `"OU"`.

graph — optional list returned by build_phylo_graph. When supplied, graph$D (cophenetic distances) is reused for label propagation and graph$R_phy (phylogenetic correlation matrix) is reused for BM imputation, avoiding duplicate $O(n^2)$ allocations. When NULL (default), both matrices are computed here.

multi_obs_aggregation — character. How to aggregate multiple observations per species before the Level-C (Rphylopars) baseline: "hard" (default) thresholds binary proportions at 0.5 and uses argmax for categorical, matching Phase 10 behaviour. "soft" preserves species-level proportions and dispatches the truncated-Gaussian soft E-step (estep_liability_binary_soft) so that intermediate class frequencies contribute fractional liability evidence. Only relevant for multi-obs data with binary or categorical traits when the Level-C joint baseline is active.

lambda_mode — character. Pagel-lambda mode for the BM baseline. "fixed_1" preserves the default Brownian correlation matrix; "estimate", "cv", and "bayes" delegate lambda handling to the per-column BM path.

em_iterations — integer. Number of Phase 6 EM iterations for the threshold-joint baseline (binary + ordinal + OVR categorical). Default 0L disables the EM loop and preserves v0.9.1 output byte-for-byte. When >= 1, the BM rate Σ learned by Rphylopars::phylopars() at iteration k is fed back as the per-trait prior SD at iteration $k+1$, up to em_iterations times or until em_tol convergence. em_iterations = 1L is a degenerate single-pass run and produces the same baseline output as 0L; >= 2L is needed for actual iteration. Only affects the threshold-joint path (continuous-only traits pass through the existing joint MVN path unchanged).

em_tol — numeric. Relative-Frobenius convergence tolerance for the Phase 6 / 7 EM loop. Early-stops when $||\Sigma_k - \Sigma_{k-1}||_F / ||\Sigma_{k-1}||_F <$ em_tol. Default 1e-3.

em_offdiag — logical. Phase 7 opt-in: when TRUE AND em_iterations >= 2L, each liability cell's prior at iteration $k+1$ is the conditional-MVN (μ, sd) given the posterior liability of other traits at iteration k, using the full off- diagonal entries of Σ. Binary + ordinal only (OVR categorical stays on Phase 6 diagonal). Default FALSE preserves Phase 6 behaviour.

Details

When splits is supplied the val and test cells are masked to NA before fitting, so the baseline is evaluated under the same conditions as fit_pigauto.

Continuous-family columns use Brownian-motion conditional MVN baselines on the phylogenetic correlation matrix, either independently or through the joint MVN path when the data and optional dependencies support it. Binary, ordinal, categorical, and zero-inflated gate columns use the appropriate label-propagation or threshold/liability baseline candidates, with per-column fallbacks when a joint path is not available.

Value

A list with:

mu Numeric matrix (n_species x p_latent), baseline means in latent scale.

se Numeric matrix (n_species x p_latent), standard errors.

Examples

```
data(avonet300, tree300, package = "pigauto")
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
pd     <- preprocess_traits(traits, tree)
splits <- make_missing_splits(pd$X_scaled, trait_map = pd$trait_map)
bl     <- fit_baseline(pd, tree, splits)
```

`fit_pigauto` *Fit a pigauto model for trait imputation*

Description

Trains a pigauto model: a gated ensemble of a phylogenetic baseline and an attention-based graph neural network correction, implemented as an internal torch module (`ResidualPhyloDAE`; "Residual" here refers to the ResNet-style skip connections in the GNN layers, not to a statistical residual). For continuous, count, and ordinal traits the baseline is Brownian motion (phylogenetic correlation matrix); for binary and categorical traits it is phylogenetic label propagation. Supports all five trait types via a unified latent space.

Usage

```
fit_pigauto(
  data,
  tree,
  splits = NULL,
  graph = NULL,
  baseline = NULL,
  hidden_dim = 64L,
  k_eigen = "auto",
  n_gnn_layers = 2L,
  gate_cap = 0.8,
  use_attention = TRUE,
  use_transformer_blocks = TRUE,
  n_heads = 4L,
  ffn_mult = 4L,
  use_trait_attention = FALSE,
  n_trait_heads = 2L,
  trait_embed_dim = 32L,
  dropout = 0.1,
  lr = 0.003,
  weight_decay = 1e-04,
  epochs = 3000L,
  corruption_rate = 0.55,
```

```
  corruption_start = 0.2,
  corruption_ramp = 500L,
  refine_steps = 8L,
  lambda_shrink = 0.03,
  lambda_gate = 0.01,
  warmup_epochs = 200L,
  edge_dropout = 0.1,
  eval_every = 100L,
  patience = 10L,
  clip_norm = 1,
  conformal_method = c("split", "bootstrap"),
  conformal_bootstrap_B = 500L,
  conformal_split_val = FALSE,
  gate_method = c("cv_folds", "median_splits", "single_split"),
  gate_splits_B = 31L,
  gate_cv_folds = 5L,
  safety_floor = TRUE,
  phylo_signal_gate = TRUE,
  phylo_signal_threshold = 0.2,
  phylo_signal_method = c("lambda", "blomberg_k"),
  min_val_cells = 20L,
  lambda_mode = c("fixed_1", "estimate", "cv", "bayes"),
  verbose = TRUE,
  seed = NULL
)
```

Arguments

data	object of class "pigauto_data".
tree	object of class "phylo".
splits	list (output of make_missing_splits) or NULL.
graph	list (output of build_phylo_graph) or NULL.
baseline	list (output of fit_baseline) or NULL.
hidden_dim	integer. Hidden layer width (default 64).
k_eigen	integer. Number of spectral node features (default 8).
n_gnn_layers	integer. Number of graph message-passing layers (default 2). Each layer has its own learnable alpha gate, layer normalisation, and ResNet-style skip connection.
gate_cap	numeric. Upper bound for the per-column blend gate (default 0.8). Safety comes from regularisation, not the cap.
use_attention	logical. Use attention in the GNN layers (default TRUE).
use_transformer_blocks	logical. Replace the legacy attention stack with pre-norm transformer-encoder blocks (multi-head attention • FFN + two residual skips). Default TRUE. Set FALSE to reconstruct pre-v0.9.0 fits (single-head attention with a learnable alpha gate per layer).

`n_heads` integer. Number of attention heads when `use_transformer_blocks = TRUE` (default `4`). Each head learns its own phylogenetic bandwidth (B2 rate-aware attention).

`ffn_mult` integer. Feed-forward width multiplier inside each transformer block (default `4`, giving `hidden_dim * 4`).

`use_trait_attention`
logical. Opt-in within-row cross-trait self-attention (B3, v0.9.3). When `TRUE` (default `FALSE`), the model builds per-trait tokens from each row's latent values (linear projection + learnable positional embedding), applies one multi-head self-attention block over the trait sequence, mean-pools to a `trait_embed_dim` feature, and concatenates it alongside `(x, coords, covs)` at the encoder input. Intended for trait sets with strong within-row functional coupling that the joint MVN / threshold-joint baseline cannot capture (e.g. nonlinear cross-trait structure). On the BIEN n=2000 plant bench it did **not** improve pooled RMSE (cross-trait covariance is already captured by the joint baseline); kept as an opt-in for datasets where it may help. Default `FALSE` preserves v0.9.2 behaviour exactly. Note: Ablation testing demonstrates that this within-row attention recovers only a small, sub-material gain, and only when cross-trait structure is explicitly non-linear and the joint-MVN baseline is present; it cannot model non-linear coupling on its own.

`n_trait_heads` integer. Number of attention heads in the within-row self-attention block when `use_trait_attention = TRUE`. Default `2`. Ignored when `use_trait_attention = FALSE`.

`trait_embed_dim`
integer. Embedding dim per trait token in the within-row self-attention block. Default `32`. Ignored when `use_trait_attention = FALSE`.

`dropout` numeric. Dropout rate (default `0.10`).

`lr` numeric. AdamW learning rate (default `0.003`).

`weight_decay` numeric. AdamW weight decay (default `1e-4`).

`epochs` integer. Maximum training epochs (default `3000`).

`corruption_rate`
numeric. Final corruption fraction if `corruption_ramp > 0`; otherwise the fixed corruption rate per epoch (default `0.55`).

`corruption_start`
numeric. Initial corruption fraction for the curriculum schedule (default `0.20`). Ignored if `corruption_ramp = 0`.

`corruption_ramp`
integer. Epochs over which corruption linearly ramps from `corruption_start` to `corruption_rate` (default `500`). Set to `0` for fixed corruption.

`refine_steps` integer. Iterative refinement steps at inference (default `8`).

`lambda_shrink` numeric. Weight on the shrinkage penalty `||delta - baseline||^2` that keeps the GNN correction close to the phylogenetic baseline (default `0.03`).

`lambda_gate` numeric. Weight on the gate regularisation penalty that pushes learnable gates toward zero. Prevents gates from staying open when the GNN provides no useful correction (default `0.01`).

warmup_epochs integer. Linear learning-rate warmup over the first N epochs (default 200). After warmup, a cosine schedule decays the LR to 1e-5.

edge_dropout numeric. Fraction of adjacency edges randomly zeroed each training epoch for graph regularisation (default 0.1). Set to 0 to disable.

eval_every integer. Evaluate on val every N epochs (default 100).

patience integer. Early-stopping patience in eval cycles (default 10).

clip_norm numeric. Gradient clip norm (default 1.0).

conformal_method
character. How the conformal prediction score is estimated from validation residuals. "split" (default, backward-compatible) takes a single sample quantile; "bootstrap" takes conformal_bootstrap_B bootstrap resamples of the val residuals and averages the per-resample quantiles. Empirically the bootstrap variant reduces the conformal-score variance across seeds by ~30% at small n_val, but on the simulation design used to evaluate it (n=150 species, 35\ into better 95\ across 10 seeds). Ship as an opt-in experimental knob; defaults to "split".

conformal_bootstrap_B
integer. Bootstrap resamples used when conformal_method = "bootstrap"; default 500. Ignored otherwise.

conformal_split_val
logical. Default FALSE (pre-2026-04-28 single-set behaviour, retained because forcing the split regresses the AVONET300 / OVR-categorical / BIEN safety-floor smoke benches by 2-26%). When TRUE, the validation set is split per latent column into a calibration half (used to pick the calibrated blend gate) and a conformal half (used to compute the conformal residual quantile). This restores split-conformal exchangeability — without the split, the gate is selected to minimise residual MSE on the very cells whose residuals drive the conformal quantile, producing systematic undercoverage (most visible at small n_val; see the coverage_investigation memo). Use this when accurate 95% coverage matters more than the bench-grade RMSE; the split only activates per-column when that column has at least 2 * min_val_cells val cells (smaller columns silently fall back to the single-set path to keep calibrate_gates()'s half-A / half-B cross-check stable).

gate_method character. How the per-trait calibrated gate is chosen. "single_split" (default, backward-compatible) runs the grid search on a single random half-A / half-B split of the val rows; "median_splits" repeats the whole procedure for gate_splits_B random splits and takes the median best_g. "cv_folds" (2026-04-30) partitions val cells into gate_cv_folds (default 5) deterministic non-overlapping folds and runs the grid + half-B-verify procedure once per fold (training set = K-1 folds, held-out = remaining fold), taking the componentwise median of K winning weight vectors. "cv_folds" uses larger training sets per split (K-1/K vs 1/2 in median_splits) and has a standard cross-validation interpretation, motivated by the open val→test drift observed on 4/32 binary cells in the discrete-bench memo. "median_splits" slightly reduces gate bimodality at small n_val (SD 0.406 → 0.360 across 10 seeds on the evaluation sim) with a small coverage-SD improvement (0.094 → 0.086). Negligible runtime cost (B × cheap grid searches).

`gate_splits_B` integer. Random splits used when `gate_method = "median_splits"`; default `31` (odd so the median is well-defined).

`gate_cv_folds` integer. Number of CV folds when `gate_method = "cv_folds"`; default `5`, must be `>= 2`. Capped at `n_val` per trait so each fold has at least 1 cell. When effective K < 2 (e.g. `n_val = 1`), the code falls back to a single split.

`safety_floor` logical. When `TRUE` (default), post-training calibration searches a 3-way simplex of BM, GNN, and grand-mean candidates. Because the grand-mean corner is always in the grid, the selected candidate cannot be worse than that corner on the validation cells under the calibration metric. When `FALSE`, the v0.9.1 1-D calibration is used exactly (`r_MEAN = 0`).

`phylo_signal_gate` logical. When `TRUE` (default since v0.9.1.9003), compute per-trait Pagel's λ on training-observed cells before fitting; for traits with `lambda < phylo_signal_threshold`, force `(r_cal_bm = 0, r_cal_gnn = 0, r_cal_mean = 1)` directly and skip BM + GNN training on those traits. Requires the `phytools` package. Falls back to safety-floor-only behaviour (`phylo_signal_gate = FALSE` effective) when `phytools` is absent.

`phylo_signal_threshold` numeric, default `0.2`. Traits with Pagel's λ below this value are routed to the grand-mean corner of the safety-floor simplex.

`phylo_signal_method` character, currently only `"lambda"` is fully implemented. Reserved `"blomberg_k"` path returns Blomberg's K via `phytools::phylosig()` but uses the same threshold — which is NOT dimensionally comparable; users selecting K must supply a K-appropriate threshold.

`min_val_cells` integer. Warn at fit time if any trait has fewer than `min_val_cells` validation cells available for gate calibration and conformal-score estimation. Default `10`: the floor of pathological territory, where the conformal quantile collapses to `max(val_residuals)` and gate calibration becomes essentially a coin flip between `0` and `gate_cap`. Recommended operational target is `n_val >= 20-30` per trait; achieve this by increasing `missing_frac` or collecting more species. See **Calibration at small n** below.

`lambda_mode` character. Pagel-lambda mode for the BM baseline. `"fixed_1"` preserves the default Brownian correlation matrix; `"estimate"`, `"cv"`, and `"bayes"` delegate lambda handling to the per-column BM path. Passed to `fit_baseline` and stored in the fitted model config.

`verbose` logical. Print training progress (default `TRUE`).

`seed` optional integer. When supplied, makes stochastic training and calibration reproducible; the default `NULL` uses the current RNG stream.

Details

Blend formulation: The prediction is $\hat{x} = (1 - r)\mu + r\delta$, where μ is the BM baseline, δ is the model's direct prediction, and $r = \sigma(\rho) \times \text{cap}$ is a per-column learnable gate bounded in $(0, \text{gate\_cap})$. When $r = 0$, the prediction collapses to the baseline. The gate is regularised toward zero via the shrinkage penalty on $\delta - \mu$, so the model defaults to the baseline unless the GNN's correction demonstrably helps on the validation set.

Training objective (per epoch):

1. A random subset of observed cells is corrupted with a learnable mask token.
2. The model predicts δ from graph context.
3. Loss = type-specific reconstruction on corrupted cells + `lambda_shrink` * MSE($\delta - \mu$) + `lambda_gate` * MSE(r).

The gate penalty on r is necessary because when $\delta = \mu$ (the BM-optimal solution for observed cells), the reconstruction and shrinkage losses both equal zero regardless of r, leaving no gradient to close the gate. The explicit penalty ensures gates default toward zero when the GNN correction provides no benefit.

Type-specific losses:

continuous/count/ordinal MSE

binary BCE with logits

categorical cross-entropy over K latent columns

Value

An object of class `"pigauto_fit"`.

Calibration at small n

pigauto's 95\ interval half-width for each trait is the empirical `(1 - alpha)` quantile of $|y - \hat{y}|$ on held-out validation cells. When the number of validation cells per trait (`n_val`) is large ($\gtrsim 30$), split-conformal gives near-exact marginal coverage under mild exchangeability assumptions.

At small `n_val` (< 20, and especially < 10) two things degrade at once:

- The conformal quantile clamps to `max(residuals)` because the required quantile level exceeds 1. The score is the single largest val residual and has substantial sampling variance across fits.
- The gate calibration's half-A / half-B split ends up with only a few cells per half, so the grid-search winner is essentially random between `0` and `gate_cap`.

Empirically (pigauto simulation harness, n=150 species, 35\ trait_MAR, 10 random seeds), default behaviour produces 92\ coverage with per-fit coverage ranging `[0.73, 1.00]`. The mean is close to the 95\ `gate_method = "median_splits"` and `conformal_method = "bootstrap"` reduce their respective estimator variances but empirically do not meaningfully narrow the coverage distribution. **Treat the 95\ regime**; increase `missing_frac` (more held-out data for calibration) or collect more species if tight per-fit coverage is required.

The `min_val_cells` warning fires at fit time whenever any trait falls below this threshold, so you know when to interpret the intervals cautiously.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
data <- preprocess_traits(traits, tree)
splits <- make_missing_splits(data$X_scaled, trait_map = data$trait_map,
                              seed = 1)
fit <- fit_pigauto(data, tree, splits = splits, epochs = 5L,
                   verbose = FALSE, seed = 1)
```

`impute` *Impute missing phylogenetic traits (convenience wrapper)*

Description

One-call interface to the full pigauto pipeline: preprocessing, baseline fitting, GNN training, and prediction. For fine-grained control, use the individual functions (`preprocess_traits`, `fit_baseline`, `fit_pigauto`, etc.) directly.

Usage

```
impute(
  traits,
  tree,
  species_col = NULL,
  trait_types = NULL,
  multi_proportion_groups = NULL,
  log_transform = TRUE,
  missing_frac = 0.25,
  n_imputations = 1L,
  covariates = NULL,
  epochs = 2000L,
  verbose = TRUE,
  seed = NULL,
  multi_obs_aggregation = c("hard", "soft"),
  lambda_mode = c("fixed_1", "estimate", "cv", "bayes"),
  em_iterations = 0L,
  em_tol = 0.001,
  em_offdiag = FALSE,
  pool_method = c("median", "mean", "mode"),
  clamp_outliers = FALSE,
  clamp_factor = 5,
  match_observed = c("none", "pmm"),
  pmm_K = 5L,
```

```
  safety_floor = TRUE,
  phylo_signal_gate = TRUE,
  phylo_signal_threshold = 0.2,
  phylo_signal_method = "lambda",
  ...
)
```

Arguments

traits	data.frame with species as rownames and trait columns, or (when species_col is supplied) a data.frame with a species column that may have multiple rows per species. Supported column types: numeric (continuous), integer (count), factor (binary/categorical), ordered (ordinal), character (factor → binary/categorical), logical (binary). See the **Trait type auto-detection** section below.
tree	object of class "phylo".
species_col	character. Name of the column in traits that identifies species. When supplied, multiple observations per species are supported. Default NULL uses row names (one row per species).
trait_types	named character vector overriding the auto-detected type for specific trait columns, e.g. c(Survival = "proportion", Parasites = "zi_count"). Required for the two types that cannot be inferred from R class (see **Trait type auto-detection** below). Default NULL (auto).
multi_proportion_groups	named list declaring compositional (multi_proportion) traits, e.g. list(colour = c("black", "blue", "red", "yellow")). Each list element names a group and gives the K trait columns that form a simplex (rows summing to 1). Encoded via CLR + per-component z-score. Multi_proportion traits *cannot* be declared through trait_types — use this argument instead. Default NULL (no multi_proportion groups).
log_transform	logical. Auto-log positive continuous columns (default TRUE).
missing_frac	numeric. Fraction of observed cells held out for validation/test evaluation (default 0.25). Set to 0 to skip splitting (all cells used for training, no evaluation).
n_imputations	integer. Number of MC-dropout imputation sets (default 1). Values > 1 enable between-imputation uncertainty.
covariates	data.frame or matrix of environmental covariates (fully observed — no NAs). Covariates are conditioners: they inform imputation but are not themselves imputed. Numeric/integer columns are z-scored; factor/ordered columns are one-hot encoded automatically. If a variable has missing values, include it in traits instead. Same number of rows as traits. Default NULL (no covariates).
epochs	integer. Maximum GNN training epochs (default 2000).
verbose	logical. Print progress (default TRUE).
seed	optional integer. When supplied, makes the stochastic fitting and prediction steps reproducible; the default NULL uses the current RNG stream.
multi_obs_aggregation	character. How to aggregate multiple observations per species before the Level-C baseline. "hard" (default) thresholds binary proportions at 0.5 and uses

argmax for categorical. "soft" preserves species-level proportions and dispatches a soft E-step so that intermediate class frequencies contribute fractional liability evidence. Passed to fit_baseline.

lambda_mode — character. Pagel-lambda mode for the BM baseline. "fixed_1" preserves the default Brownian correlation matrix; "estimate", "cv", and "bayes" delegate lambda handling to the per-column BM path. Passed to fit_baseline and stored in the fitted model config.

em_iterations — integer. Phase 6 EM iterations for the threshold-joint baseline (binary + ordinal + OVR categorical). Default 0L preserves v0.9.1 behaviour byte-for-byte. When >= 2L, the BM rate Σ learned by Rphylopars::phylopars() at iteration k is fed back as the per-trait prior SD at iteration $k + 1$, up to em_iterations times or until em_tol convergence. Passed to fit_baseline.

em_tol — numeric. Relative-Frobenius convergence tolerance for the Phase 6 / 7 EM loop. Default 1e-3.

em_offdiag — logical. Phase 7 opt-in: when TRUE AND em_iterations >= 2L, each liability cell's prior uses the full conditional-MVN from Σ's off-diagonal entries, so that observing one discrete trait shifts (not just tightens) the prior on correlated other traits. Binary + ordinal only; OVR categorical stays on Phase 6 diagonal. Default FALSE. Passed to fit_baseline.

pool_method — character. How to pool multiple imputation draws (n_imputations > 1) for count, proportion, and zi_count magnitude traits: "median" (default) takes the per-cell median of the M decoded draws — robust to dropout-noisy latents amplified by expm1() / plogis() decoders. "mean" restores the pre-v0.9.2 arithmetic-mean pooling. "mode" (Phase H, v0.9.1.9010+) is intended for ordinal traits: per-cell majority vote across the M draws, avoiding the integer-mean-round bias toward middle classes. For continuous-family traits, "mode" falls back to "median". Binary / categorical / multi_proportion traits always pool by probability average; unaffected by this argument. See issue #40.

clamp_outliers — logical. Phase G (v0.9.1.9011+). When TRUE, post-back-transform predictions for log-transformed continuous, count, and zi_count magnitude traits are capped at tm$obs_max * clamp_factor (and tm$obs_max is the observed maximum on the original scale, recorded at preprocess time). Targets the AVONET Mass tail-extrapolation mode documented in useful/MEMO_2026-05-01_avonet_mass_diag.md where a $+3$-4σ latent overshoot becomes a 50x-100x value error after expm1(). Default FALSE preserves v0.9.1 behaviour exactly.

clamp_factor — numeric scalar (>= 1). Multiplicative factor on the observed maximum used by clamp_outliers. Default 5 (Tukey-style outlier definition: anything >= 5x the observed max is implausible). Ignored when clamp_outliers = FALSE.

match_observed — character, one of c("none", "pmm"). Phase G' (v0.9.1.9012+). Pass-through to predict.pigauto_fit. When "pmm", uses Predictive Mean Matching for log-transformed continuous, count, zi_count magnitude, and proportion traits: imputed values are drawn from the observed value pool, never extrapolated.

When to use: PMM is a niche feature. pigauto already provides conformal prediction intervals calibrated against held-out residuals and stochastic prediction draws for diagnostics. PMM is only worth enabling for: (a) methodological comparison against mice, or (b) workflows that specifically require imputed values to come from the observed data pool. For tail safety, prefer clamp_outliers

= TRUE. PMM failed the downstream fixed-effect redesign pilot and is unsupported for inference; see multi_impute_analysis() for the analysis-aware backend.

Default "none" preserves pre-G' behaviour.

pmm_K — integer (>= 1). Donor pool size for PMM. Default 5L (mice convention). Ignored when match_observed = "none".

safety_floor — logical. When TRUE (default since v0.9.1.9002), calibration searches the 3-way simplex r_BM * BM + r_GNN * GNN + r_MEAN * MEAN so the grand mean is always in the candidate set. Under the validation metric used for calibration, the selected candidate cannot be worse than that grand-mean corner on the validation cells. This is a validation safeguard, not a guarantee about future held-out data. When FALSE, the v0.9.1 1-D calibration is used exactly. See the Safety floor section below.

phylo_signal_gate, phylo_signal_threshold, phylo_signal_method — Pass-through to fit_pigauto(). See that help page for details.

... — additional arguments passed to fit_pigauto.

Value

An object of class "pigauto_result" with components:

completed The input traits data.frame with observed values preserved and only missing cells filled in. This is the primary output – typically what users want.

imputed_mask Logical matrix (same shape as completed) that is TRUE for cells that were imputed (originally NA) and FALSE for observed cells.

prediction A pigauto_pred object from predict.pigauto_fit containing raw model predictions for every cell (observed + missing), standard errors, class probabilities, and conformal intervals.

fit The trained pigauto_fit object.

baseline The phylogenetic baseline.

data The preprocessed pigauto_data object.

splits The val/test splits (or NULL if missing_frac = 0).

evaluation Evaluation metrics on test set (or NULL).

Trait type auto-detection

pigauto infers each trait's type from its R class — no trait_types argument is needed for most data:

R class	pigauto type
numeric	continuous (auto-log if all-positive)
integer	count
factor with 2 levels	binary
factor (unordered) with >2 levels	categorical
ordered / factor(..., ordered = TRUE)	ordinal
character	→ factor → binary or categorical

`logical`	binary

Two types cannot be inferred from class alone and *must* be declared via `trait_types`:

`"proportion"` A `numeric` bounded 0–1, e.g. survival rate: `trait_types = c(Survival = "proportion")`.

`"zi_count"` An `integer` with excess zeros, e.g. parasite count: `trait_types = c(Parasites = "zi_count")`.

Use the `trait_types` argument directly (it is an explicit parameter, not a `...` pass-through).

Traits vs covariates

The distinction is **functional, not ontological**: a trait is something you want to impute (NA values allowed in `traits`); a covariate is something you use to sharpen imputation accuracy (must be fully observed, passed via `covariates`). The same variable can be either depending on the scientific question.

Examples:

- **IUCN status with Data Deficient species** → put it in `traits` as `ordered(c("LC","NT","VU","EN","CR"))` so pigauto predicts the unknown categories.
- **IUCN status fully known for all species** → pass as a covariate to inform imputation of other traits (e.g. body mass, range size).
- **Realm / biome (factor)** → pass as a covariate; pigauto one-hot encodes factor columns automatically (v0.6.1+).

Variables that belong in `traits`: anything with missing values you care about predicting. Variables that belong in `covariates`: fully observed, exogenous to the trait space (geography, climate, habitat, experimental treatment).

Safety floor (v0.9.1.9002+)

With `safety_floor = TRUE` (the new default), the post-training calibration grid searches a 3-way convex combination of the Brownian-motion baseline, the GNN delta, and the per-trait grand mean. The simplex is sampled at step 0.05 (231 candidates per latent column). Because the corner `(0, 0, 1)` — pure grand mean — is always in the grid, the selected candidate cannot be worse than the grand-mean corner on the validation cells under the calibration metric. The fit object gains four new slots: `r_cal_bm`, `r_cal_gnn`, `r_cal_mean` (each a named numeric of length `p_latent`), and `mean_baseline_per_col`.

Set `safety_floor = FALSE` to reproduce the pre-v0.9.1.9002 1-D calibration bit-identically (no mean term; `r_cal_mean = 0`; `r_cal_bm = 1 - r_cal_gnn`). See `specs/2026-04-23-safety-floor-mean-gate-design.md` for the design rationale and `plans/2026-04-23-safety-floor-mean-gate.md` for the implementation plan.

What gets imputed (read this first)

pigauto only *imputes* cells that are NA in the input. Observed cells are preserved as-is in `result$completed`. The slot `result$prediction$imputed` contains the model's prediction for **every** cell – observed and missing alike – and is intended for diagnostics (e.g. checking calibration on training cells), *not* as the imputed-values output. The imputed values themselves are `result$completed[result$imputed_mask]`.

Common pitfall. If you call `impute()` on a fully observed trait matrix (no NAs anywhere), there is nothing to impute. `result$completed` is identical to the input, `sum(result$imputed_mask)` is `0`, and `result$prediction$imputed` is just model predictions for already-known values. This is the right behaviour, but it can look surprising: e.g. on `avonet300` (fully observed), the "imputed" Mass values you see are simply the observed body masses passed through (some bird species are 24 kg). To exercise the imputation path on a complete dataset, mask some cells first (see Examples).

Imbalanced K-class traits. At default settings (`n_imputations = 1L`, `pool_method = "median"`), a small ordinal / categorical trait whose marginal distribution is heavily skewed (e.g. AVONET `Migration` is ~78\ ~14\ collapse onto a corner that predicts the majority class everywhere. When this matters, increase `n_imputations` (>= 20 in our K=3 ordinal benches) and set `pool_method = "mode"` (Phase H, +6.6 pp on AVONET Migration K=3 vs the default median pool). See `useful/MEMO_2026-05-01_phase_h`

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
df <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                c("Mass", "Wing.Length"), drop = FALSE]
rownames(df) <- tree$tip.label
df$Mass[seq_len(3L)] <- NA_real_
result <- impute(df, tree, epochs = 5L, verbose = FALSE)
head(result$completed)
```

---

load_pigauto *Load a saved pigauto model*

Description

Reads a `pigauto_fit` object previously saved with `save_pigauto`.

Usage

```
load_pigauto(path)
```

Arguments

`path`	character. File path to load from.

Value

An object of class `"pigauto_fit"`.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
traits$Mass[seq_len(3L)] <- NA_real_
fit <- impute(traits, tree, epochs = 5L, verbose = FALSE)$fit
path <- tempfile(fileext = ".pigauto")
save_pigauto(fit, path)
pred <- predict(load_pigauto(path))
```

`make_missing_splits` *Split cells into train/val/test for imputation evaluation*

Description

Randomly designates a fraction of cells as "missing" and splits them into validation and test sets. When a `trait_map` is supplied, masking operates at the **original trait level** – all latent columns belonging to one trait are held out together (important for categorical traits).

Usage

```
make_missing_splits(
  X,
  missing_frac = 0.25,
  val_frac = 0.25,
  seed = NULL,
  trait_map = NULL,
  mechanism = c("MCAR", "MAR_trait", "MAR_phylo", "MNAR"),
  mechanism_args = list(),
  tree = NULL
)
```

Arguments

`X`	numeric matrix (species x latent columns from `preprocess_traits`). Used only for dimensions.
`missing_frac`	numeric. Fraction of all (species, trait) cells to designate as missing (default `0.25`).
`val_frac`	numeric. Fraction of missing cells for validation (default `0.25`); the rest become the test set.
`seed`	optional integer. When supplied, makes split construction reproducible; the default `NULL` uses the current RNG stream.
`trait_map`	list of trait descriptors (from `pigauto_data`). If `NULL`, masking is applied per latent column (v0.1 behaviour).

`mechanism` character. Missingness mechanism: `"MCAR"` (default, uniform random), `"MAR_trait"` (trait-dependent), `"MAR_phylo"` (clade-structured), or `"MNAR"` (value-dependent).

`mechanism_args` named list of mechanism-specific parameters:

For `"MAR_trait"`: `driver_col` (integer, column index in `X` that drives missingness; default 1), `beta` (numeric, severity; default 2.0).

For `"MAR_phylo"`: `n_clades` (integer, number of high-missingness clades; default 2), `p_clade` (numeric, within-clade missingness probability; default 0.7), `p_base` (numeric, background missingness probability; default 0.1).

For `"MNAR"`: `beta` (numeric, severity; default 2.0).

`tree` object of class `"phylo"`. Required for `mechanism = "MAR_phylo"`, ignored otherwise.

Details

The returned index vectors use linear (column-major) indexing. Both original-trait-space and latent-space indices are returned when a `trait_map` is present.

Value

A list with:

val_idx Integer vector of linear indices (latent space).

test_idx Integer vector of linear indices (latent space).

val_idx_trait Integer vector in original-trait space (if `trait_map` supplied).

test_idx_trait Integer vector in original-trait space (if `trait_map` supplied).

n Number of species (rows).

p Number of latent columns.

n_traits Number of original traits.

mask Logical matrix (n x p_latent). `TRUE` = observed.

mechanism Character string of the mechanism used.

Examples

```
X <- matrix(rnorm(100), nrow = 20)
splits <- make_missing_splits(X, missing_frac = 0.25, seed = 1)
length(splits$val_idx)

# MAR: missingness depends on another trait
splits_mar <- make_missing_splits(X, mechanism = "MAR_trait",
  mechanism_args = list(driver_col = 1, beta = 2))
```

mask_missing *Create an observed/missing mask matrix*

Description

Returns a logical matrix of the same dimensions as `X`, with `TRUE` where values are observed (not `NA`).

Usage

```
mask_missing(X)
```

Arguments

`X`	numeric matrix (species x traits or species x latent columns).

Value

Logical matrix, `TRUE` = observed.

Examples

```
X <- matrix(c(1, NA, 3, 4), nrow = 2)
mask_missing(X)
```

multi_impute *Generate experimental stochastic completion datasets*

Description

Run pigauto's full imputation pipeline and return `M` stochastic completions of the trait matrix instead of a single point estimate. The conformal-width and Brownian/MC-dropout draws returned here are experimental prediction-diagnostic draws. A 3,000-fit known-DGP campaign found that neither method passed any of the 12 downstream fixed-effect gate cells. Do not use these datasets for downstream inference or Rubin pooling. A separate analysis-aware backend, `multi_impute_analysis()`, has passed its package-level fixed-effect gate for a narrow set of analyses.

Usage

```
multi_impute(
  traits,
  tree,
  m = 100L,
  draws_method = c("conformal", "mc_dropout"),
  species_col = NULL,
  trait_types = NULL,
```

```
  multi_proportion_groups = NULL,
  log_transform = TRUE,
  missing_frac = 0.25,
  covariates = NULL,
  epochs = 2000L,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

traits	data.frame with species as rownames and trait columns. Same input format as impute(). Supported column types are numeric, integer, factor, ordered factor, and logical.
tree	object of class phylo aligned with traits.
m	integer. Number of stochastic completion datasets to generate (default 100). Observed cells are identical across all M datasets; only originally-missing cells vary.
draws_method	character. How stochastic draws are generated for missing cells. One of: "conformal" (default) Run the model once, then sample each originally-missing cell from a Normal distribution centred on the point estimate with SD = conformal_score / 1.96. Converting a split-conformal residual quantile to a Normal scale is a heuristic; the conformal coverage guarantee does not establish that these draws are proper multiple imputations. Falls back to BM-SE-based Normal sampling when conformal scores are unavailable, and to Bernoulli / Categorical draws for discrete traits. "mc_dropout" Run M stochastic GNN forward passes in training mode (dropout active) on top of stochastic Brownian-motion baseline draws. Brownian draws still contribute between-draw variation when a calibrated GNN gate is zero.
species_col	character or NULL. If set, marks the column in traits containing species identifiers and enables multiple observations per species. See impute() for details.
trait_types	named character vector overriding auto-detected trait types for specific columns. Required for "proportion" and "zi_count". See impute() and preprocess_traits(). Default NULL (auto-detect).
multi_proportion_groups	named list declaring compositional trait groups (rows summing to 1), e.g. list(diet = c("plant", "invert", "vert")). Forwarded to impute() / preprocess_traits(). Default NULL.
log_transform	logical. Auto-log positive continuous columns (default TRUE).
missing_frac	numeric. Fraction of observed cells held out for validation/test during training (default 0.25). Passed through to impute().
covariates	data.frame or matrix of environmental covariates (fully observed, numeric). Passed through to impute(). Default NULL (no covariates).
epochs	integer. Maximum GNN training epochs (default 2000).

verbose	logical. Print progress (default TRUE).
seed	optional integer. When supplied, makes fitting and imputation draws reproducible; the default NULL uses the current RNG stream.
...	additional arguments forwarded to fit_pigauto() via impute(). See fit_pigauto() for the full list; the "Safety floor" section below describes the relevant new v0.9.1.9002 argument.

Details

These draws do not condition on a declared substantive analysis model. Consequently, stochastic variation alone does not make them proper or congenial multiple imputations. The analysis-aware backend requires the analysis model before generating draws and dispatches only across its documented supported model classes.

`draws_method = "conformal"` **(default)**: Run the model once; missing cells are sampled from $x_{ij}^{(k)} \sim \mathrm{N}(\hat{\mu}_{ij},\ q_j/1.96)$ where q_j is the trait-level split-conformal residual quantile. Dividing this quantile by 1.96 is a pragmatic Normal-scale construction, not a consequence of the conformal coverage guarantee. For discrete traits (binary, categorical) it uses Bernoulli / categorical draws from the estimated probability vector. For `multi_proportion` groups it draws the K CLR latent columns with their BM latent SEs, projects back to sum-zero CLR space, and decodes to the simplex.

`draws_method = "mc_dropout"`: Run `M` GNN forward passes in training mode (dropout active) on top of stochastic BM baseline draws. When `r_cal = 0`, the GNN-dropout term disappears but the BM draw still contributes between-draw variance.

Nakagawa & Freckleton (2008, 2011) review the consequences of ignoring missing data in ecological and comparative analyses and argue for multiple imputation as the default.

Value

An object of class `"pigauto_mi"` with components:

`datasets` A list of length `m`. Each element is a data.frame with the same shape and column types as the input `traits`; observed cells are preserved and missing cells are filled with the corresponding stochastic draw. These datasets are for prediction diagnostics, not downstream inference.

`m` Number of stochastic completion datasets.

`pooled_point` A single data.frame whose missing cells are replaced by the MC-averaged point estimate. Convenient for reporting but does *not* provide a valid downstream MI analysis.

`se` Matrix of per-cell uncertainty summaries combining the baseline SE and the between-draw standard deviation.

`imputed_mask` Logical matrix; `TRUE` where a cell was originally missing.

`fit` The underlying `pigauto_fit` object, retained for diagnostics and for calls to `predict()` on new data.

`data` The `pigauto_data` object.

`tree` The input phylogeny.

`species_col` Passed-through species-column name or `NULL`.

When to use this

This function is useful for comparing stochastic prediction behavior from one tree. It is not the analysis-aware inferential backend.

`multi_impute_trees()` provides an experimental posterior-tree sensitivity path, but tree uncertainty is not supported by `multi_impute_analysis()`.

Safety floor (v0.9.1.9002+)

When `fit_pigauto()` was called with `safety_floor = TRUE` (the default since v0.9.1.9002), the 3-way blend `r_BM * BM + r_GNN * GNN + r_MEAN * MEAN` propagates through every imputation draw automatically via the updated `predict.pigauto_fit()`. For `draws_method = "mc_dropout"` the mean term contributes no between-draw variance (it is a deterministic scalar per column); between-draw variance comes from the BM-draw and GNN-dropout terms only. For `draws_method = "conformal"` the blend centre is the 3-way prediction and conformal scores remain calibrated on the blended residuals.

References

Rubin DB (1987). *Multiple Imputation for Nonresponse in Surveys.* Wiley.

Nakagawa S, Freckleton RP (2008). "Missing inaction: the dangers of ignoring missing data." *Trends in Ecology & Evolution* 23(11): 592-596.

Nakagawa S, Freckleton RP (2011). "Model averaging, missing data and multiple imputation: a case study for behavioural ecology." *Behavioral Ecology and Sociobiology* 65(1): 103-116.

See Also

`impute()` for point imputation and `multi_impute_analysis()` for the narrow analysis-aware inferential backend.

Examples

```
library(pigauto)
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
df <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                c("Mass", "Wing.Length"), drop = FALSE]
rownames(df) <- tree$tip.label
df$Mass[seq_len(3L)] <- NA_real_
mi <- multi_impute(df, tree, m = 2L, epochs = 5L, verbose = FALSE)
print(mi)
lapply(mi$datasets, head)
```

`multi_impute_analysis` *Analysis-aware multiple imputation for narrow regression models*

Description

Generate multiple imputations conditional on a specified downstream regression model. Validity requires MAR, a correctly specified conditional imputation model, and adequate SMCFCS or jomo convergence. This experimental interface is deliberately narrow: exactly one continuous covariate may contain missing values, while the outcome, all other predictors, auxiliary variables, and any grouping variable must be fully observed.

Usage

```
multi_impute_analysis(
  data,
  formula,
  missing,
  model = c("lm", "glm", "lmer"),
  m = 50L,
  auxiliary = character(),
  seed = NULL,
  control = list()
)
```

Arguments

`data`	A data frame containing the analysis variables.
`formula`	A two-sided additive model formula. For `model = "lmer"`, exactly one random-intercept term of the form `(1 | group)` is required.
`missing`	A single character string naming the continuous numeric covariate to impute. It must occur as a main-effect predictor in `formula` and must be the only column in `data` containing missing values.
`model`	One of `"lm"`, `"glm"`, or `"lmer"`. The `"glm"` route is restricted to binomial-logit regression. The `"lmer"` route is restricted to a Gaussian random-intercept model.
`m`	Integer number of completed datasets. Must be at least two.
`auxiliary`	Character vector naming fully observed numeric columns used only by the imputation model. They cannot duplicate formula variables. Derived terms must be created explicitly in `data` before calling this function.
`seed`	Optional integer random seed. When supplied, makes the imputation draws reproducible; the default `NULL` uses the current RNG stream.
`control`	Named list of engine controls. The `"lm"` route accepts no controls. The `"glm"` route accepts `numit` (default 20) and `rjlimit` (default 100000). The `"lmer"` route accepts `nburn` (default 1000) and `nbetween` (default 100).

Value

An object with classes "pigauto_analysis_mi" and "pigauto_mi". Its datasets component is a list of m completed data frames compatible with with_imputations() and pool_mi(). The object also records the analysis model, formula, engine, controls, engine version, seed, runtime provenance, and imputed-variable metadata. Any engine warning aborts the call instead of returning an inference-ready object.

Experimental scope

This function does not fit pigauto's phylogenetic baseline or graph neural network and does not propagate posterior-tree uncertainty. The "lm" route uses proper Bayesian Normal-regression draws. The "glm" route requires the optional smcfcs package, and the "lmer" route requires the optional jomo and lme4 packages. Interactions, transformed terms, random slopes, nested or crossed random effects, multiple incomplete variables, missing outcomes, and pooling of variance components or BLUPs are unsupported.

See Also

multi_impute(), with_imputations(), pool_mi()

Examples

```
set.seed(11)
dat <- data.frame(
  y = stats::rnorm(80),
  x = stats::rnorm(80),
  z = stats::rnorm(80)
)
dat$x[seq(4, 80, by = 5)] <- NA_real_
mi <- multi_impute_analysis(
  dat, y ~ x + z, missing = "x", model = "lm", m = 5, seed = 12
)
fits <- with_imputations(
  mi, function(d) stats::lm(y ~ x + z, data = d), .progress = FALSE
)
pool_mi(fits)
```

---

multi_impute_trees *Posterior-tree prediction sensitivity*

Description

Run pigauto's full imputation pipeline on each of T posterior phylogenies, generating m_per_tree stochastic completions per tree for a total of T * m_per_tree completed datasets. Each completed dataset is conditional on a specific posterior tree (recorded in mi$tree_index).

Usage

```
multi_impute_trees(
  traits,
  trees,
  m_per_tree = 1L,
  species_col = NULL,
  trait_types = NULL,
  multi_proportion_groups = NULL,
  log_transform = TRUE,
  missing_frac = 0.25,
  covariates = NULL,
  epochs = 2000L,
  verbose = TRUE,
  seed = NULL,
  share_gnn = TRUE,
  reference_tree = NULL,
  ...
)
```

Arguments

traits	data.frame. Same format as multi_impute() and impute().
trees	list of phylo objects (class multiPhylo or plain list). Each tree must contain the species in traits as tips. Posterior samples from BirdTree.org (Jetz et al. 2012) are ideal; the bundled trees300 dataset provides 50 posterior trees for avonet300.
m_per_tree	integer. Number of stochastic completion draws per tree (default 1). Total datasets = length(trees) * m_per_tree.
species_col	character or NULL. See impute().
trait_types	named character vector overriding auto-detected trait types. Required for "proportion" and "zi_count". See impute() / preprocess_traits(). Default NULL (auto-detect).
multi_proportion_groups	named list declaring compositional trait groups (rows summing to 1), forwarded to impute() / preprocess_traits(). Default NULL.
log_transform	logical. Auto-log positive continuous columns (default TRUE).
missing_frac	numeric. Fraction held out for validation/test during training (default 0.25).
covariates	data.frame or matrix of environmental covariates (fully observed, numeric). Passed through to impute(). Default NULL (no covariates).
epochs	integer. Maximum GNN training epochs per tree (default 2000).
verbose	logical. Print progress (default TRUE).
seed	optional integer. Base random seed; when supplied, each tree uses seed + t - 1 for reproducible results. The default NULL uses the current RNG stream.
share_gnn	logical. If TRUE (default), fit the GNN once on a reference tree and reuse it across all posterior trees, recomputing only the BM baseline per tree. Gives a ~10-15x speedup at n=10k. See the "Share-GNN" section below for tree-uncertainty

propagation details. Set FALSE to fit from scratch on every tree (the pre-v0.9.1 behaviour) when you need exact tree-by-tree model independence.

`reference_tree` optional phylo used as the training tree when share_gnn = TRUE. Default NULL selects the maximum-clade-credibility tree via phangorn::maxCladeCred(trees). If phangorn is not installed, falls back to trees[[1]] with a warning.

`...` additional arguments forwarded to fit_pigauto() via impute().

Details

This is an experimental prediction-sensitivity workflow. It compares stochastic completions across a posterior tree sample; it does not provide a validated downstream inferential workflow. See vignette("tree-uncertainty").

Every completed dataset carries a different tree's signal so that between-tree variation enters the returned stochastic completions. This path is not supported by multi_impute_analysis() and has not passed the analysis-aware inferential gate. Do not use it for downstream inference.

For each tree the function runs the full pigauto pipeline (preprocess -> baseline -> GNN -> predict) when share_gnn = FALSE. With the default share_gnn = TRUE, the GNN is trained once and only the baseline is recomputed per tree. Topologies and branch lengths vary across trees, so the phylogenetic baseline covariance differs for each tree.

The returned datasets may be compared descriptively to assess sensitivity of point imputations to the tree sample. No calibrated downstream standard error or Rubin-pooling claim is made for this path.

Computation time. With share_gnn = TRUE (default): one GNN fit

- T cheap baseline passes. Rough budget on a modern CPU laptop:

Species n	1 fit	T = 50 share_gnn=TRUE	T = 50 share_gnn=FALSE
300	~30-60 s	~3-5 min	25-50 min
5,000	~5-10 min	~10-20 min	4-8 hr
10,000	~20-40 min	~30-60 min	17-33 hr

Value

An object of class "pigauto_mi_trees", inheriting from "pigauto_mi", with components:

`datasets` List of T * m_per_tree completed data.frames. Observed cells are preserved; missing cells are filled with stochastic draws for prediction-sensitivity diagnostics.

`m` Total number of datasets (T * m_per_tree).

`n_trees` Number of posterior trees used.

`m_per_tree` Imputations per tree.

`tree_index` Integer vector of length m; element i gives the tree index (1..T) for dataset i.

`pooled_point` Single data.frame averaging across all T * m_per_tree datasets. For reporting, not inference.

`se` Matrix of per-cell pooled SEs (NA if not available).

`imputed_mask` Logical matrix; TRUE where a cell was originally missing.

share_gnn Logical; TRUE if the shared-GNN path was used.

fit Single pigauto_fit trained on the reference tree when share_gnn = TRUE; NULL otherwise.

fits List of T pigauto_fit objects (one per tree) when share_gnn = FALSE; NULL when share_gnn = TRUE.

reference_tree The reference phylo used for GNN training when share_gnn = TRUE; NULL otherwise.

trees The input posterior trees.

species_col Passed-through species column name.

When to use this

Choose the prediction-diagnostic function based on how many trees you have:

- **One tree** (single published phylogeny, single time-calibrated tree): use multi_impute() and select a draw method explicitly when needed.
- **Multiple posterior trees** (BirdTree samples, BEAST posterior, etc.): use multi_impute_trees() only for prediction sensitivity. Tree uncertainty is unsupported by the analysis-aware backend.

Share-GNN (tree-sharing) mode

Under share_gnn = TRUE the GNN weights and spectral features are trained once on the reference tree (MCC by default). For each posterior tree the BM / joint-MVN baseline is recomputed, and the prediction is the blend (1 - r_cal) * baseline_t + r_cal * gnn_shared. Because r_cal is calibrated once on held-out data at the reference tree and applied uniformly, the tree-uncertainty contribution is:

- Fully preserved when the gate is closed (r_cal near 0): the GNN contributes nothing, and the baseline varies per tree.
- Partially preserved when the gate is open: the baseline portion still varies, but the GNN portion is a tree-invariant constant — this slightly under-estimates tree variance in the GNN channel.
- Lost in the GNN channel when the gate is fully open (rare on real data; the baseline channel still carries tree variation).

On every real dataset benchmarked in the v0.9.0 campaign the gate closed partially or fully. This evidence is specific to those benchmark regimes and does not guarantee tree-variance calibration elsewhere. Set share_gnn = FALSE if you need exact per-tree model independence.

Safety floor + share_gnn (v0.9.1.9002+)

When share_gnn = TRUE with safety_floor = TRUE, the grand-mean baseline mean_baseline_per_col and the three calibrated weights (r_cal_bm, r_cal_gnn, r_cal_mean) are computed ONCE on the reference tree and reused across all posterior trees. They are properties of the observed training traits, not of the tree topology. Each posterior tree only recomputes the BM baseline; the GNN delta and the three weights stay fixed. This preserves prediction sensitivity to the per-tree baseline without re-calibrating the safety floor; it does not establish downstream inferential validity.

References

Nakagawa S, de Villemereuil P (2019). "A general method for simultaneously accounting for phylogenetic and species sampling uncertainty via Rubin's rules in comparative analysis." *Systematic Biology* 68(4): 632-641.

Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). "The global diversity of birds in space and time." *Nature* 491(7424): 444-448.

See Also

`multi_impute()` for single-tree prediction diagnostics, `multi_impute_analysis()` for analysis-aware MI, and trees300

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
trees <- structure(list(tree, tree), class = "multiPhylo")
df <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                c("Mass", "Wing.Length"), drop = FALSE]
rownames(df) <- tree$tip.label
df$Mass[seq_len(3L)] <- NA_real_
mi <- multi_impute_trees(df, trees, m_per_tree = 1L, epochs = 5L,
                         verbose = FALSE)
print(mi)
mass_by_tree <- vapply(mi$datasets, function(dat) dat$Mass,
                       numeric(nrow(df)))
apply(mass_by_tree, 1L, stats::sd)
```

---

pigauto_report *Generate an HTML benchmark report from a pigauto fit*

Description

Produces a self-contained HTML file with interactive charts comparing the GNN against the phylogenetic baseline. The report includes per-trait metrics, gate values, conformal coverage, and training history.

Usage

```
pigauto_report(
  fit,
  data = NULL,
  splits = NULL,
  output_path = "pigauto_report.html",
  title = "pigauto Imputation Report",
  open = TRUE
)
```

Arguments

fit	A pigauto_fit object (or a pigauto_result from impute).
data	Optional pigauto_data object. Extracted automatically when fit is a pigauto_result.
splits	Optional splits object. Extracted automatically when fit is a pigauto_result.
output_path	Character. File path for the HTML report (default "pigauto_report.html" in the working directory).
title	Character. Report title.
open	Logical. Open the report in a browser when done (default TRUE).

Value

The output path (invisibly).

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
traits$Mass[seq_len(3L)] <- NA_real_
result <- impute(traits, tree, epochs = 5L, verbose = FALSE)
pigauto_report(result, output_path = tempfile(fileext = ".html"),
               open = FALSE)
```

---

plot.pigauto_benchmark

Plot a pigauto benchmark

Description

Creates a multi-panel comparison plot showing BM baseline vs pigauto performance across all simulation scenarios.

Usage

```
## S3 method for class 'pigauto_benchmark'
plot(x, metric = "rmse", ...)
```

Arguments

x	pigauto_benchmark object.
metric	character. Which metric to plot (default "rmse").
...	passed to base plot functions.

Value

Invisible NULL.

Examples

```
benchmark <- structure(list(summary = data.frame(
  scenario = rep("BM", 2), trait = rep("trait1", 2),
  method = c("baseline", "pigauto"), metric = rep("rmse", 2),
  mean = c(1, 0.8)
)), class = "pigauto_benchmark")
plot(benchmark)
```

---

plot.pigauto_fit *Plot diagnostics for a fitted pigauto model*

Description

S3 plot method for pigauto_fit objects, using base R graphics. Three plot types are available: training history, calibrated gate values, and conformal prediction scores.

Usage

```
## S3 method for class 'pigauto_fit'
plot(x, type = "history", ...)
```

Arguments

x	An object of class "pigauto_fit".
type	Character. "history" (default): 2x2 panel of training loss components (reconstruction, shrinkage, gate regularisation) and validation loss over epochs. "gates": bar plot of calibrated gate values per trait, coloured by trait type (green = continuous, blue = count, orange = ordinal, red = binary, purple = categorical). "conformal": bar plot of conformal prediction scores per trait with a reference line at the median score.
...	Additional arguments passed to base plotting functions.

Value

Invisible NULL. Called for its side effect (plotting).

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
data <- preprocess_traits(traits, tree)
```

```
splits <- make_missing_splits(data$X_scaled, trait_map = data$trait_map)
fit <- fit_pigauto(data, tree, splits = splits, epochs = 5L,
                   eval_every = 1L, verbose = FALSE)
plot(fit)
plot(fit, type = "history")
plot(fit, type = "gates")
plot(fit, type = "conformal")
```

`plot.pigauto_pred` *Plot predictions from a pigauto model*

Description

S3 `plot` method for `pigauto_pred` objects, using base R graphics. Three plot types are available: scatter plots of observed vs predicted values, interval plots per species, and probability charts for discrete traits.

Usage

```
## S3 method for class 'pigauto_pred'
plot(x, data = NULL, splits = NULL, type = "scatter", trait = NULL, ...)
```

Arguments

`x`	An object of class `"pigauto_pred"`.
`data`	Optional numeric matrix or data.frame of observed values in original scale. When supplied, observed values are overlaid on scatter and interval plots.
`splits`	Optional list (output of `make_missing_splits`). Used to identify which cells were held out for evaluation.
`type`	Character. `"scatter"` (default): grid of observed-vs-predicted scatter plots for continuous, count, and proportion traits, with 1:1 line and conformal interval bands where available. `"intervals"`: for each trait, species sorted by predicted value with conformal-interval ribbon; observed values shown in red where available. `"probabilities"`: for binary traits, boxplot of predicted probabilities grouped by true class; for categorical traits, stacked bar chart of mean predicted probabilities by true class.
`trait`	Character vector of trait names to plot. If `NULL` (default), all traits appropriate for the chosen `type` are plotted.
`...`	Additional arguments passed to base plotting functions.

Value

Invisible `NULL`. Called for its side effect (plotting).

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
traits$Mass[seq_len(3L)] <- NA_real_
pred <- impute(traits, tree, epochs = 5L, verbose = FALSE)$prediction
plot(pred)
plot(pred, type = "scatter", data = traits)
plot(pred, type = "intervals", trait = "Mass")
```

---

plot_comparison *Forest-plot style comparison of benchmark results*

Description

Takes a data.frame of benchmark results and produces a forest-plot style figure comparing baseline and GNN performance per trait. Each trait appears on the y-axis; points mark RMSE (continuous/count) or accuracy (binary/categorical) for the two methods, connected by a horizontal line colour-coded by whether the GNN improves over the baseline.

Usage

```
plot_comparison(
  results,
  metric = NULL,
  methods = c("BM_baseline", "pigauto_GNN"),
  ...
)
```

Arguments

`results`	A data.frame with columns `trait`, `type`, `metric`, `method`, and `value`. Typically the output of benchmark scripts or reshaped output from `evaluate_imputation`.
`metric`	Character. Which metric to compare. Default `NULL` auto-selects: `"rmse"` for continuous/count traits and `"accuracy"` for binary/categorical traits. If a single metric name is supplied, only traits with that metric are shown.
`methods`	Character vector of length 2: the baseline method name and the GNN method name. Default `c("BM_baseline", "pigauto_GNN")`.
`...`	Additional arguments passed to `plot()`.

Details

When both RMSE and accuracy metrics are present and `metric` is `NULL`, a two-panel figure is produced automatically.

Value

Invisible NULL. Called for its side effect (plotting).

Examples

```
results <- data.frame(
  trait = rep("Mass", 2L), type = rep("continuous", 2L),
  metric = rep("rmse", 2L), method = c("BM_baseline", "pigauto_GNN"),
  value = c(1, 0.8)
)
plot_comparison(results)
plot_comparison(results, metric = "rmse")
```

---

plot_history_gg *Plot training history (ggplot2, deprecated)*

Description

Deprecated. Use the base-R S3 method instead: plot(fit, type = "history").

Produces a ggplot2 validation loss training curve.

Usage

```
plot_history_gg(x, ...)
```

Arguments

x	object of class "pigauto_fit".
...	ignored.

Value

A ggplot object.

Examples

```
fit <- structure(
  list(history = data.frame(epoch = 1:3, val_loss = c(1, 0.7, 0.5))),
  class = "pigauto_fit"
)
plot_history_gg(fit)
```

plot_uncertainty *Plot uncertainty ribbons for imputed trait values*

Description

Plots conformal prediction intervals when present, otherwise an approximate `prediction +/- 1.96 * SE` ribbon, sorted by predicted value. If `truth` is supplied, observed values are overlaid as points. Works for continuous, count, and ordinal traits. For binary traits, plots predicted probabilities with an uncertainty ribbon.

Usage

```
plot_uncertainty(pred_result, truth = NULL, trait_name)
```

Arguments

`pred_result`	list with `imputed` and `se` components (output of `predict.pigauto_fit`).
`truth`	matrix or data.frame of true values (same scale as `pred_result$imputed`), or `NULL`.
`trait_name`	character. Which trait to plot (must match column name).

Value

A `ggplot` object.

Examples

```
pred <- list(
  imputed = data.frame(mass = c(10, 12, 15)),
  se = matrix(c(1, 1.5, 2), ncol = 1,
              dimnames = list(NULL, "mass"))
)
plot_uncertainty(pred, trait_name = "mass")
```

pool_mi *Pool downstream model fits across multiple imputations (Rubin's rules)*

Description

Combine regression coefficients from `M` model fits – one per imputed dataset – into a single pooled table using Rubin's rules. The pooled standard errors combine within-imputation sampling variance and between-imputation variance. Correct Rubin arithmetic does not make an incompatible imputation model inferentially valid. For pigauto inference, fits should come from the documented `multi_impute_analysis()` workflow. The backend is experimental and supports fixed-effect coefficients only. Variance components, correlations, random-effect predictions, BLUPs/conditional modes, latent loadings, and other structured parameters are unsupported.

Usage

```
pool_mi(
  fits,
  conf.level = 0.95,
  coef_fun = NULL,
  vcov_fun = NULL,
  df_fun = NULL,
  tidy_fun = NULL
)
```

Arguments

fits	A list of model fits of length M >= 2. Automatic fixed-effect adapters are provided for stats::lm, stats::glm, nlme::gls, nlme::lme, lme4::merMod, glmmTMB::glmmTMB, drmTMB, and gllvmTMB_multi fits. Other classes implementing coef() and vcov() also work. The output of with_imputations() is accepted directly. MCMCglmm fits are rejected – see Details.
conf.level	Confidence level for the pooled interval (default 0.95).
coef_fun	Optional function extracting a named numeric fixed-effect vector from one fit. NULL uses the automatic class adapter. Custom coefficient and covariance extractors can be supplied independently; callers must ensure custom extractors select fixed effects only.
vcov_fun	Optional function extracting the fixed-effect covariance matrix. NULL uses the automatic class adapter. Base matrices and Matrix objects are accepted; the selected covariance block must be square, finite, symmetric, and have non-negative diagonal entries.
df_fun	Optional function returning the complete-data residual degrees of freedom nu_com from one fit. When supplied, pooled degrees of freedom use the Barnard & Rubin (1999) small-sample correction, which is less biased for short series. When NULL (the default) the classical Rubin (1987) formula is used.
tidy_fun	Optional function returning a data.frame with unique term, numeric estimate, and non-negative finite std.error columns. This is an alternative to coef_fun and vcov_fun, not a supplement; combining them is an error. Callers must ensure it selects fixed effects only.

Details

Let $\hat{\theta}_i$ be the coefficient vector from fit i and $U_i = \mathrm{vcov}(\mathrm{fit}_i)$, for $i = 1, \ldots, M$. Rubin's rules (Rubin 1987) give

$$\bar{\theta} = M^{-1} \sum_i \hat{\theta}_i$$

$$W = M^{-1} \sum_i \mathrm{diag}(U_i)$$

$$B = (M-1)^{-1} \sum_i (\hat{\theta}_i - \bar{\theta})^2$$

$$T = W + (1 + 1/M) B$$

with pooled standard error $\sqrt{T}$. The relative increase in variance is $r = (1 + 1/M)B/W$, the classical pooled df is $\nu_{\text{old}} = (M - 1)(1 + 1/r)^2$, and the fraction of missing information is

$$\text{fmi} = (r + 2/(\nu + 3))/(r + 1).$$

When `df_fun` returns finite complete-data df `nu_com`, the Barnard-Rubin (1999) correction combines $\nu_{\text{obs}} = ((\nu_{\text{com}} + 1)/(\nu_{\text{com}} + 3))\nu_{\text{com}}(1 - \lambda)$ with `nu_old` via $\nu_{\text{BR}} = 1/(1/\nu_{\text{old}} + 1/\nu_{\text{obs}})$. With no between-imputation variation (`B = 0`), the classical limit is `df = Inf`, `riv = 0`, and `fmi = 0`. If finite complete-data df are supplied, Barnard–Rubin instead retains finite observed-data df and its small-sample FMI adjustment. A completely deterministic quantity (`B = W = 0`) always has zero FMI and infinite df.

Only fixed-effect coefficients are pooled in version 0.10.0. Random-effect variances and correlations, BLUPs/conditional modes, latent loadings, and other structured parameters require parameter-specific transformations and are not supported by the automatic `pool_mi()` adapters. Custom extractors are an expert escape hatch and cannot be inspected by pigauto; using them to select unsupported structured parameters is outside the documented scope. The `glmmTMB` adapter selects conditional fixed effects only. The `drmTMB` adapter includes named distributional fixed-effect blocks such as regression coefficients for `mu` and `sigma`; those are fixed coefficients, not random-effect variance components.

MCMCglmm fits are rejected because Rubin's rules are not the right tool for posterior samples: variance decomposition does not generalise cleanly to posterior distributions. No MCMCglmm downstream workflow is supported by the initial analysis-aware backend.

Value

A data.frame with one row per coefficient and columns:

`term` Coefficient name.

`estimate` Pooled point estimate (`mean` across fits).

`std.error` Pooled standard error `sqrt(T)` where `T = W + (1 + 1/M) * B`.

`df` Pooled degrees of freedom (Barnard-Rubin if `df_fun` supplied, else classical Rubin).

`statistic` `estimate / std.error`.

`p.value` Two-sided p-value from a t distribution on `df`.

`conf.low`, `conf.high` Pooled `conf.level` interval.

`fmi` Fraction of missing information.

`riv` Relative increase in variance due to non-response.

References

Rubin DB (1987). *Multiple Imputation for Nonresponse in Surveys.* Wiley.

Barnard J, Rubin DB (1999). "Small-sample degrees of freedom with multiple imputation." *Biometrika* 86(4): 948-955.

Nakagawa S, Freckleton RP (2008). "Missing inaction: the dangers of ignoring missing data." *Trends in Ecology & Evolution* 23(11): 592-596.

Nakagawa S, Freckleton RP (2011). "Model averaging, missing data and multiple imputation: a case study for behavioural ecology." *Behavioral Ecology and Sociobiology* 65(1): 103-116.

See Also

multi_impute_analysis(), with_imputations()

Examples

```
dat <- data.frame(y = stats::rnorm(30L), x = stats::rnorm(30L),
                  z = stats::rnorm(30L))
dat$x[seq(3L, 30L, by = 5L)] <- NA_real_
mi <- multi_impute_analysis(
  data = dat, formula = y ~ x + z, missing = "x",
  model = "lm", m = 2L
)
fits <- with_imputations(mi, function(d) lm(y ~ x + z, data = d))
pool_mi(fits)
```

---

predict.pigauto_fit *Impute missing traits using a fitted pigauto model*

Description

Runs a single forward pass through the fitted model and returns imputed trait values back-transformed to the original scale. Supports all trait types (continuous, binary, categorical, ordinal, count, proportion, zero-inflated count, multi-proportion) and MC dropout for stochastic prediction diagnostics (when `n_imputations` > 1). The fitted model is a gated ensemble of a phylogenetic baseline and a graph neural network correction; prediction is the per-trait blend `(1 - r_cal) * baseline + r_cal * delta_GNN`.

Usage

```
## S3 method for class 'pigauto_fit'
predict(
  object,
  newdata = NULL,
  return_se = TRUE,
  n_imputations = 1L,
  baseline_override = NULL,
  pool_method = c("median", "mean", "mode"),
  clamp_outliers = FALSE,
  clamp_factor = 5,
  match_observed = c("none", "pmm"),
  pmm_K = 5L,
  ...
)
```

Arguments

object	object of class "pigauto_fit".
newdata	NULL (use the training data) or a "pigauto_data" object for new species.
return_se	logical. Compute standard errors? (default TRUE).
n_imputations	integer. Number of stochastic imputation draws — BM posterior samples plus GNN dropout — (default 1L). Set to e.g. 10 or 20 to inspect stochastic prediction variation. These are not validated analysis-aware multiple imputations.
baseline_override	optional list(mu, se) with the same shape as object$baseline. When supplied, predictions use this baseline instead of the one saved in the fit. Used internally by multi_impute_trees() to reuse a trained GNN across posterior trees. Most users can ignore this. Default NULL (use the fit's own baseline).
pool_method	character. How to pool the M decoded draws when n_imputations > 1: "median" (default) takes the per-cell median for count / proportion / zi_count magnitude traits — robust to single dropout-noisy draws amplified by non-linear decoders (expm1 / plogis). "mean" restores the pre-v0.9.2 arithmetic-mean pooling. Continuous / ordinal / binary / categorical pooling is unchanged (linear / probability-averaged). See issue #40.
clamp_outliers	logical. Phase G (v0.9.1.9011+). When TRUE, post-back-transform predictions for log-transformed continuous, count, and zi_count magnitude traits are capped at tm$obs_max * clamp_factor (tm$obs_max is the observed maximum on the original scale, set at preprocess time). Targets tail-extrapolation modes amplified by exp() / expm1() back-transforms. Default FALSE preserves v0.9.1 behaviour exactly.
clamp_factor	numeric scalar (>= 1). Multiplicative factor on the observed maximum used by clamp_outliers. Default 5. Ignored when clamp_outliers = FALSE.
match_observed	character, one of c("none", "pmm"). Phase G' (v0.9.1.9012+). When "pmm", uses Predictive Mean Matching (Little 1988; Buuren mice) on the at-risk types (log-transformed continuous, count, zi_count magnitude, proportion). For each missing cell, finds the pmm_K observed cells whose own predictions are closest to the missing cell's prediction, samples one, and returns its observed value. Imputed values are guaranteed to lie in the observed data range – no extrapolation is possible by construction. **When to use:** PMM is a niche feature in pigauto. The package already provides conformal prediction intervals (calibrated against held-out residuals) and stochastic conformal-width and Brownian/MC-dropout prediction draws. PMM is only worth enabling for: (a) methodological comparison against mice / equivalent packages, or (b) workflows that specifically require imputed values to come from the observed data pool. For tail safety on single-imputation point estimates, prefer clamp_outliers = TRUE. PMM failed the downstream fixed-effect redesign pilot and is unsupported for inference. The Phase G' acceptance bench (useful/MEMO_2026-05-01_phase_g_prime_results.md) confirmed PMM does not strictly improve point-estimate RMSE over the no-PMM default: it wins on extrapolating cells (e.g. AVONET Casuarius) but loses on cells where the GNN's prediction is already accurate (donor-mismatch noise).

	Discrete-class types (binary / categorical / ordinal / multi_proportion) and un-log continuous: no-op. Default "none" preserves v0.9.1.9011 behaviour exactly.
pmm_K	integer scalar (>= 1). Donor pool size for PMM. Default 5L (mice convention). Ignored when match_observed = "none".
...	ignored.

Details

When n_imputations > 1, each imputation m draws a BM posterior sample t_BM_draw ~ N(BM_mu, BM_se) on the latent scale for originally-missing cells (BM_se = 0 for observed cells so they are never perturbed). The model runs in train mode (GNN dropout active) using t_BM_draw as input. The final blend is (1 - r_cal) * t_BM_draw + r_cal * GNN_delta(t_BM_draw): when the calibrated gate is zero the imputation is a pure BM posterior draw; when r_cal > 0 both BM draws and GNN dropout contribute variance. Point estimates are the mean (continuous, count) or mode (binary, categorical, ordinal) across passes. The M complete datasets are returned in imputed_datasets for prediction diagnostics. These datasets are not supported for downstream inference. Use multi_impute_analysis() for the narrow analysis-aware backend after consulting its supported-model and lifecycle documentation.

Decoding per type:

continuous reverse z-score, then exp() if log-transformed

binary sigmoid(latent) to probability, round to 0/1

count reverse z-score of log1p, expm1(), round, clip >= 0

ordinal reverse z-score, round to nearest valid integer level

categorical softmax() over K latent columns, argmax

Value

A list of class "pigauto_pred" with:

imputed data.frame of imputed values in original scale with proper R types (numeric, integer, factor, ordered).

imputed_latent Numeric matrix (n x p_latent) of predictions in latent scale.

se Numeric matrix (n x n_original_traits) of per-cell uncertainty. Continuous/count/ordinal/proportion: SE in original scale (BM conditional SD, delta-method back-transformed). Binary: min(p, 1-p) — probability of being wrong (0 = certain, 0.5 = maximally uncertain); **not** a Gaussian SE. Categorical: 1 - max(p_k) — margin from certainty; **not** a Gaussian SE. NULL if return_se = FALSE.

probabilities Named list. Binary traits: numeric probability vector. Categorical traits: n x K probability matrix. Other types: not present.

imputed_datasets List of M stochastic prediction data.frames when n_imputations > 1; NULL otherwise. Not supported for downstream inference.

trait_map Trait map from the fitted model.

species_names Character vector.

trait_names Character vector.

n_imputations Integer, number of imputations performed.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
data <- preprocess_traits(traits, tree)
splits <- make_missing_splits(data$X_scaled, trait_map = data$trait_map)
fit <- fit_pigauto(data, tree, splits = splits, epochs = 5L,
                   verbose = FALSE)
pred <- predict(fit, return_se = TRUE)
pred$imputed
pred$se
pred$probabilities
pred2 <- predict(fit, n_imputations = 2L)
pred2$imputed_datasets
```

---

preprocess_traits *Preprocess trait data: align to tree, encode into latent space*

Description

Aligns species in the trait data frame to the tree, detects or accepts trait types (continuous, binary, categorical, ordinal, count, proportion, zi_count), and encodes each trait into a continuous latent matrix.

Usage

```
preprocess_traits(
  traits,
  tree,
  species_col = NULL,
  trait_types = NULL,
  multi_proportion_groups = NULL,
  log_cols = NULL,
  log_transform = TRUE,
  center = TRUE,
  scale = TRUE,
  covariates = NULL
)
```

Arguments

traits	data.frame with species as row names (one row per species), or with a species column identified by species_col (potentially multiple rows per species). Columns may be numeric, integer, factor, ordered, character, or logical.
tree	object of class "phylo".

`species_col` character. Name of the column in `traits` that identifies species. When supplied, `traits` may have multiple rows per species. The column is removed from trait columns before encoding. Default `NULL` uses row names (one row per species).

`trait_types` named character vector overriding auto-detection, e.g. `c(Mass = "continuous", Diet = "categorical")`. Valid types: `"continuous"`, `"binary"`, `"categorical"`, `"ordinal"`, `"count"`, `"proportion"`, `"zi_count"`. Proportion and zi_count are override-only (not auto-detected). Unspecified traits are auto-detected. Note that `"multi_proportion"` is NOT set here — use `multi_proportion_groups` instead.

`multi_proportion_groups` named list declaring compositional (multi-proportion) trait groups. Each element is a character vector of column names whose row-wise values sum to 1 (e.g. `list(diet = c("plants", "insects", "fish"))`). The group name becomes a single trait in the output, encoded via centred log-ratio (CLR) + per-component z-score. Group names must NOT match any column in `traits`. Default `NULL`.

`log_cols` character vector of continuous trait names to log-transform. Default `NULL` means auto-detect (log if all observed values are positive). Set to `character(0)` to disable.

`log_transform` logical. Legacy parameter: if `TRUE` and `log_cols` is `NULL`, log-transform all continuous traits with all-positive values. Overridden by `log_cols` when both are supplied.

`center` logical. Subtract column means for continuous/count/ordinal? Default `TRUE`.

`scale` logical. Divide by column SDs for continuous/count/ordinal? Default `TRUE`.

`covariates` data.frame or numeric matrix of environmental covariates. Covariates are conditioners: they inform imputation but are not themselves imputed, so they must be **fully observed** (no NAs — if a variable has missing values, put it in `traits` instead). Must have the same number of rows as `traits` after alignment to the tree.

Numeric / integer columns z-scored automatically.

Factor / ordered columns one-hot encoded (K binary columns per factor with K levels). Column names become `"var.level"`.

Character / logical columns coerced to factor, then one-hot.

Default `NULL` (no covariates).

Details

When each species has one observation (the default), output rows match `tree$tip.label` order. When `species_col` is supplied, multiple observations per species are supported: the output matrix has one row per observation, plus an `obs_to_species` mapping for the GNN (which operates at species level).

Automatic type detection (when `trait_types = NULL`) follows the R class of each column — no user input is required for most data:

R class	pigauto type

`numeric` (non-integer)	continuous
`integer`	count
`factor` with 2 levels	binary
`factor` (unordered) with >2 levels	categorical
`ordered` / `factor(..., ordered = TRUE)`	ordinal
`character`	converted to `factor`, then binary or categorical
`logical`	binary (FALSE = 0, TRUE = 1)

Two types require an explicit override because they cannot be distinguished from their R class alone:

`"proportion"` A `numeric` column bounded 0–1 looks identical to continuous. Declare it explicitly: `trait_types = c(SurvivalRate = "proportion")`. Encoded via `qlogis(clamp(x, 0.001, 0.999))`.

`"zi_count"` An `integer` column with excess zeros looks identical to count. Declare it explicitly: `trait_types = c(Parasites = "zi_count")`. Encoded as a binary zero/non-zero gate plus `log1p`-z magnitude.

Practical examples of type assignment:

- Body mass (`numeric`, all positive) → `continuous`, auto-log-transformed.
- Clutch size (`integer`) → `count`.
- Migratory (`factor` with levels "Yes"/"No") → `binary`.
- Diet (`factor` with >2 levels) → `categorical`.
- IUCN status (`ordered` factor, LC < NT < VU < EN < CR) → `ordinal`. If left as an unordered factor, it becomes `categorical` — both are valid depending on the question.
- Parasite load (`integer` with many zeros) → needs `trait_types = c(Parasites = "zi_count")`.
- Survival rate (`numeric`, values in 0 to 1) → needs `trait_types = c(Survival = "proportion")`.

Latent encoding per type:

continuous optional `log()`, then z-score (1 latent column)

binary 0/1 encoding (1 latent column)

count `log1p()`, then z-score (1 latent column)

ordinal integer coding (0 to K-1), then z-score (1 latent column)

categorical one-hot encoding (K latent columns)

proportion `qlogis(clamp(x, 0.001, 0.999))`, then z-score (1 latent column)

zi_count gate (0/1) + `log1p`-z of non-zeros (2 latent columns)

multi_proportion centred log-ratio (CLR) + per-component z-score (K latent columns per group). Rows must sum to 1. Declared via the `multi_proportion_groups` argument, not `trait_types`.

Value

A list of class "pigauto_data" with components:

X_scaled Numeric matrix (n_obs x p_latent), latent encoding. When species_col is NULL, n_obs = n_species.

X_raw Numeric matrix of continuous traits after optional log but before z-scoring (for backward compatibility).

X_original Original data.frame (aligned to tree, before encoding).

means Named numeric vector of column means used for z-scoring (continuous/count/ordinal traits only).

sds Named numeric vector of column SDs.

species_names Character vector of unique species matching tree$tip.label order (length = n_species).

obs_species Character vector of species labels per observation (length = n_obs). When multi-obs, can have duplicates. NULL when species_col is NULL.

obs_to_species Integer vector (length = n_obs) mapping each observation to its species index in species_names. NULL when species_col is NULL.

n_species Integer, number of unique species.

n_obs Integer, number of observations (= n_species when single-obs).

multi_obs Logical, TRUE when multiple observations per species are present.

trait_names Character vector of original trait names.

latent_names Character vector of latent column names.

trait_map List of trait descriptors (see Details).

p_latent Integer, total number of latent columns.

log_transform Logical, legacy field (TRUE if any continuous trait was log-transformed).

Examples

```
# Single-obs per species (backward compatible)
data(avonet300, tree300, package = "pigauto")
traits <- avonet300
rownames(traits) <- traits$Species_Key
traits$Species_Key <- NULL
pd <- preprocess_traits(traits, tree300)
dim(pd$X_scaled)   # 300 x p_latent

# Multi-obs per species (via species_col)
pd2 <- preprocess_traits(avonet300, tree300, species_col = "Species_Key")
pd2$n_obs      # number of observations
pd2$n_species  # number of unique species
```

`pull_gbif_centroids` *Fetch species range-centroid covariates from GBIF*

Description

Pulls occurrence records from the Global Biodiversity Information Facility (GBIF) for each species in `species`, aggregates them to a median lat / lon centroid, and returns a data.frame ready to be passed to `impute` via its `covariates` argument.

Usage

```
pull_gbif_centroids(
  species,
  cache_dir = NULL,
  occurrence_limit = 500L,
  sleep_ms = 100L,
  verbose = TRUE,
  refresh_cache = FALSE,
  store_points = FALSE
)
```

Arguments

`species`	character vector of species names.
`cache_dir`	directory to cache per-species RDS files. `NULL` (default) disables caching — not recommended for production use.
`occurrence_limit`	integer, maximum number of occurrences to fetch per species (default 500; paginated if > 300).
`sleep_ms`	integer, polite delay between API calls in milliseconds (default 100).
`verbose`	logical, print progress every 50 species.
`refresh_cache`	logical, force re-fetch even when cache exists.
`store_points`	logical. When `TRUE`, persists the raw filtered lat/lon occurrence points in each species' cache RDS under the `points` field. Used by `pull_worldclim_per_species` for per-occurrence bioclim extraction. Default `FALSE` preserves the pre-v0.9.1.9006 cache format.

Details

Caching is strongly recommended — GBIF rate-limits anonymous calls and the per-species fetch is expensive. With `cache_dir` set, each species gets one RDS file; subsequent calls skip the API.

For each species: resolve taxon via `name_backbone`, fetch up to `occurrence_limit` records via `occ_search` (paginated at 300 per GBIF call), filter out records with `hasGeospatialIssues = TRUE` and `basisOfRecord` in `c("FOSSIL_SPECIMEN", "LIVING_SPECIMEN")`, drop out-of-range coordinates, then take the median latitude and longitude as the species centroid.

Species with zero post-filter records receive NA centroids; their rows are still included in the returned data.frame so it aligns with the input species list.

Requires the optional rgbif package (in DESCRIPTION Suggests). If rgbif is not installed the function errors with an installation message.

Value

A data.frame with columns species, centroid_lat, centroid_lon, n_occurrences. Rownames are set to species. NA centroids are returned for species with no GBIF hits; the caller should decide how to handle them (typical: drop or impute).

See Also

impute (pass the return value as covariates).

Examples

```
sp <- c("Quercus alba", "Pinus taeda", "Acer saccharum")
cov <- pull_gbif_centroids(sp, cache_dir = "script/data-cache/gbif")
cov_num <- cov[, c("centroid_lat", "centroid_lon"), drop = FALSE]
```

---

pull_worldclim_per_species

Fetch per-species bioclim covariates from WorldClim v2.1

Description

Extends pull_gbif_centroids by extracting 19 WorldClim v2.1 bioclim variables at each species' GBIF occurrence points, aggregating per species (median + IQR), and returning a data.frame ready for impute via its covariates argument.

On first call, downloads the WorldClim 10-arc-minute raster stack (~130 MB compressed, ~500 MB unzipped) to worldclim_cache_dir. A sentinel file makes subsequent calls fully offline.

Per-species extracts are also cached (one RDS per species in worldclim_cache_dir/extracts/), so repeated calls for the same species list are instantaneous.

Since v0.9.1.9006 (2026-04-30) extraction operates on the raw per-occurrence point list when pull_gbif_centroids was called with store_points = TRUE. For legacy caches written by older pull_gbif_centroids (centroid-only) calls, extraction silently falls back to the species centroid.

Requires the optional terra package (in DESCRIPTION Suggests). If terra is not installed the function errors with an installation message.

Usage

```
pull_worldclim_per_species(
  species,
  gbif_cache_dir,
  worldclim_cache_dir,
  resolution = "10m",
  verbose = TRUE,
  refresh_cache = FALSE
)
```

Arguments

species	character vector of species binomials.
gbif_cache_dir	character path, GBIF per-species cache directory (as written by pull_gbif_centroids).
worldclim_cache_dir	character path, directory for WorldClim rasters + per-species extract cache. Created if absent.
resolution	one of "10m" (default), "5m", "2.5m".
verbose	logical. Print progress every 50 species.
refresh_cache	logical. Force re-extract even when per-species cache exists.

Value

A data.frame with length(species) rows and columns:

- species
- bio1_median, bio1_iqr, ..., bio19_median, bio19_iqr
- n_extracted (integer, number of valid occurrence points that contributed)

Rownames are set to species. Species with no GBIF hits get all-NA bio values and n_extracted = 0.

References

Fick SE, Hijmans RJ (2017). WorldClim 2: new 1-km spatial resolution climate surfaces for global land areas. International Journal of Climatology 37, 4302–4315.

See Also

pull_gbif_centroids (B.1, required input), impute (consume as covariates).

Examples

```
sp <- c("Quercus alba", "Pinus taeda", "Acer saccharum")
gbif_df <- pull_gbif_centroids(sp,
  cache_dir = "script/data-cache/gbif")
wc_df   <- pull_worldclim_per_species(
  species            = sp,
```

```
  gbif_cache_dir      = "script/data-cache/gbif",
  worldclim_cache_dir = "script/data-cache/worldclim")
# Combine for impute()
cov <- cbind(gbif_df[, c("centroid_lat", "centroid_lon")],
             wc_df[, grep("^bio", colnames(wc_df))])
```

`read_traits` *Read trait data from a CSV file or data frame*

Description

Loads trait data and sets species names as row names. If a CSV path is supplied, it is read with `read.csv`. By default, all non-species columns are returned (numeric, factor, integer, etc.). Use `trait_cols` to select a subset.

Usage

```
read_traits(
  x,
  species_col = "species",
  trait_cols = NULL,
  factor_cols = NULL,
  ordered_cols = NULL
)
```

Arguments

`x`	character path to a CSV file, or a `data.frame`.
`species_col`	character. Name of the column containing species names (default `"species"`).
`trait_cols`	character vector of column names to include. If `NULL` (default), all columns except `species_col` are used.
`factor_cols`	character vector of columns to coerce to `factor`.
`ordered_cols`	character vector of columns to coerce to `ordered`.

Value

A `data.frame` with species names as row names.

Examples

```
df <- data.frame(species = c("Sp_a", "Sp_b"), mass = c(10, 20))
traits <- read_traits(df, species_col = "species")
```

read_tree *Read a phylogenetic tree from a file*

Description

Reads a Newick or NEXUS tree file using **ape**. Tries `ape::read.tree` first; falls back to `ape::read.nexus` if that fails.

Usage

```
read_tree(path)
```

Arguments

`path`	character. Path to the tree file.

Value

An object of class `"phylo"`.

Examples

```
path <- tempfile(fileext = ".tre")
ape::write.tree(ape::rtree(10L), path)
tree <- read_tree(path)
```

save_pigauto *Save a fitted pigauto model*

Description

Serialises a `pigauto_fit` object to disk, including the torch model state. The saved file can be loaded on any machine with the same pigauto version.

Usage

```
save_pigauto(fit, path, compress = TRUE)
```

Arguments

`fit`	pigauto_fit object.
`path`	character. File path to save to (recommended extension: `.pigauto`).
`compress`	logical. Use gzip compression (default `TRUE`).

Details

Standard `saveRDS()` does not work for pigauto fits because torch tensor states cannot be serialised by R's native mechanism. This function converts the model state to a portable raw format before saving.

Value

Invisible `path`.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
traits$Mass[seq_len(3L)] <- NA_real_
fit <- impute(traits, tree, epochs = 5L, verbose = FALSE)$fit
path <- tempfile(fileext = ".pigauto")
save_pigauto(fit, path)
fit2 <- load_pigauto(path)
```

---

simulate_benchmark *Run a simulation benchmark for pigauto*

Description

Generates trait data under various evolutionary models, introduces missing data, fits both the Brownian motion baseline and the full pigauto GNN, and compares performance. This is the recommended way to assess pigauto on data with known properties before applying it to real data.

Usage

```
simulate_benchmark(
  n_species = 100L,
  n_traits = 4L,
  scenarios = c("BM", "OU", "regime_shift", "nonlinear", "mixed"),
  missing_frac = 0.25,
  n_reps = 3L,
  epochs = 500L,
  verbose = TRUE,
  seed = NULL,
  ...
)
```

Arguments

n_species	integer. Number of tips in the simulated tree (default 100).
n_traits	integer. Number of continuous traits (default 4). Ignored for scenario = "mixed", which generates a fixed trait set.
scenarios	character vector. Subset of c("BM", "OU", "regime_shift", "nonlinear", "mixed"). Default runs all.
missing_frac	numeric. Fraction of observed cells held out (default 0.25).
n_reps	integer. Number of replicate trees per scenario (default 3).
epochs	integer. Maximum GNN training epochs (default 500).
verbose	logical. Print progress (default TRUE).
seed	optional integer. When supplied, makes the generated benchmark replicates reproducible.
...	additional arguments passed to fit_pigauto.

Details

Available scenarios:

"BM" Pure Brownian motion – the baseline is exact, so the GNN should tie or slightly improve via inter-trait correlations.

"OU" Ornstein-Uhlenbeck – stabilising selection constrains variation. BM over-estimates evolutionary variance.

"regime_shift" Two-regime BM – clade-specific optima create bimodal distributions that BM cannot capture.

"nonlinear" Non-linear inter-trait relationships – the GNN's multi-layer message passing can capture quadratic and interaction effects that BM's linear covariance misses.

"mixed" Mixed trait types: 2 continuous + 1 binary + 1 categorical (3 levels). Tests the full type pipeline.

Value

An object of class "pigauto_benchmark" with:

results data.frame with columns: scenario, rep, method, trait, type, metric, value, n_test.

summary data.frame averaged across replicates.

scenarios character vector of scenarios run.

n_reps integer.

n_species integer.

Examples

```
bench <- simulate_benchmark(n_species = 20L, n_traits = 2L,
                            scenarios = "BM", epochs = 5L, n_reps = 1L,
                            verbose = FALSE)
bench$summary
plot(bench)
```

simulate_non_bm *Simulate non-BM trait data for benchmarking*

Description

Generates species trait data under models that deviate from Brownian Motion. Supports OU (stabilising selection), regime shifts (clade-specific optima), and non-linear inter-trait correlations.

Usage

```
simulate_non_bm(
  tree,
  n_traits = 4,
  scenario = c("OU", "regime_shift", "nonlinear"),
  alpha = 2,
  theta = 0,
  sigma = 1,
  shift_magnitude = 2,
  seed = NULL
)
```

Arguments

tree	Object of class "phylo".
n_traits	Integer. Number of traits to simulate (default 4).
scenario	Character. One of "OU", "regime_shift", "nonlinear".
alpha	Numeric. OU pull strength (only for "OU").
theta	Numeric. OU optimum (only for "OU").
sigma	Numeric. BM diffusion rate.
shift_magnitude	Numeric. Regime shift size in SD units (only for "regime_shift").
seed	Integer seed or NULL.

Value

A data.frame with species as rownames.

Examples

```
tree <- ape::rtree(12)
traits <- simulate_non_bm(tree, n_traits = 2, scenario = "OU", seed = 1)
head(traits)
```

`suggest_next_observation` *Suggest which cell to observe next to maximise imputation precision*

Description

For a fitted `pigauto_result` (returned by `impute`), compute the closed-form expected reduction in total predictive variance across all currently-missing cells if each candidate cell were observed next. Useful for sampling-design guidance: when you have time/budget to measure *k* more species, this function tells you which ones contribute most to imputation precision.

Usage

```
suggest_next_observation(
  result,
  top_n = 10L,
  by = c("cell", "species"),
 types = c("continuous", "count", "ordinal", "proportion", "binary", "categorical",
    "zi_count", "multi_proportion")
)
```

Arguments

`result`	A `pigauto_result` object returned by `impute`. Must have been produced from single-obs data; multi-obs inputs error with a clear message.
`top_n`	integer, default `10L`. Number of suggestions to return (descending by `delta`).
`by`	character, one of `"cell"` (default) or `"species"`. `"cell"` returns individual `(species, trait)` pairs. `"species"` aggregates by species (summing reductions across the species' currently-missing traits).
`types`	character vector of pigauto trait types to include. Default includes all eight supported types: `continuous`, `count`, `ordinal`, `proportion`, `binary`, `categorical`, `zi_count` (added v2, 2026-05-01), `multi_proportion` (added v2).

Details

Two metrics are supported, dispatched by trait type:

Continuous-family traits (continuous, count, ordinal, proportion) use the BM variance-reduction formula. The variance- reduction formula is derived from a Sherman-Morrison rank-1 inverse update on the BM conditional MVN: adding species s to the observed set updates the inverse correlation matrix by a known closed form. For each candidate cell (s, t),

$$\Delta V(s,t) = \sigma_t^2 \sum_{i \in \text{miss}_t} \frac{D_{ik}^2}{\alpha_k}$$

where $D = R_{mm} - R_{mo}R_{oo}^{-1}R_{om}$ is the residual matrix at currently-missing cells, $\alpha_k = D_{kk}$ is the current relative leverage of cell k, and σ_t^2 is the REML BM variance for trait t.

Discrete traits (binary, categorical) use a label- propagation expected-entropy-reduction formula. The current LP probability at species i is $p_i = \mathrm{sim}[i, \mathrm{obs}] y_{\mathrm{obs}} / \sum \mathrm{sim}[i, \mathrm{obs}]$, with entropy $H(p_i) = -\sum_k p_{i,k} \log p_{i,k}$. After observing s_{new} with unknown class y_{new}, the new LP probability has a closed form, and the expected entropy is averaged over $P(y_{\mathrm{new}})$ = current LP estimate at s_{new}. Total expected entropy reduction sums across all currently-missing cells (the entropy at s_{new} itself drops to 0).

Variance and entropy are NOT directly comparable. The output sorts within each metric and the cross-metric ordering by `delta` is approximate. When you want a strict ranking, filter by `metric` first.

Reductions are summed across the included traits for each species when `by = "species"`, supporting the typical use case where measuring a species observes all of its currently-missing traits at once. At `by = "species"`, the per-trait variance and entropy reductions are summed separately into `delta_var_total` and `delta_entropy_total` columns; the `delta` column is whichever is non-NA (or `delta_var_total` when both are populated). Cross-type species-level ranking is approximate – see the variance-vs- entropy caveat above.

`zi_count` (v2): observing a missing zi_count cell reveals the gate value (entropy reduction at the gate column, computed via the LP binary formula) AND, with probability p_{gate} (current LP estimate at s_{new}), reveals a magnitude (variance reduction at the magnitude column, computed via the BM Sherman-Morrison formula on the gate=1 subset). Output rows for zi_count populate BOTH `delta_var_total` (= expected magnitude variance reduction = $p_{\mathrm{gate}} \times \Delta V_{\mathrm{mag}}$) AND `delta_entropy_total` (= gate entropy reduction). `metric` is set to `"variance"` so the row sorts on the magnitude scale; `delta_entropy_total` is available for users who care about gate-uncertainty separately.

`multi_proportion` (v2): observing a row reveals all K simplex components simultaneously. Per-component variance reductions are computed via BM Sherman-Morrison on each CLR-z latent column, summed across components. `metric` is `"variance"`; `delta_var_total` is the K-component sum.

Value

A data.frame of class `"pigauto_active"`. Columns when `by = "cell"`: `species`, `trait`, `type`, `metric` (`"variance"` or `"entropy"`), `delta`, `delta_var_total` (NA for discrete rows), and `delta_entropy_total` (NA for continuous rows), sorted by `delta` descending. When `by = "species"`: `species`, `delta_var_total`, `delta_entropy_total`, `n_traits_missing`, sorted by the SUM of available metrics.

See Also

impute

Examples

```
data(avonet300, tree300, package = "pigauto")
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
traits$Mass[seq_len(3L)] <- NA_real_
res <- impute(traits, tree, epochs = 5L, verbose = FALSE)
suggest_next_observation(res, top_n = 5L)
```

```
suggest_next_observation(res, top_n = 5L, by = "species")
```

`summary.pigauto_fit` *Summary method for pigauto_fit objects*

Description

Prints a formatted evaluation table including per-trait metrics, gate calibration, and conformal coverage. Requires the original `pigauto_data` object to compute test-set performance.

Usage

```
## S3 method for class 'pigauto_fit'
summary(object, ..., data = NULL)
```

Arguments

`object`	pigauto_fit object.
`...`	ignored.
`data`	pigauto_data object used for fitting (optional; when `NULL` the summary skips per-trait test metrics).

Value

Invisibly returns the evaluation data.frame (or `NULL` if `data` is not supplied).

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30L)])
traits <- avonet300[match(tree$tip.label, avonet300$Species_Key),
                    c("Mass", "Wing.Length"), drop = FALSE]
rownames(traits) <- tree$tip.label
pd <- preprocess_traits(traits, tree)
splits <- make_missing_splits(pd$X_scaled, trait_map = pd$trait_map)
fit <- fit_pigauto(pd, tree, splits = splits, epochs = 5L,
                   verbose = FALSE)
summary(fit, data = pd)
```

tree300 *Example bird phylogeny for the 300 species in* avonet300

Description

An object of class 'phylo' from the **ape** package. The tree is a posterior Hackett-backbone sample distributed by the MIT-licensed **megatrees** package, pruned to the 300 species present in avonet300. It is intended for examples, not as a consensus or recommended tree for substantive analysis.

Usage

```
tree300
```

Format

An object of class phylo with 300 tips.

Source

Li (2026), **megatrees** 1.0.0, MIT licence; underlying tree sample from Jetz et al. (2012), Hackett et al. backbone.

trees300 *50 posterior phylogenies for the 300 species in* avonet300

Description

A multiPhylo list of 50 phylogenetic trees randomly sampled from the BirdTree Hackett backbone posterior (Jetz et al. 2012), each pruned to the 300 species in avonet300. These trees capture phylogenetic uncertainty: topologies and branch lengths vary across the posterior sample.

Usage

```
trees300
```

Format

An object of class multiPhylo containing 50 phylo objects, each with 300 tips.

Details

Use with multi_impute_trees for experimental sensitivity of point imputations to posterior-tree choice. Tree uncertainty is not supported by the analysis-aware MI backend, and these stochastic datasets are not validated for downstream inference.

Source

Li (2026), **megatrees** 1.0.0, MIT licence; trees pruned from `megatrees::get_tree_bird_n100()`. Underlying posterior from Jetz et al. (2012), Hackett et al. backbone.

See Also

`tree300`, `avonet300`, `multi_impute_trees`

---

tree_full *Example bird phylogeny for the species in* avonet_full

---

Description

An object of class `'phylo'` from the **ape** package. The tree is the same posterior Hackett-backbone sample used for `tree300`, but pruned to the species present in `avonet_full` rather than a 300-species random subset.

Usage

```
tree_full
```

Format

An object of class `phylo` with 9,993 tips.

Details

Row order in `avonet_full` matches tip order in `tree_full`: `all(avonet_full$Species_Key == tree_full$tip.label)` returns `TRUE`.

Source

Li (2026), **megatrees** 1.0.0, MIT licence; underlying tree sample from Jetz et al. (2012), Hackett et al. backbone.

See Also

`tree300`, `avonet_full`

`with_imputations` *Fit a downstream model on every imputed dataset*

Description

Apply a user-supplied model-fitting function `.f` to each of the `M` complete datasets stored in a `pigauto_mi` object and return the list of fits. For inference, use an object returned by `multi_impute_analysis()`; conformal-width, Brownian/MC-dropout, and PMM prediction-diagnostic draws are unsupported. The analysis-aware workflow is experimental and is limited to fixed-effect coefficients and their covariance matrices.

Usage

```
with_imputations(
  mi,
  .f,
  ...,
  .progress = interactive(),
  .on_error = c("continue", "stop")
)
```

Arguments

`mi`	A `pigauto_mi` object returned by `multi_impute_analysis()`. Plain lists of data.frames are also accepted and treated as the `datasets` slot directly.
`.f`	A function of the form `function(dataset, ...)` that fits a model to one complete data.frame and returns a model object. `pool_mi()` supplies automatic fixed-effect adapters for its documented model classes. Other classes require `coef()` and `vcov()` methods that return compatible fixed-effect quantities, or explicit extractor functions supplied to `pool_mi()`; extractability does not imply that an unlisted analysis model has passed the analysis-aware validation gate. When `mi` comes from `multi_impute_trees()`, `.f` may also declare explicit `tree`, `tree_index`, or `imputation` arguments; these are filled with the posterior tree object, its index in `mi$trees`, and the stochastic-completion index. The dataset also carries matching attributes. This metadata support is for prediction-sensitivity diagnostics only; tree-aware downstream inference is unsupported.
`...`	Additional arguments passed to `.f` for every imputation.
`.progress`	Logical. Show a text progress indicator (default `TRUE` in interactive sessions).
`.on_error`	One of `"continue"` (default) or `"stop"`. When `"continue"`, errors from `.f` are captured per imputation and the loop proceeds; a warning at the end summarises failures. When `"stop"`, the first error aborts the entire run.

Value

A list of length `M` with class `"pigauto_mi_fits"`. Each element is either a model fit or, if `.f` errored on that imputation and `.on_error = "continue"`, an object of class `"pigauto_mi_error"` containing the captured condition. `pool_mi()` filters error elements automatically.

See Also

multi_impute_analysis(), pool_mi()

Examples

```
dat <- data.frame(y = stats::rnorm(30L), x = stats::rnorm(30L),
                  z = stats::rnorm(30L))
dat$x[seq(3L, 30L, by = 5L)] <- NA_real_
mi <- multi_impute_analysis(
  data = dat, formula = y ~ x + z, missing = "x",
  model = "lm", m = 2L
)
fits <- with_imputations(mi, function(d) stats::lm(y ~ x + z, data = d))
pool_mi(fits)
```

Index