Package 'pigauto'

November 9, 2026

Title Fill in Missing Species Traits Using a Phylogenetic Tree

Version 0.11.1

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 default method is a phylogenetic baseline calibrated on held-out data; an optional graph neural network prediction is blended in through a per-trait gate that retains a baseline fallback. Covariates can inform the network and selected baseline paths. Provides conformal prediction intervals for continuous, count, ordinal, and proportion 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.
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, covr, yaml, jsonlite, knitr, rmarkdown, pkgdown, RSpectra, Rphylopars, jomo, MCMCglmm, brms, glmmTMB, lme4, nlme, phangorn, phylolm, rgbif, smcfcs, terra, phytools

VignetteBuilder knitr

Config/testthat/edition 3

Config/roxygen2/version 8.0.0

RoxygenNote 7.3.3

NeedsCompilation no

Author Shinichi Nakagawa [aut, cre, cph], Daniel Noble [ctb]

Maintainer Shinichi Nakagawa <itchyshin@gmail.com>

Repository CRAN

Date/Publication 2026-11-09 13:20:02 UTC

Contents

avonet300 . . . 3
avonet_full . . . 4
build_phylo_graph . . . 5
calibration_df . . . 7
check_pigauto . . . 7
compare_methods . . . 9
completed_data . . . 11
confusion_matrix . . . 12
cross_validate . . . 13
ctmax_sim . . . 15
evaluate . . . 16
evaluate_imputation . . . 17
fit_baseline . . . 19
fit_pigauto . . . 24
get_imputed_datasets . . . 32
impute . . . 33
load_pigauto . . . 42
make_missing_splits . . . 43
mask_missing . . . 45
multi_impute . . . 45
multi_impute_analysis . . . 56
multi_impute_trees . . . 58
pigauto_report . . . 64
plot.pigauto_benchmark . . . 65
plot.pigauto_fit . . . 66

plot.pigauto_pred . . . 67
plot.pigauto_result . . . 68
plot_comparison . . . 69
plot_history_gg . . . 70
plot_missingness . . . 70
plot_uncertainty . . . 71
pool_mi . . . 72
predict.pigauto_fit . . . 76
preprocess_traits . . . 82
print.pigauto_check . . . 86
print.summary_pigauto_result . . . 87
pull_gbif_centroids . . . 87
pull_worldclim_per_species . . . 89
read_traits . . . 91
read_tree . . . 92
save_pigauto . . . 93
simulate_benchmark . . . 94
simulate_non_bm . . . 97
suggest_next_observation . . . 98
summary.pigauto_fit . . . 101
summary.pigauto_result . . . 102
tree300 . . . 103
trees300 . . . 104
tree_full . . . 105
with_imputations . . . 106

Index **108**

`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 using `min(max(ceiling(n/20), 4), 32)`, then clamps the result to at most `max(n - 1, 1)` for the tree size. For example, the formula gives 5 for 100 tips, 8 for 160, 15 for 300, and 32 for 640 or more. For a one-tip tree the clamp yields one, but the eigendecomposition rank guard still stops because no non-trivial eigenvector is available. An explicitly supplied integer is not reduced by this automatic clamp.
`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; matching dimensions do not establish that the tree tip identities/order or bandwidth are the same. Use a distinct cache path for each unchanged tree, tip order, and `sigma_mult` combination, and remove or refresh the cache after changing any of them. `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 to add a learnable per-head Gaussian distance bias to attention scores. During `impute()` training, `D_sq` remains allocated while `D` and `R_phy` are cleared before model fitting.

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

The implementation stores the dense cophenetic-distance, adjacency, and Laplacian matrices, so memory use remains $O(n^2)$. For trees with more than 7500 tips, **RSpectra**'s iterative Lanczos eigensolver (`eigs_sym`) requests a partial spectrum from the dense Laplacian; it reduces the eigendecomposition work, but does not make graph storage sparse. The dense `base::eigen()` path is used below this threshold and as a fallback when **RSpectra** is unavailable or Lanczos fails. Runtime depends on tree structure, hardware, and convergence. A benchmark on one 9,993-tip AVONET tree is a fixture-specific result, not a general timing guarantee. 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 in [0, 1]. Pairs with an NA in either vector are dropped with a warning; an error is raised if no complete pairs remain.
n_bins	integer, number of bins (default 10). Must be a positive integer.

Value

A data.frame with columns bin_mid, obs_freq, mean_pred, and n. Empty bins are omitted. For each returned bin, n counts the complete truth/prob pairs assigned to that bin.

Examples

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

check_pigauto *Check whether traits and a tree are ready for pigauto*

Description

Inspect declared inputs without constructing a phylogenetic graph or fitting a model. The check is intended to make input problems and likely compute scale visible before calling impute(). Its fingerprint identifies the declared input object for comparison within a workflow; it is not a security hash.

Usage

```
check_pigauto(
  traits,
  tree,
  species_col = NULL,
  trait_types = NULL,
  multi_proportion_groups = NULL,
  covariates = NULL,
  gnn = FALSE,
  ordinal_levels = NULL
)
```

Arguments

traits	A data.frame of traits, with row names or a species column.
tree	A phylo tree.
species_col	Optional name of the column in the traits data frame that contains species identities.
trait_types	Optional named type overrides passed to preprocess_traits().
multi_proportion_groups	Optional named composition groups passed to preprocess_traits().
covariates	Optional environmental covariates passed to preprocess_traits(). Non-finite values are an error.
gnn	Whether the planned imputation will use the graph neural network. The default FALSE skips the torch runtime probe, matching impute()'s default. Set TRUE to include the torch runtime in the preflight.
ordinal_levels	Optional named list giving the order of ordinal levels, applied before type detection so the report shows the same types the matching impute() call will use. Invalid declarations are reported as errors rather than thrown. See impute() / read_traits().

Value

A list of class pigauto_check containing a schema version, status, readiness flag, stable input fingerprint, input summaries, and structured messages.

Reading the result

Printing the result shows its status and, when present, up to three message/action pairs. Inspect check$messages for the complete table and follow its action column. The four status values mean:

- "ready": the implemented checks found no errors or warnings.
- "ready_with_warnings": review the warnings before deciding to fit.
- "error": repair the reported problems before calling impute().

- "not_needed": all matched target cells are observed, so no imputation is needed. Use cross_validate() to assess prediction instead.

check$ready is TRUE for "ready" and "ready_with_warnings": the current input checks found no errors, though warnings may remain. It is FALSE for "error" and "not_needed". Review any warnings and assess whether a model is scientifically suitable for your data separately. Inspect check$species and check$traits to confirm species matching and trait declarations.

Examples

```
tree <- ape::read.tree(text = "((a:1,b:1):1,c:2);")
traits <- data.frame(x = c(1, NA, 3), row.names = tree$tip.label)
check <- check_pigauto(traits, tree)
check
check$ready
check$messages

# An input error is returned as a report with a suggested action.
bad_tree <- tree
bad_tree$edge.length <- NULL
check_pigauto(traits, bad_tree)
```

---

compare_methods *Compare BM baseline and pigauto methods across replicates*

Description

Compares the BM baseline and pigauto (with attention and calibration). The default performs one run using the current random-number state; supply a vector of seeds for replicated runs. Returns a tidy data.frame suitable for plotting or downstream analysis. This helper uses the default baseline controls, including estimated continuous-family lambda and automatic exact/per-column prediction routing. Arguments in ... do not change the already-fitted baseline used for the held-out comparison.

Usage

```
compare_methods(
  data,
  tree,
  splits = NULL,
  seeds = NULL,
  epochs = 500L,
  verbose = TRUE,
  gnn = 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 random-number state.
epochs	number of training epochs.
verbose	logical.
gnn	logical. Whether the pigauto arm trains the GNN. Default TRUE, so this function keeps comparing the baseline with the GNN-corrected model even though fit_pigauto and impute default to gnn = FALSE.
...	additional arguments passed to fit_pigauto. Baseline controls supplied here do not change the already fitted baseline used for held-out comparison.

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.

The default comparison trains the optional torch GNN. The example checks that torch can create a tensor before fitting; it reports a skip when the Lantern runtime is unavailable.

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
torch_ready <- requireNamespace("torch", quietly = TRUE) &&
  isTRUE(tryCatch(torch::torch_is_installed(), error = function(e) FALSE)) &&
  isTRUE(tryCatch({
    probe <- torch::torch_tensor(1)
    as.numeric(probe$item()) == 1
  }, error = function(e) FALSE))
if (torch_ready) {
  cmp <- compare_methods(preprocess_traits(traits, tree), tree, seeds = 1L,
                         epochs = 5L, verbose = FALSE, gnn = TRUE)
```

```
  aggregate(value ~ method + trait + metric, data = cmp, FUN = mean)
} else {
  message("Skipping GNN comparison: install torch and its Lantern runtime.")
}
```

`completed_data` *Extract completed trait data*

Description

Extract completed trait data

Usage

```
completed_data(result)
```

Arguments

`result`	A `pigauto_result`.

Value

The input-shaped data frame returned by `impute()`, with observed trait values and non-trait columns retained. Predictions fill modeled missing trait cells when available; cells that could not be filled remain `NA`.

See Also

`impute()`. For analyses that should carry imputation uncertainty, use `multi_impute()` and extract its `m` datasets with `get_imputed_datasets()` instead of a single completion.

Examples

```
## Not run: completed_data(result)
```

`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 labels (default: sorted union of labels observed in truth and predicted). If you supply this, include every label you want counted. Any truth or prediction label not listed is treated as missing; that pair is omitted from the confusion table and the denominator used to calculate accuracy.

Details

Pairs with a missing truth or prediction are omitted from the confusion table and accuracy denominator. If no complete pairs remain, accuracy is `NaN` (undefined). Per-class precision or recall is `NA` when its denominator is zero. F1 is `NA` when precision or recall is `NA`, or when both are zero.

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

Evaluates trait-imputation performance by rotating observed cells within each trait through k test folds. Folds do not hold out whole species or clades, and they are not stratified by outcome class. Returns per-fold and aggregated metrics.

Displays the number of folds and replicates, then reports each trait's mean and standard deviation for every metric. Returns x invisibly. Additional arguments are ignored by this print method.

Usage

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

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

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

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 forwarded by `cross_validate()` to `fit_pigauto()`; ignored by the print and summary methods.
`x`	A `pigauto_cv` object returned by `cross_validate()`.
`object`	A `pigauto_cv` object returned by `cross_validate()`.

Details

`cross_validate()` precomputes one graph shared across folds and uses `k_eigen = 8L` for that graph when `k_eigen` is omitted. An explicit `k_eigen` is used for the shared graph and forwarded to `fit_pigauto()`. Folds divide observed cells within each trait; they do not hold out whole species or clades or stratify by outcome class. Interpret each metric on its documented scale in `evaluate_imputation`; continuous and count RMSE use latent scales. Conformal coverage is calculated separately after predictions are decoded.

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.

Invisibly returns the summary data frame stored in `object$summary`.

Printing

`print.pigauto_cv()` reports `k` folds and the replicate count, followed by each trait's metric means and SDs. Displayed `n` is the number of represented fold-by-replicate metric rows, not just the fold count. This count includes rows with missing metric values; means and SDs omit those values. When available, a separate conformal block shows mean coverage and its SD against a nominal 95% target; the target is not a guarantee. Empty summaries print "No results." The method ignores `...` and returns `x` invisibly.

Summary method

`summary.pigauto_cv()` prints a compact per-trait table with RMSE, Pearson r for continuous traits, Spearman ρ for ordinal traits, and accuracy when returned (binary/categorical accuracy or multi-proportion dominant-component accuracy). A dash means the metric row is absent. A present but nonfinite metric may display `NA` or `NaN`. An empty summary prints "No results." The full `object$summary` data frame, which can contain metrics omitted from this display, is returned invisibly. The method ignores `....`

Examples

```
data(avonet300, tree300, package = "pigauto")
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(12L)])
traits <- avonet300[match(tree$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 = 1L, verbose = FALSE,
                     k_eigen = 4L, hidden_dim = 8L,
                     use_transformer_blocks = 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

$$\begin{aligned} CTmax_{ij} &= 38 + phylo_i \\ &\quad + 0.50 \times acclim_temp_{ij} \\ &\quad + \epsilon_{ij} \end{aligned}$$

where $phylo_i$ follows Brownian motion on `tree300`, the within-species acclimation response ratio is 0.50, and $\epsilon_{ij} \sim N(0, 1.5^2)$, where the residual standard deviation is 1.5 degrees Celsius. Thirty percent of species are entirely unobserved (all CTmax values are NA), and an additional 15% of remaining 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. An alternative `data` can provide the truth matrix, and alternative `splits` can select compatible held-out cells from the original fit; evaluation does not refit the baseline or GNN for a new holdout design.

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`). Alternative indices must be a compatible subset of the cells held out when the fit was made; this argument does not refit for a new holdout design.

Details

Metrics by trait type:

continuous RMSE, Pearson r, and MAE on the z-scored latent scale

count RMSE, MAE, and Pearson r on the z-scored log1p latent scale

binary Accuracy, Brier score

categorical Accuracy (overall)

ordinal RMSE on the z-scored latent scale, Spearman rho

proportion RMSE and MAE on the transformed latent scale, Pearson r

zi_count RMSE, MAE, and Pearson r on the original count scale, comparing observed counts with predicted expected counts; zero-accuracy and Brier score are computed for the zero/nonzero gate

multi_proportion Aitchison distance in CLR space, z-scored CLR RMSE, and decoded simplex MAE

When conformal scores are present in the fit, `conformal_coverage_95` is reported for continuous, count, ordinal, and proportion traits. This is the empirical fraction of test predictions inside the same trait-level conformal radius around each prediction. It uses one score per trait, not cell-specific or Mondrian bounds. Cross-validation and model reports can also report coverage of their actual per-cell bounds.

When the fit includes a baseline, baseline metrics are appended with `method = "baseline"` for direct comparison. This uses the baseline already stored in the fit and its selected defaults; `evaluate()` has no argument for alternative baseline settings. To compare another baseline, construct it explicitly in a separate evaluation workflow.

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)
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$X_scaled`).
`splits`	list (output of `make_missing_splits`).
`pred_se`	numeric matrix of prediction SEs on the scale of the predictions being evaluated. For a `pigauto_pred`, this means the latent predictions in `pred$imputed_latent`; it does not accept original-scale SEs as an override. Used for empirical coverage of the symmetric $\hat{y} + / - 1.96SE$ interval on the latent scale for continuous and

proportion traits. This coverage_95 is distinct from the conformal bound coverage metric reported by evaluate(). When pred is a pigauto_pred, an explicit pred_se takes precedence; otherwise its se_latent is used when available.

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

Predictions must exclude the cells being scored from both fitting and prediction inputs. Use predict(fit, evaluation = TRUE) or evaluate(fit). Plain predict(fit) uses observed values for production imputation and must not be scored as held-out data. Predictions with provenance are rejected if scored indices were not held out. For raw matrices or older prediction objects this cannot be checked; callers must ensure the cells were held out themselves.

Metrics by trait type:

continuous RMSE on the latent scale, Pearson r, and coverage_95: empirical coverage of the symmetric $\hat{y} + / - 1.96SE$ interval on that scale, when SE is supplied

proportion RMSE on the latent scale, Pearson r, and coverage_95 on that scale, when SE is supplied

count RMSE, MAE, and Pearson r on the z-scored log1p latent scale

ordinal RMSE on the z-scored latent scale, Spearman rho

binary Accuracy, Brier score

categorical Accuracy

zi_count RMSE, MAE, and Pearson r on the original count scale, comparing observed counts with predicted expected counts; zero-accuracy and Brier score on the zero/nonzero gate

multi_proportion Aitchison distance in CLR space, RMSE on the z-scored CLR latent scale, and MAE on the decoded simplex

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)
# Equal inputs illustrate the interface and output shape only: RMSE is zero
# by construction, correlation is unavailable with one cell per split, and
# this is not a performance evaluation.
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("estimate", "fixed_1", "cv", "bayes"),
  lambda_fixed = NULL,
  discrete_lambda = c("estimate", "fixed_1"),
  em_iterations = 0L,
  em_tol = 0.001,
  em_offdiag = FALSE,
  joint_solver = c("inhouse", "rphylopars"),
  predict_method = c("auto", "exact", "per_column"),
  joint_refine_iter = 0L,
  predict_route = NULL,
  seed = NULL
)
```

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 a joint baseline. `"hard"` (default) thresholds binary proportions at 0.5 and uses the most frequent categorical class. `"soft"` preserves species-level proportions and combines truncated-Gaussian liability expectations, so intermediate class

frequencies retain fractional evidence. Only relevant to multi-observation binary/categorical joint baselines.

`lambda_mode` character. Pagel-lambda mode for the CONTINUOUS-FAMILY baseline (continuous, count, proportion, zi_count magnitude columns; NOT ordinal, which the threshold-joint path treats as a liability column via `estep_liability_ordinal()` – see "Per-type lambda dispatch" in Details). `"estimate"` (default) fits each continuous-family column's own Pagel's lambda via profile REML; when the joint MVN or threshold-joint baseline fires, this now runs INSIDE that joint fit (`R/joint_mvn_solver.R`'s per-column `lambda_cols`) rather than being discarded in favour of a separate per-column re-fit. `"fixed_1"` preserves the classic Brownian correlation matrix (lambda = 1 everywhere). `"cv"` and `"bayes"` have no joint analogue and force the per-column BM path for continuous-family columns (as before); binary/ordinal/categorical/ zi_gate columns are unaffected by `lambda_mode` in every case; their lambda is governed by `discrete_lambda` (below), and is 1 when `discrete_lambda = "fixed_1"`. **Covariate caveat**: when `data$covariates` is supplied, the per-column path switches to `bm_impute_col_with_cov()`, which accepts a numeric lambda or `"estimate"` but not `"cv"` / `"bayes"`. When `lambda_fixed = NULL`, requesting either unsupported mode issues a warning and fits these columns at lambda = 1. An explicit `lambda_fixed` takes precedence.

`lambda_fixed` optional named numeric vector (names = latent column names, i.e. `colnames(data$X_scaled)`) giving a FIXED lambda per continuous-family column, overriding `lambda_mode` entirely for those columns. To replay a previous fit's per-trait lambda without re-estimating it, supply that fit's `$lambda_per_trait`. Columns not present in `lambda_fixed` keep their lambda = 1 default. `NULL` (default) means every continuous-family column follows `lambda_mode` normally.

`discrete_lambda`

character, `"estimate"` (default) or `"fixed_1"`. With `"estimate"`, the binary, ordinal and zero-inflated-count gate liability columns of the threshold-joint baseline, and the one-vs-rest column of each categorical class, get their own Pagel's lambda, so a weak-signal discrete trait shrinks toward its prevalence instead of copying close relatives. `"fixed_1"` restores the previous behaviour (binary and ordinal at lambda = 1, or at the shared joint lambda on the "exact" route; categorical at 1). In the reported simulated Brownian-motion and OU comparison (n = 100 to 1000, 30 percent missing; see the changelog), estimating discrete lambda alone added about 0.01 to 0.04 discrete accuracy at Pagel's lambda of 0.3 or below; together with `safety_floor = FALSE` and `phylo_signal_gate = FALSE` (also the defaults) the gain was 0.04 to 0.12, and binary and categorical accuracy matched or beat BACE's. Costs: with lambda near 1 and n >= 1000, accuracy -0.002 to -0.005 and Brier score +0.010 to +0.016; on AVONET300's categorical traits, Brier +0.014 to +0.020 with accuracy unchanged or better. Ordinal accuracy still trails BACE's at lambda = 0.3. The categorical one-vs-rest lambdas are not reported in `$lambda_per_trait`. Recorded in the returned list and in a fit's `model_config`; binary and ordinal lambdas are replayed by `lambda_fixed`.

`em_iterations` integer. Maximum passes through each liability-prior refinement loop for binary, ordinal, and one-vs-rest categorical baselines. Default `0L` disables iterative refinement and uses a single-pass baseline. `1L` also permits only the initial pass;

2L permits at most one feedback pass. After each pass, estimated liability variances inform the next prior. Refinement can stop early on convergence or a later fit failure. A categorical pass may require K one-vs-rest fits; predict_method = "auto" may fit two route candidates. Continuous-only joint MVN fits are unaffected.

em_tol — numeric. Relative-Frobenius convergence tolerance for liability-prior refinement (default 1e-3). The binary/ordinal threshold-joint loop compares the variance diagonal by default, or the full covariance matrix when em_offdiag = TRUE. Categorical one-vs-rest refinement compares the vector of class variances.

em_offdiag — logical. When TRUE and em_iterations >= 2L, binary/ordinal liability priors use conditional-MVN means and SDs based on the other traits and the full covariance matrix from the preceding pass. With the default FALSE, refinement updates per-trait prior SDs only. This option does not change categorical one-vs-rest refinement.

joint_solver — character. Solver for the joint MVN, threshold-joint, and categorical one-vs-rest baselines. "inhouse" (default) uses pigauto's joint solver. "rphylopars" delegates to Rphylopars::phylopars(), using its lambda model under lambda_mode = "estimate", with automatic fallback to "inhouse" on failure or implausible output. This does not affect routes outside these joint fits. "cv"/"bayes" lambda modes instead use per-column numeric fits. See fit_baseline for dispatch and fallbacks.

predict_method — character. Prediction route for the in-house joint solver. "auto" (default) fits the baseline once with the "exact" route and once with the "per_column" route, then for each TRAIT (each trait_map entry; a categorical trait's K latent columns and a zi_count trait's gate + magnitude columns are chosen together) picks whichever route has the lower loss on that trait's cells: mean squared error on the z-scored latent scale for continuous/count/proportion/ordinal/zi_count magnitude, mean log-loss of plogis(mu) against the observed 0/1 truth for binary/zi_count gate, and mean multinomial log-loss for categorical. **Which cells the choice sees**: a trait's held-out validation rows (species in multi-observation data) are split into two halves (seeded via seed, or the ambient RNG state when seed = NULL) when the trait has at least 38 of them and the two candidate fits predict it differently. The route choice sees only half A for this outer exact-versus-per-column choice; half B is reserved for gate calibration and conformal scoring. This separation applies only to that outer choice. For ordinal traits, each candidate fit's inner threshold-joint-versus-BM selector has already used the full validation split, including half B. Below 38 rows each half would keep fewer than the 19 cells needed for a 95 percent split-conformal interval, so the same rows both choose the route and calibrate; when both fits agree, no choice is made and all rows calibrate. With fewer than 5 validation cells, or on a genuine tie, a trait requests "exact". With splits = NULL, no validation comparison is possible, so every trait requests exact prediction; unsupported or unusable exact fits may still fall back to per-column prediction. The chosen route per trait is returned as $predict_method_by_trait. "exact" uses the full cross-trait conditional mean and variance of vec(L) ~ MVN(0, Sigma %x% R(lambda_block)) in the sparse precision form (Hadfield & Nakagawa, 2010), with each column centred at its own GLS phylogenetic mean at lambda_block before the solve so the mean and covariance models use the same phylogenetic correlation. Discrete

liability columns (binary, zi gate, ordinal-via-OVR synthetic columns) share `lambda_block` under `"exact"` rather than staying fixed at lambda = 1, so the whole joint fit uses one internally-consistent `R(lambda_block)`. Falls back to `"per_column"` above roughly 20000 unknown cells (roughly 4000 species at 5 traits), on a singular/unusable Sigma, or when fewer than 2 joint columns or no Henderson sparse precision are available; the fallback prints a one-time `message()` (not a warning) per R session when `predict_method` was left at its default, or a `warning()` every time when `"exact"` was requested explicitly. `"per_column"` retains the original per-column conditional prediction route (each column's own posterior, no cross-trait borrowing in the prediction step; `se` is that column's own conditional SE). Neither option changes covariance estimation or the `"rphylopars"` solver.

`joint_refine_iter`
: integer, default `0L`. Maximum passes of optional cross-trait refinement of joint-baseline cell imputations using the estimated covariance matrix. `0L` disables this refinement. The covariance step must shrink each iteration; otherwise the last accepted iterate is restored and `$diverged` is set. This has no effect on traits predicted by the `"exact"` route, which returns the analytic conditional directly. Under `predict_method = "auto"`, it can therefore affect only traits routed to `"per_column"`. Evaluate this opt-in setting on held-out cells from the intended data.

`predict_route`
: optional named character vector (names = trait names, i.e. `vapply(data$trait_map, function(tm) tm$name, character(1))`; values `"exact"` or `"per_column"`). Forces the prediction route for each named trait, bypassing the `"auto"` validation-loss comparison entirely (and overriding `predict_method`, if a value other than `"auto"` was also supplied). Traits not named default to `"exact"`. A name that matches no trait in `data$trait_map` is ignored, with a warning (once per call). Used internally to replay a validation-split `"auto"` choice onto a production refit that has no validation cells (e.g. `fit_pigauto()`'s `baseline_full`, fit with `splits = NULL`) so that refit reuses the SAME per-trait route rather than re-deciding with zero validation evidence. `NULL` (default) disables forcing; `predict_method` governs normally.

`seed`
: optional integer. Seeds the deterministic half-A (route choice) / half-B (reserved for `fit_pigauto()`/`impute()`'s gate calibration and conformal scoring) split of each trait's validation cells under `predict_method = "auto"` with real validation cells (see ‘"auto"’ above). `NULL` (default) leaves the split to whatever the ambient RNG state is at call time, matching this package's other optional `seed` arguments. Ignored when `predict_method` is `"exact"` / `"per_column"`, when `predict_route` is supplied, or when there are no validation cells.

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 meet its route requirements. The default in-house joint solver does not require Rphylopars; that package is an optional alternative solver. 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.

Covariates and baseline routes: `data$covariates` are used by the per-column BM path (a separate Brownian-motion fit for each eligible numeric trait). Joint MVN paths fit eligible numeric traits together; threshold-joint paths add discrete liabilities to a joint fit. Joint MVN and threshold-joint fits both omit the covariate design and emit a warning. One-versus-rest categorical fits omit covariates but do not trigger the separate baseline warning themselves. In a mixed fit, another selected joint route can still trigger it. Each category gets its own class-versus-rest fit. In a mixed fit, per-column numeric BM fallbacks can still use covariates for the columns they handle. The warning applies to the selected joint baseline only. GNN covariate use is separate and occurs only when a GNN correction is enabled in `fit_pigauto()`. These inputs do not guarantee improved predictions.

Per-type lambda dispatch: `lambda_mode` controls the continuous-family columns (continuous, count, proportion, and zi_count magnitude). Under `"estimate"`, the selected joint or per-column numeric route estimates each eligible column's lambda; `"cv"` and `"bayes"` use the per-column route and are unsupported by the covariate-aware per-column fallback, which uses lambda 1 and warns when no explicit `lambda_fixed` was supplied. Ordinal, binary, and zi_count gate liabilities are handled by the discrete-liability rules instead of this numeric-column setting. `discrete_lambda` controls their route-specific behavior: estimated values may be used in threshold-joint fits, while exact joint predictions share the fit's `lambda_block`. Label-propagation and categorical one-vs-rest routes have additional reporting limits. Covariance estimates alone do not imply that returned predictions used cross-trait conditioning; see `predict_method` below.

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) with route-specific uncertainty values. Brownian-motion conditional-MVN routes return conditional standard deviations; the zero-inflated-count constant magnitude fallback returns its observed-sample SD. Decoded threshold, one-vs-rest categorical, and label-propagation routes may store zero placeholders, which do not indicate certainty. A zero can also be genuine conditional uncertainty for a numeric observation. These baseline values are distinct from the type-specific uncertainties in `pred$se`.

path Named character vector, one entry per `trait_map` entry (name = trait/group name), recording which dispatch branch produced that trait's baseline: `"joint_mvn"` (continuous-family joint MVN), `"threshold_joint"` (binary, ordinal, and continuous-family liability joint), `"ovr_categorical"` (K one-vs-rest categorical fits), `"per_column_bm"` (per-column Brownian-motion conditional MVN, including lambda-aware fits and an ordinal candidate selected by fit-specific validation MSE against threshold-joint), `"multi_proportion_bm"` (component-wise BM on CLR columns), `"label_propagation"` (phylogenetic label propagation for eligible discrete columns), `"zi_gate_lp"` (label propagation for a zero-inflated count's gate column), or `"zi_mag_constant"` (global mean/sd fallback for a zero-inflated count's magnitude column when fewer than 5 non-zero observations exist). For `zi_count` traits (2 latent columns: gate then magnitude), `path` reports the GATE column's dispatch – the magnitude column can independently land on `"joint_mvn"`, `"threshold_joint"`, `"per_column_bm"`, or `"zi_mag_constant"` and is not separately surfaced here.

lambda_per_trait Named numeric vector with one entry per latent column, named by `colnames(data$X_scaled)`. Entries report values available for each column; exact joint prediction also uses the shared `lambda_block`; categorical one-vs-rest fit lambdas are not reported in this vector. For covariate-aware columns where lambda could not be estimated, the effective fallback is 1 and the latent column names are recorded in the vector's `lambda_unestimated` attribute.

lambda_block Numeric scalar, the shared lambda used internally by the joint fit for covariance estimation and cross-trait conditional calculations; `NA` when no joint fit ran.

lambda_mode Character, echoes the resolved `lambda_mode` argument.

predict_method_used Character scalar, `"exact"`, `"per_column"`, or `"auto"`. Under a concrete `predict_method` (`"exact"` / `"per_column"`), aggregated across every joint fit that ran: `"exact"` only if every joint fit achieved exact; `"per_column"` if any fell back, if none ran (a pure per-column baseline), or if every joint fit used `joint_solver = "rphylopars"` (no exact/per_column dichotomy there). Under `predict_method = "auto"` (or a forced `predict_route`), always `"auto"`; see `$predict_method_by_trait` for the resolved per-trait routes.

predict_method_by_trait Named character vector (names = trait names), one entry per `trait_map` entry, giving the route (`"exact"` or `"per_column"`) actually used for that trait's cells. Under a concrete `predict_method`, every entry equals `predict_method_used`. Under `"auto"` (or a forced `predict_route`), the per-trait validation choice (see `predict_method`'s `"auto"` entry above).

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

Fits a phylogenetic baseline, with an optional attention-based graph neural network (GNN) contribution. The GNN is disabled by default (`gnn = FALSE`). Baseline routes depend on trait type, prediction settings, and fit eligibility: they include continuous-family Brownian motion, threshold/liability joint fits, categorical one-vs-rest fits, label propagation, and component-wise BM on transformed compositions. See `fit_baseline` for route eligibility and fallbacks. All eight trait types share a latent representation. The internal torch class `ResidualPhyloDAE` refers to residual skip connections; its output is a full prediction, not a statistical residual.

Usage

```
fit_pigauto(
  data,
  tree,
  splits = NULL,
  graph = NULL,
  baseline = NULL,
  gnn = FALSE,
  baseline_full = 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", "mondrian"),
  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 = FALSE,
  phylo_signal_gate = FALSE,
  phylo_signal_threshold = 0.2,
  phylo_signal_method = c("lambda", "blomberg_k"),
  min_val_cells = 20L,
  lambda_mode = c("estimate", "fixed_1", "cv", "bayes"),
  discrete_lambda = c("estimate", "fixed_1"),
```

```
  joint_solver = c("inhouse", "rphylopars"),
  predict_method = c("auto", "exact", "per_column"),
  joint_refine_iter = 0L,
  verbose = TRUE,
  seed = NULL
)

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

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.
gnn	logical. When TRUE, trains the attention-based GNN correction as described above. When FALSE (default), no GNN is constructed or trained and fit_pigauto() makes **zero** torch:: calls – the fit is the phylogenetic baseline alone, optionally re-weighted against a grand-mean floor. safety_floor and phylo_signal_gate keep their usual semantics: the blend collapses to $r_{BM} \cdot \mu_{BM} + r_{MEAN} \cdot \mu_{MEAN}$ (the GNN corner degenerates to the baseline, so its calibrated weight is folded into r_{BM}). With the defaults (safety_floor = FALSE, phylo_signal_gate = FALSE) this is the pure traditional-stats arm. conformal_method = "mondrian" is not supported when gnn = FALSE (its locality statistic conditions on a calibrated GNN prediction surface that does not exist here) and raises an error. User covariates enter the GNN encoder/refinement and an additive fixed-effect term when gnn = TRUE. A selected per-column numeric BM baseline (a separate Brownian-motion fit for each numeric trait) can also use them when gnn = FALSE. Joint MVN and threshold-joint baselines fit traits together, omit covariates, and emit a separate baseline warning. One-versus-rest categorical baselines also omit them but do not trigger that separate warning themselves. In a mixed fit, another selected joint route can still trigger it. Covariate use is route-dependent and does not imply improved predictions. Without a validation split (splits = NULL) the fit is pure baseline ($r_{BM} = 1$) and carries no conformal scores. In the reported simulated and real-data comparison at the evaluated baseline settings, GNN-on did not lower imputation error and took 20 to 90 times longer to fit (see the changelog for that study). These results do not establish performance on new data.
baseline_full	list (output of fit_baseline with splits = NULL) or NULL. Only used when gnn = FALSE: the production-mode baseline, fit on ALL observed cells (no val/test hold-out), used by predict.pigauto_fit() for ordinary (non-evaluation) predictions. baseline itself always stays the held-out fit and is what every scorer (val_rmse, test_rmse, conformal_scores, evaluate()) reads. When NULL (default) and gnn = FALSE, it is computed internally via fit_baseline(data, tree, splits = NULL, ...); ignored when gnn = TRUE unless explicitly supplied (e.g. by impute()), in which case it is stored but not used for training.

`hidden_dim` integer. Hidden layer width (default `64`).

`k_eigen` integer or `"auto"`. Number of spectral node features. The default `"auto"` scales with tree size; an integer specifies the dimension directly.

`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 bandwidth for a Gaussian phylogenetic-distance bias added to the attention scores.

`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. 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 within-row functional coupling not represented by the selected baseline. Assess this opt-in architecture with held-out evaluation on the intended data. Default `FALSE` omits this cross-trait block.

`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`). Gate calibration and conformal scoring use the same number of steps so the calibrated surface matches predict.

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 residual score is estimated from held-out validation cells. `"split"` (default, backward-compatible) takes a sample quantile; `"bootstrap"` averages quantiles from `conformal_bootstrap_B` bootstrap resamples; and `"mondrian"` stratifies validation cells by phylogenetic sampling locality before estimating a score per stratum. These choices support nominal held-out diagnostics, not package-certified coverage. `"mondrian"` is single-observation only because its locality is species-level. If either stratum is too small, that trait uses the global `"split"` score instead.

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

conformal_split_val
: logical. Default `FALSE`: gate selection and residual-score estimation share validation cells. When `TRUE`, each eligible latent column separates those tasks into calibration and residual-estimation subsets. This separation does not certify package-wide coverage. Columns without enough validation cells retain the shared-data path. With either setting, use `cross_validate()` and report held-out diagnostics for the intended data.

gate_method
: character. How the per-trait calibrated gate is chosen. `"single_split"` 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"` (default) partitions validation 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. Validation calibration remains distinct from assessment on held-out test cells. "median_splits" repeats the split procedure before taking a componentwise median. Assess calibration settings on held-out data.

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. Default FALSE. When TRUE, calibration searches a three-way convex combination of phylogenetic baseline, GNN, and grand-mean candidates. The selected candidate is no worse than the mean-corner candidate on the validation cells under the calibration metric. An additive fixed-effect term remains when present, so that corner is not always a pure grand mean; multi_proportion groups are excluded. This is a validation-set safeguard, not a guarantee for future data. With FALSE, calibration uses the baseline/GNN line with r_cal_mean = 0.

phylo_signal_gate
logical. Default FALSE. When TRUE, estimate each trait's Pagel's lambda from training-observed cells, excluding validation and test cells. After model training and calibration, traits with lambda < phylo_signal_threshold use the mean corner when safety_floor = TRUE, or the Brownian baseline corner otherwise. This changes the calibrated blend; it does not skip GNN training. Requires **phytools**; without it, no signal-based override is applied. Evaluate the choice on held-out data.

phylo_signal_threshold
numeric, default 0.2. Traits with Pagel's λ below this value are routed to the mean corner when the safety floor is enabled, or to the Brownian baseline corner otherwise.

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 20: with fewer than 19 cells the split-conformal level $n/(n+1)$ is below 0.95; with 38 or fewer cells the conformal score is the largest validation residual; and with very few cells 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 continuous-family baseline columns (continuous, count, proportion, and zi_count magnitude). "estimate" (default) estimates each eligible column's lambda within its selected joint or per-column fit. "fixed_1" uses lambda = 1. "cv" and "bayes" use alternative per-column estimators and cannot use the joint numeric route. Discrete-liability lambda

is controlled separately by discrete_lambda. Exact joint prediction and optional cross-trait cell refinement also use a shared lambda_block for cross-trait computations, without replacing continuous-family column-specific lambdas. A covariate-aware per-column BM fit accepts "estimate" or fixed lambda; unsupported "cv"/"bayes" requests warn and use lambda = 1. See fit_baseline for route details and fixed-lambda controls.

discrete_lambda

character, "estimate" (default) or "fixed_1". Controls lambda for binary, ordinal, and zi_count gate liabilities in threshold-joint fits and for each categorical one-vs-rest fit. Exact joint predictions share lambda_block; categorical class-fit lambdas are not reported in $lambda_per_trait. See fit_baseline for route-specific behavior and the evaluated benefits and costs. Stored in the fitted model configuration.

joint_solver character. Solver for the joint MVN, threshold-joint, and categorical one-vs-rest baselines. "inhouse" (default) uses pigauto's joint solver. "rphylopars" delegates to Rphylopars::phylopars(), using its lambda model under lambda_mode = "estimate", with automatic fallback to "inhouse" on failure or implausible output. This does not affect routes outside these joint fits. "cv"/"bayes" lambda modes instead use per-column numeric fits. See fit_baseline for dispatch and fallbacks.

predict_method character. Prediction route for the in-house joint solver, passed to fit_baseline. "auto" (default) fits the baseline with both the "exact" and "per_column" routes and picks, per trait, whichever has the lower loss on that trait's validation cells (mean squared error on the z-scored latent scale for continuous/count/proportion/ordinal/zi_count magnitude; mean log-loss for binary/zi_count gate; mean multinomial log-loss for categorical), falling back to "exact" on a tie or when fewer than 5 validation cells belong to that trait. With no validation split, exact prediction is requested but route-specific per-column fallbacks remain possible. "exact" uses the full cross-trait conditional mean and variance of vec(L) ~ MVN(0, Sigma %x% R(lambda_block)) (Hadfield & Nakagawa, 2010 sparse precision form), each column GLS-mean-centred at lambda_block before the solve; discrete liability columns share lambda_block under this route (see lambda_mode above). Falls back to "per_column" above roughly 20000 unknown cells (roughly 4000 species at 5 traits), on a singular/unusable Sigma, or when fewer than 2 joint columns or no Henderson sparse precision are available; prints a one-time message() per R session when predict_method was left at its default, or a warning() every time when "exact" was requested explicitly. "per_column" retains the original per-column conditional prediction route (no cross-trait borrowing in the prediction step). None of the three options change covariance estimation or the "rphylopars" solver. The route actually used is recorded in $model_config$predict_method_used ("exact", "per_column", or "auto") and, per trait, in $model_config$predict_method_by_trait. Under "auto" with real validation cells, $model_config $route_val_n / $score_val_n record, per trait, how many of its validation cells chose the route versus were reserved for gate calibration and conformal scoring (see fit_baseline's predict_method docs); both are NULL under an explicit "exact"/"per_column" request.

joint_refine_iter

integer, default 0L. Maximum passes of optional cross-trait refinement of joint-baseline cell imputations using the estimated covariance matrix. 0L disables this

refinement. The covariance step must shrink each iteration; otherwise the last accepted iterate is restored and `$diverged` is set. This has no effect on traits predicted by the `"exact"` route, which returns the analytic conditional directly. Under `predict_method = "auto"`, it can therefore affect only traits routed to `"per_column"`. Evaluate this opt-in setting on held-out cells from the intended data.

`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.

`x` A fitted pigauto model.

`...` Ignored. Included for compatibility with the base `print()` generic.

Details

Blend formulation: The neural blend is $\hat{x} = (1 - r)\mu + r\delta$, where μ is the baseline, δ is the model's direct prediction, and $r = \sigma(\rho) \times \text{cap}$ is a per-column gate. A zero neural weight returns the baseline only when no mean or covariate shortcut contributes; validation calibration does not guarantee target-set error.

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. On randomly masked training cells, the objective combines the type-specific loss for $\delta + fixed_effect$, 0.1 times the type-specific loss for the blended prediction plus fixed effect, `lambda_shrink` times MSE($\delta - \mu$), and `lambda_gate` times 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, and proportion MSE on the latent scale

multi-proportion MSE in centred log-ratio (CLR) space

binary binary cross-entropy with logits

zero-inflated count binary cross-entropy for the nonzero gate, plus MSE for the conditional magnitude

categorical cross-entropy over the K class logits

Value

An object of class `"pigauto_fit"`.

The print method returns `x` invisibly.

Calibration at small n

Conformal bounds use the empirical `(1 - alpha)` quantile of $|y - \hat{y}|$ on held-out validation cells. They are nominal held-out diagnostics and do not certify package-wide coverage.

Small validation sets make both the residual quantile and gate selection unstable. The `min_val_cells` warning identifies traits where this is especially relevant. `conformal_split_val = TRUE` can separate gate selection from residual estimation for eligible columns, but it does not remove the small-sample limitation. Use `cross_validate()` on the intended data and report the resulting held-out diagnostics rather than treating an interval target as certified.

Printing

Prints configuration, losses, gates and signal summaries. A printed signal label does not verify that a mean fallback was enabled. The `...` argument is currently unused.

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)
```

`get_imputed_datasets` *Extract the imputed datasets from a multiple-imputation object*

Description

Returns the list of completed datasets stored in a multiple-imputation object, equivalent to `mi$datasets`.

Usage

```
get_imputed_datasets(mi)
```

Arguments

`mi`	A `pigauto_mi` object from `multi_impute()`, `multi_impute_analysis()`, or `multi_impute_trees()`.

Value

A list of `mi$m` input-shaped data frames. Observed cells are preserved; missing trait cells vary across datasets according to the draw mechanism. Check `mi$draws_method` before analysing them: `"posterior"` draws support the documented pooling workflow via `with_imputations()` and `pool_mi()`, while `"conformal"` and `"mc_dropout"` draws are prediction diagnostics that describe the spread of plausible imputed values and are refused by the pooling functions. Datasets from `multi_impute_trees()` are tree-sensitivity diagnostics; `mi$tree_index` records the tree behind each one.

See Also

`multi_impute()`, `with_imputations()`, `pool_mi()`, and `completed_data()` for the single completion returned by `impute()`.

Examples

```
## Not run:
mi <- multi_impute(traits, tree, m = 20)
datasets <- get_imputed_datasets(mi)

## End(Not run)
```

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

Description

One-call interface for preprocessing, phylogenetic baseline fitting, and prediction. The default `gnn = FALSE` uses the baseline alone. Set `gnn = TRUE` to also train a GNN and calibrate its contribution. 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,
  k_eigen = "auto",
  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("estimate", "fixed_1", "cv", "bayes"),
  discrete_lambda = c("estimate", "fixed_1"),
  joint_solver = c("inhouse", "rphylopars"),
  predict_method = c("auto", "exact", "per_column"),
  joint_refine_iter = 0L,
  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 = FALSE,
  phylo_signal_gate = FALSE,
  phylo_signal_threshold = 0.2,
  phylo_signal_method = "lambda",
  conformal_split_val = FALSE,
  gnn = FALSE,
  ...,
  ordinal_levels = NULL
)

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

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).

`k_eigen` integer or `"auto"`. Number of spectral node features used to build the phylogenetic graph. The default `"auto"` scales the dimension to tree size; an integer overrides that choice.

`log_transform` logical. Log-transform every continuous trait whose observed values are all positive (default `TRUE`), then back-transform the imputations. Set it to `FALSE` when traits are already on a log scale (for example log body mass): otherwise they are logged a second time and imputed on a log(log) scale, which can bias analyses done on the log scale. With a mix of raw and already-logged traits, log the raw ones yourself and set `FALSE`. In a real-data check (PanTHERIA, 4,027 mammal species, four already-logged traits, posterior draws), the default shifted Rubin-pooled regression slopes by up to 3.9 standard errors of the complete-data slope, in a direction that depended on which values were missing; with `FALSE` the largest shift was 1.8.

`missing_frac` numeric. Fraction of observed cells held out for validation/test evaluation (default `0.25`). Must be in `[0, 1)`. `0` adds no extra missingness (no val/test holdout).

`n_imputations` integer. Number of stochastic prediction-diagnostic draws (default `1`). Must be a positive integer. With multiple draws, prediction adds independent Gaussian perturbations per latent cell using the selected baseline SD; GNN dropout also varies across runs when `gnn = TRUE`. `pool_method` combines decoded draws for the returned point prediction: `completed` is one data frame, not a list of completed datasets. Despite this legacy argument name, these are not joint phylogenetic posterior draws or validated imputations for downstream inference. See `multi_impute` and the multiple- imputation article for the separate workflows and their limits.

`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. Rows are aligned to the same species / observation order as `traits` after tree matching (single-obs: rownames vs tip labels; multi-obs: `species_col` if present, otherwise input-row order). Matching `nrow` alone does not pair species to environment. Covariate use depends on the selected route: a covariate-aware per-column BM path may use them. Joint MVN and threshold-joint baselines omit them and emit a separate baseline warning. One-versus-rest categorical baselines also omit them but do not trigger the separate baseline warning themselves. In a mixed fit, another selected joint route can still trigger it. With `gnn = TRUE`, covariates also enter the GNN and its additive linear component. `predict_method = "per_column"` selects predictions within a joint fit; it does not activate covariate regression. In multi-observation data the baseline aggregates observations and covariates to species level; with `gnn = FALSE` it does not predict observation-specific covariate effects within a species. Default `NULL` (no covariates).

`epochs` integer. Maximum GNN training epochs (default `2000`). Must be a positive integer.

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. Predictions can vary between seeds; a fixed seed supports reproducibility, not a measure of predictive uncertainty.

multi_obs_aggregation — character. How to aggregate multiple observations per species before a joint 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 continuous-family baseline columns (continuous, count, proportion, and zi_count magnitude). "estimate" (default) estimates each eligible column's lambda within its selected joint or per-column fit. "fixed_1" uses lambda = 1. "cv" and "bayes" use alternative per-column estimators and cannot use the joint numeric route. Discrete-liability lambda is controlled separately by discrete_lambda. Exact joint prediction and optional cross-trait cell refinement also use a shared lambda_block for cross-trait computations, without replacing continuous-family column-specific lambdas. A covariate-aware per-column BM fit accepts "estimate" or fixed lambda; unsupported "cv"/"bayes" requests warn and use lambda = 1. See fit_baseline for route details and fixed-lambda controls.

discrete_lambda — character, "estimate" (default) or "fixed_1". Controls lambda for binary, ordinal, and zi_count gate liabilities in threshold-joint fits and for each categorical one-vs-rest fit. Exact joint predictions share lambda_block; categorical class-fit lambdas are not reported in $lambda_per_trait. See fit_baseline for route-specific behavior and the evaluated benefits and costs. Stored in the fitted model configuration.

joint_solver — character. Solver for the joint MVN, threshold-joint, and categorical one-vs-rest baselines. "inhouse" (default) uses pigauto's joint solver. "rphylopars" delegates to Rphylopars::phylopars(), using its lambda model under lambda_mode = "estimate", with automatic fallback to "inhouse" on failure or implausible output. This does not affect routes outside these joint fits. "cv"/"bayes" lambda modes instead use per-column numeric fits. See fit_baseline for dispatch and fallbacks.

predict_method — character. Prediction route for the in-house joint solver, passed to fit_baseline. "auto" (default) fits the baseline with both the "exact" and "per_column" routes and picks, per trait, whichever has the lower loss on that trait's validation cells (mean squared error on the z-scored latent scale for continuous/count/proportion/ordinal/zi_count magnitude; mean log-loss for binary/zi_count gate; mean multinomial log-loss for categorical), falling back to "exact" on a tie or when fewer than 5 validation cells belong to that trait. With no validation split, exact prediction is requested but route-specific per-column fallbacks remain possible. "exact" uses the full cross-trait conditional mean and variance of vec(L) ~ MVN(0, Sigma %x% R(lambda_block)) (Hadfield & Nakagawa, 2010 sparse precision form), each column GLS-mean-centred at lambda_block before the solve; discrete liability columns share lambda_block under this route (see lambda_mode above).

Falls back to "per_column" above roughly 20000 unknown cells (roughly 4000 species at 5 traits), on a singular/unusable Sigma, or when fewer than 2 joint columns or no Henderson sparse precision are available; prints a one-time message() per R session when predict_method was left at its default, or a warning() every time when "exact" was requested explicitly. "per_column" retains the original per-column conditional prediction route (no cross-trait borrowing in the prediction step). None of the three options change covariance estimation or the "rphylopars" solver. The route actually used is recorded in fit$model_config$predict_method_used ("exact", "per_column", or "auto") and, per trait, in fit$model_config$predict_method_by_trait. Under "auto" with real validation cells, fit$model_config $route_val_n / $score_val_n record, per trait, how many of its validation cells chose the route versus were reserved for gate calibration and conformal scoring (see fit_baseline's predict_method docs).

joint_refine_iter

integer, default 0L. Maximum passes of optional cross-trait refinement of joint-baseline cell imputations using the estimated covariance matrix. 0L disables this refinement. The covariance step must shrink each iteration; otherwise the last accepted iterate is restored and $diverged is set. This has no effect on traits predicted by the "exact" route, which returns the analytic conditional directly. Under predict_method = "auto", it can therefore affect only traits routed to "per_column". Evaluate this opt-in setting on held-out cells from the intended data.

em_iterations integer. Maximum passes through each liability-prior refinement loop for binary, ordinal, and one-vs-rest categorical baselines. Default 0L disables iterative refinement and uses a single-pass baseline. 1L also permits only the initial pass; 2L permits at most one feedback pass. After each pass, estimated liability variances inform the next prior. Refinement can stop early on convergence or a later fit failure. A categorical pass may require K one-vs-rest fits; predict_method = "auto" may fit two route candidates. Continuous-only joint MVN fits are unaffected. Passed to fit_baseline.

em_tol numeric. Relative-Frobenius convergence tolerance for liability-prior refinement (default 1e-3). The binary/ordinal threshold-joint loop compares the variance diagonal by default, or the full covariance matrix when em_offdiag = TRUE. Categorical one-vs-rest refinement compares the vector of class variances. Passed to fit_baseline.

em_offdiag logical. When TRUE and em_iterations >= 2L, binary/ordinal liability priors use conditional-MVN means and SDs based on the other traits and the full covariance matrix from the preceding pass. With the default FALSE, refinement updates per-trait prior SDs only. This option does not change categorical one-vs-rest refinement. Passed to fit_baseline.

pool_method character. How to pool stochastic prediction-diagnostic 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" takes the arithmetic mean. "mode" 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. When `TRUE`, cap back-transformed predictions for log-transformed continuous, count, and zi_count magnitude traits at `tm$obs_max * clamp_factor`, where `tm$obs_max` is the observed maximum on the original scale, recorded during preprocessing. This optional cap limits extrapolation after `exp()` or `expm1()`; it is not an uncertainty bound. Default `FALSE` leaves these predictions uncapped.

`clamp_factor` numeric scalar (>= 1). User-selected multiplier of the observed maximum for the upper cap when clamping is enabled. Default `5`. Ignored when `clamp_outliers = FALSE`.

`match_observed` character, one of `c("none", "pmm")`. 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"` leaves predictions unmatched.

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

`safety_floor` logical. Default `FALSE`. When `TRUE`, calibration searches a three-way convex combination of phylogenetic baseline, GNN, and grand-mean candidates. The selected candidate is no worse than the mean-corner candidate on the validation cells under the calibration metric. An additive fixed-effect term remains when present, so that corner is not always a pure grand mean; `multi_proportion` groups are excluded. This is a validation-set safeguard, not a guarantee for future data. With `FALSE`, calibration uses the baseline/GNN line with `r_cal_mean = 0`.

`phylo_signal_gate`, `phylo_signal_threshold`, `phylo_signal_method`
Pass-through to `fit_pigauto()`. `phylo_signal_gate = FALSE` by default; enable it to apply the documented training-data signal override after calibration. Assess the selected setting on held-out data.

`conformal_split_val`
logical. Default `FALSE`. When `TRUE`, the validation cells are split 50/50 per trait into a gate-calibration half and a conformal-scoring half, so the conformal quantile is not post-selected on the same cells the gate was tuned on. Costs interval width at small validation sizes; see `fit_pigauto` for details. Previously reachable only via `...`; exposed here for discoverability.

`gnn` logical. When `TRUE`, trains the attention-based GNN correction (see `fit_pigauto`). When `FALSE` (default), no GNN is constructed or trained – `impute()` makes no

torch/GPU calls, and the fit is the phylogenetic baseline alone (optionally reweighted against a grand-mean floor when `safety_floor = TRUE`). With the defaults (`safety_floor = FALSE, phylo_signal_gate = FALSE`) this is the pure traditional-stats arm. Production predictions (`result$completed`, `result$prediction`) use a full-data baseline fitted using all observed cells (`baseline_full`, with no validation/test hold-out); `result$evaluation` and every other scorer keep using the held-out `baseline`, so evaluation never leaks test cells. User `covariates` enter the GNN and its additive fixed-effect term, and may also enter a selected per-column numeric BM fallback, including when `gnn = FALSE`. Joint MVN and threshold-joint routes omit covariates and emit a separate baseline warning. One-versus-rest categorical routes also omit them but do not trigger that warning themselves. In a mixed fit, another selected joint route can still trigger it. Covariate use does not imply improved predictions. Without a validation split (`missing_frac = 0`) the fit is pure baseline and carries no conformal scores. In the reported simulated and real-data comparison at the evaluated baseline settings, GNN-on did not lower imputation error and took 20 to 90 times longer to fit (see the changelog for that study). These results do not establish performance on new data.

`...` Additional arguments passed by `impute()` to `fit_pigauto()`; ignored by `print.pigauto_result()`.

`ordinal_levels` named list giving the scientific order of ordinal levels, e.g. `list(status = c("LC", "NT", "VU", "EN", "CR"))`. Each named column is coerced to an ordered factor with exactly those levels before type detection (so no `trait_types` entry is needed), and `$completed` returns it as an ordered factor. Observed values absent from the declared levels are an error. Same semantics as the `ordinal_levels` argument of `read_traits`. Default `NULL`.

`x` A `pigauto_result` object returned by `impute()`.

Details

A single completed data set is useful for point prediction, but analysing it as if its imputed values were observed ignores imputation uncertainty and can also change estimates. See `multi_impute` and the multiple-imputation article for downstream workflows and their limitations.

Value

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

completed The input `traits` data.frame. Observed values are preserved. Missing cells are filled when the input row is included in the fit and a prediction is available; other missing cells remain `NA`.

imputed_mask Logical matrix in original input row order, with one column per input trait. The `species_col` column, when supplied, is excluded. `TRUE` marks an originally missing target in a row matched to the fitted data; a marked cell can remain `NA` if no prediction is available. Observed cells and rows outside the fitted tree are `FALSE`.

prediction A `pigauto_pred` object from `predict.pigauto_fit` containing raw model predictions for every cell (observed + missing), per-cell uncertainty (`$se` – see `predict.pigauto_fit`'s "What `$se` means per trait type" section; it is **not** uniformly a standard error), class probabilities, and conformal intervals. For an untransformed continuous trait, the usual global per-trait

score yields the same interval width across rows; it does not adapt to each species' number of observations. Other scoring methods or transformations can change this pattern.

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`).

check A compact `pigauto_check` preflight summary.

Invisibly returns `x`.

Trait type auto-detection

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

- `numeric`: continuous (automatically log-transformed when all observed values are positive)
- `integer`: count, except that columns with any negative observed value are detected as continuous
- `factor` with two levels: binary
- unordered `factor` with more than two levels: categorical
- ordered `factor` with two levels: binary; ordered factors with other numbers of levels: ordinal
- `character`: converted to a factor, then treated as binary or categorical according to its number of levels
- `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 a named `trait_types` override when the inferred type does not match the measurement. This 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.

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

With `safety_floor = TRUE`, post-training calibration grid searches a 3-way convex combination of the Brownian-motion baseline, the GNN delta, and the mean baseline. The simplex is sampled at step 0.05 (231 candidates per latent column). Because the corner `(0, 0, 1)` is always in the grid, the selected candidate cannot be worse than the mean-corner candidate on the validation cells under the calibration metric. An additive fixed-effect term is retained when present, so this is not always a pure grand mean; `multi_proportion` groups are excluded. This safeguard applies only to the validation cells used for calibration. The fit object stores: `r_cal_bm`, `r_cal_gnn`, `r_cal_mean` (each a named numeric of length `p_latent`), and `mean_baseline_per_col`.

With the default `safety_floor = FALSE`, calibration uses no mean term: `r_cal_mean = 0` and `r_cal_bm = 1 - r_cal_gnn`.

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 mask marks originally missing targets in fitted rows; some selected cells can remain `NA` when no prediction was available. To extract target values by trait without coercing mixed columns:

```
mask <- result$imputed_mask
imputed_values <- lapply(colnames(mask), function(nm) {
  result$completed[[nm]][mask[, nm]]
})
names(imputed_values) <- colnames(mask)
```

For successful fills only, also exclude `NA` values from each trait vector.

Common pitfall. If you call `impute()` on a fully observed trait matrix (no `NA`s among matched target rows), there is nothing to impute. `impute()` stops before graph construction or training and directs you to `cross_validate()` for predictive assessment. To exercise imputation, supply a real missing target cell.

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 about 78 percent Resident, 14 percent Partial, and 8 percent Full at n=300) can have its calibrated gate collapse onto a corner that predicts the majority class everywhere. Compare candidate settings, including `pool_method = "mode"`, with `cross_validate()` on the intended trait table rather than applying a generic setting.

Printing

`print.pigauto_result()` reports species, traits and types, the total and per-trait targeted-cell counts, and a route-specific inference note. Held-out test metrics are shown for continuous and count traits (RMSE and, when available, Pearson correlation), ordinal traits (Spearman correlation), and binary and categorical traits (accuracy). The printout does not show metrics for proportion, zero-inflated count, or multi-proportion traits; see `result$evaluation` or `evaluate_imputation()` for the available evaluation metrics. Completed values are stored in `result$completed`. Some targeted cells can remain `NA` when a prediction is unavailable. The method ignores `...` and returns `x` invisibly.

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")
suppressMessages(save_pigauto(fit, path))
pred <- predict(load_pigauto(path))
```

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

Description

Randomly selects a fixed number of originally observed trait units, designates them as missing, and splits them into validation and test sets. The requested holdout count is the floor of `missing_frac` times the number of eligible observed units. Mechanism-specific weights affect selection among those units; they are not per-cell Bernoulli probabilities. For a `"MAR_trait"` design, keep the driver trait observed if interpreting the design as missing at random. 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`). Existing missing values are excluded from the eligible holdout units. Values also determine weights for `"MAR_trait"` and `"MNAR"`.
`missing_frac`	numeric. Fraction of all (species, trait) cells to designate as missing (default `0.25`); the realised count is fixed from the eligible observed cells.
`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). Selection weights depend on the observed driver values; keep the driver trait observed when interpreting the design as MAR.

For `"MAR_phylo"`**:** `n_clades` (integer, number of selected clades; default 2), `p_clade` (numeric, relative selection weight for cells within selected clades; default 0.7), `p_base` (numeric, relative selection weight for other cells; default 0.1). The fixed overall holdout count is sampled without replacement using these weights.

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` means not artificially held out; it includes cells that were originally missing in `X`. Combine with `!is.na(X)` to identify observed training cells.

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`. Here `TRUE` means the value in `X` is observed (non-missing), and `FALSE` means it is an existing `NA`. This is different from the `mask` returned by `make_missing_splits()`, where `TRUE` means a cell was not artificially held out and can include cells that were already `NA`. Combine that split mask with `!is.na(X)` to identify observed training cells.

Usage

```
mask_missing(X)
```

Arguments

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

Value

Logical matrix, `TRUE` = observed (non-missing) in `X`.

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 mc_dropout draws returned here are experimental prediction diagnostics. The mc_dropout route uses independent Gaussian perturbations per latent cell, scaled by selected baseline SD, and adds GNN dropout variation only when the GNN is enabled. Do not use these datasets for downstream inference or Rubin pooling.

Usage

```
multi_impute(
  traits,
  tree,
  m = 100L,
  draws_method = c("auto", "conformal", "mc_dropout", "posterior"),
  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,
  gnn = FALSE,
  lambda_mode = c("estimate", "fixed_1", "cv", "bayes"),
  posterior_control = list(),
  ...,
  ordinal_levels = NULL
)

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

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:

"auto" (default; previously "conformal") Use "posterior" when the data fit its requirements (every trait continuous, one row per species, no covariates, no multi_proportion_groups); otherwise use "conformal" and print a message saying why and that the draws cannot be pooled. The chosen method is stored in result$draws_method. To get the previous default, set draws_method = "conformal".

"conformal" Use this method when "auto" falls back because the posterior sampler's requirements are not met, or when explicitly requested. 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; a nominal held-out conformal diagnostic does not establish that

these draws are suitable for downstream multiple-imputation inference. For continuous, count, ordinal, and proportion traits, when no finite conformal score is available, pigauto uses the reported `pred$se` as a scale input for fallback draws. For log-transformed continuous traits, log1p-transformed counts, and logit-transformed proportions, `pred$se` is reported on the original scale, but the draw is made on the transformed scale. Pigauto does not fully convert `pred$se` between these scales, so diagnostic draw spread may be too wide or too narrow. Fitted point predictions are unchanged. Binary and categorical traits use Bernoulli / categorical draws from predicted probabilities.

`"mc_dropout"` Add independent Gaussian perturbations per latent cell, scaled by selected baseline SD. This returns `M` stochastic draws; when the GNN is enabled, its network runs use dropout in training mode. These prediction-diagnostic draws are not joint phylogenetic BM posterior draws and are unsupported for downstream inference.

`"posterior"` Continuous traits only, one row per species, no covariates. Fit a Bayesian multivariate phylogenetic mixed model by MCMC and return `m` completions drawn from the posterior predictive distribution of the missing cells. No GNN is fitted. See "Posterior draws" below. In simulation (continuous traits, 30% missing completely at random, 200 datasets per cell, n = 100 to 1000, lambda 0.3 to 1, trait correlation 0 or 0.5), a Rubin-pooled downstream PGLS slope and phylogenetic correlation had 95% interval coverage of 0.925 to 0.975 in every cell. At lambda = 1 with trait correlation 0.5, slope coverage was 0.955, 0.940 and 0.965 at n = 100, 300 and 1000 (reference frequentist MI: 0.955, 0.920, 0.935). At n = 100 with correlated traits the pooled slope was biased by about -0.02 towards zero, with coverage still 0.96. These results concern only the stated simulation regimes and do not establish inferential performance elsewhere. The registered validation found five tested posterior slope regimes failed and three indeterminate; see the registered inferential results. The previous residual prior (`posterior_control = list(residual_prior = "iw")`) attenuated the relationship between correlated traits at lambda near 1: coverage 0.96, 0.88 and 0.75 at the same three n.

Conformal and MC-dropout draws perturb missing cells around a point prediction rather than drawing them jointly from their conditional distribution given the observed data. In a 16-regime simulation of a downstream phylogenetic regression (PGLS slope, two traits, 120 replicates per regime), conformal draws biased the pooled slope by -0.20 to -0.46, and the pooled 95% intervals covered the truth in 0 to 17% of replicates, in all 16 regimes; MC-dropout draws biased it by -0.03 to -0.38. For continuous traits, the documented route permits eligible `"posterior"` draws to be fitted with `with_imputations()` and pooled with `pool_mi()` under the conditions below. Software eligibility does not show that the resulting estimates and uncertainty are reliable. In the registered validation, five of eight tested posterior-slope regimes failed the prespecified criteria and three were indeterminate; none was supported. These results apply to those tested regimes. Other analyses need evidence for their model, data, and missingness process; see the registered inferential results.

`species_col` character or `NULL`. If set, marks the column in `traits` containing species identi-

fiers 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. Log-transform every continuous trait whose observed values are all positive (default `TRUE`), then back-transform the imputations. Set it to `FALSE` when traits are already on a log scale (for example log body mass): otherwise they are logged a second time and imputed on a log(log) scale, which can bias analyses done on the log scale. With a mix of raw and already-logged traits, log the raw ones yourself and set `FALSE`. In a real-data check (PanTHERIA, 4,027 mammal species, four already-logged traits, posterior draws), the default shifted Rubin-pooled regression slopes by up to 3.9 standard errors of the complete-data slope, in a direction that depended on which values were missing; with `FALSE` the largest shift was 1.8.

`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 fully observed environmental covariates. For `"mc_dropout"` and `"conformal"` diagnostic draws, forwarded to `impute()`: a selected per-column numeric BM baseline may use them even with `gnn = FALSE`. Joint MVN and threshold-joint baselines omit them and emit a separate baseline warning. One-versus-rest categorical baselines also omit them but do not trigger that warning themselves. In a mixed fit, another selected joint route can still trigger it. With `gnn = TRUE`, they also enter the GNN and additive linear component. Choosing `predict_method = "per_column"` within a joint fit does not enable covariate regression. The posterior sampler is a separate model route; see its eligibility requirements below. Default `NULL`.

`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.

`gnn` logical. Passed through to `impute()` / `fit_pigauto()`. When `TRUE`, the usual GNN correction is trained. When `FALSE` (default), no GNN is used (baseline-only fit; see `fit_pigauto()`). With `draws_method = "mc_dropout"` this uses independent latent-cell Gaussian perturbations scaled by baseline SD, with no GNN dropout; an explanatory message is printed for that call. These are not joint BM posterior draws; `draws_method = "conformal"` is unaffected. Covariate use is route-dependent as described above; the warning under `gnn = FALSE` does not rule out covariate-aware per-column BM predictions/draws.

`lambda_mode` character. Pagel-lambda mode for the BM baseline, forwarded to `impute()` / `fit_pigauto()`. `"estimate"` (default) fits a per-trait Pagel's lambda on each continuous-family (BM-eligible) latent column. Discrete traits estimate their own lambda by default (`discrete_lambda = "estimate"`, passed through ...

to impute()); set discrete_lambda = "fixed_1" to retain lambda = 1 for their baseline. Joint exact prediction can also use a shared lambda_block for its cross-trait conditional. "fixed_1" preserves the pre-lambda Brownian correlation matrix everywhere; "cv" and "bayes" are alternative per-column estimators. See fit_pigauto() for the full contract, including the predict_method = "exact" / joint_refine_iter > 0 interaction with lambda_block.

posterior_control
: list of settings for draws_method = "posterior" (ignored otherwise). Elements, with defaults:

  n_chains 4L. Number of MCMC chains. Chain 1 starts at the per-trait REML Pagel's lambda clamped to [0.02, 0.98] (up to 2,000 tips; above that, or if the REML fit fails, at lambda = 0.9), and the other chains start from values dispersed around it. mi$posterior$start holds the REML lambda and chain 1's lambda before clamping.

  n_iter 5000L. Sweeps per chain after burn-in.

  burnin 1000L. Burn-in sweeps per chain (also used to tune the Metropolis step sizes).

  thin Default: n_iter integer-divided by ceiling(keep_draws / n_chains), so that the chains together keep keep_draws sweeps.

  keep_draws 1000L. Target number of kept posterior predictive draws across chains, used for the per-cell intervals. Must be at least m. The default thin keeps at least this many whenever n_iter is at least ceiling(keep_draws / n_chains). If a user-set n_iter or thin keeps fewer (but at least m), a warning is raised and the intervals use the sweeps that were kept; keeping fewer than m is an error.

  param_uncertainty "full" (default), "none" or "both". "none" is an improper plug-in mode for validation only: the covariance matrices are fixed at their posterior means from a full run, and only the missing cells (and the trait means) are drawn. Its result carries mi_workflow = "pigauto_posterior_plugin_diagnostic and with_imputations() and pool_mi() refuse it. "both" is also for validation only: it runs the sampler once and returns the proper results exactly as "full" does, plus the plug-in draws from the same run in mi$posterior_improper.

  seed Integer or NULL. Defaults to seed.

  auto_extend TRUE. If the chains fail the convergence rule (any split R-hat at least 1.05 or any bulk ESS at most 400), every chain continues from where it stopped for another n_iter sweeps and the diagnostics are recomputed on all sweeps after burn-in; this repeats until the rule is met or max_extend extensions have been made. Burn-in is not repeated and the Metropolis step sizes stay as tuned in burn-in, so an extended chain is the same chain as one run longer from the start. After k extensions the kept draws are every thin * (k + 1)-th sweep, spread over the whole run, so their number does not change. A fit that meets the rule first time is not affected. FALSE never extends.

  max_extend 3L. Maximum number of extensions, a whole number from 0 to 10. The default allows at most 4 times n_iter sweeps per chain after burn-in.

`residual_prior` `"sep"` (default) or `"iw"`. The default separation strategy uses half-Cauchy priors for residual standard deviations, scaled by the trait's observed latent-trait standard deviation, and an LKJ(1) prior for the full residual-correlation matrix. Before applying the numerical cutoff, the LKJ(1) prior does not give each pairwise correlation a uniform distribution when there are more than two traits. The cutoff also narrows the prior support; see the posterior-draw details. `"iw"` selects the legacy inverse-Wishart prior used before pigauto 0.11.0.9002; it is retained for reproducing earlier results.

`...` additional arguments forwarded to `fit_pigauto()` via `impute()`. See `fit_pigauto()` for the full list, including the optional `safety_floor` control described below.

`ordinal_levels` named list giving the scientific order of ordinal levels, e.g. `list(status = c("LC", "NT", "VU", "EN", "CR"))`. Each named column is coerced to an ordered factor with exactly those levels before type detection, and the completed datasets return it as an ordered factor. Observed values absent from the declared levels are an error. See `impute()` / `read_traits()`. Default `NULL`.

`x` A stochastic-completion or posterior MI object.

Details

For continuous traits, `draws_method = "posterior"` instead returns posterior predictive draws that the documented software route permits analysts to fit and pool under the conditions below.

Validation boundary: the registered validation found five of eight tested posterior-slope regimes failed and three were indeterminate; none met the support criteria. These results are limited to the tested regimes and do not establish performance for a new analysis. See the registered inferential results before using `with_imputations()` and `pool_mi()`. The separate analysis-aware backend, `multi_impute_analysis()`, covers its own documented narrow regime.

With the default `draws_method = "auto"`, eligible continuous-trait data use posterior draws; other inputs use conformal prediction-diagnostic draws. Check `result$draws_method` before downstream fitting.

The conformal and MC-dropout 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"`: Run the model once. For a continuous latent column standardized to mean zero and unit variance, missing values are sampled as $z_{ij}^{(k)} \sim \mathrm{N}(\hat{z}_{ij},\ (q_{ij}/1.96)^2)$. Here q_{ij} is the available split-conformal score: the global trait score or, with Mondrian calibration, the near/far score for cell i. Other trait types use their type-specific latent representation and decoding described below. Dividing the score by 1.96 is a pragmatic Normal-scale construction, not an inference consequence of a nominal held-out conformal diagnostic. For discrete traits (binary, categorical) it uses Bernoulli / categorical draws from the estimated probability vector. For `zi_count` the gate is Bernoulli from P(nonzero) and the conditional magnitude is drawn on the log1p-z latent using the conformal score (or the magnitude latent SE when scores are unavailable) — not the reported SE of expected count. 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"`: Add independent Gaussian perturbations per latent cell, scaled by selected baseline SD. This returns `M` stochastic draws; when the GNN is enabled, its network

runs use dropout in training mode. These are prediction-diagnostic draws, not joint phylogenetic BM posterior draws or validated downstream-inference imputations. A baseline perturbation can contribute variation when the GNN weight is zero, but this does not guarantee non-zero variation.

`draws_method = "posterior"`: see the "Posterior draws" section.

Nakagawa & Freckleton (2008, 2011) provide ecological missing-data context. Their full-text passages were not verified for a blanket recommendation about this workflow; they do not establish validity of pigauto draws.

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. Missing cells are filled with stochastic draws where available. A cell marked for imputation can remain `NA` if no prediction is available. These datasets are for prediction diagnostics, not downstream inference, except with `draws_method = "posterior"` (unless `posterior_control$param_uncertainty = "none"`), whose datasets are posterior imputations: fit the analysis to them with `with_imputations()` and pool the fits with `pool_mi()` (see `posterior` below and "Posterior draws").

`m` Number of stochastic completion datasets.

`pooled_point` A single data.frame of observed values and point completions for reporting; missing cells can remain `NA` when no prediction is available. It is the point completion returned by the selected route, not generally an average of the `m` decoded datasets. Use `datasets` for an eligible downstream multiple-imputation analysis. For the posterior route, missing-cell values decode the mean latent value over retained posterior predictive draws, not necessarily the `m` returned datasets.

`se` Matrix of per-cell uncertainty summaries. For the conformal and MC-dropout routes, these come from the prediction step; for posterior draws, they are original-scale posterior predictive standard deviations calculated from all retained draws. Their meaning depends on the method and trait type, and they need not describe the spread among the returned datasets. They are descriptive, not Rubin's-rules pooled standard errors, and must not be used for downstream pooling.

`draws_method` The method actually used, including the selected method when `draws_method = "auto"`.

Diagnostic prediction outputs The `conformal_lower`, `conformal_upper`, `probabilities`, and `evaluation` components are returned for the conformal and MC-dropout routes, not the posterior route.

`conformal_lower`, `conformal_upper` For diagnostic routes, lower and upper prediction-bound matrices when conformal scores are available. A bound matrix may contain `NA` for traits without scores; both components are `NULL` when no score object is available. See `predict.pigauto_fit()` for scale and trait-specific interpretation.

`probabilities` For diagnostic routes, a named list of class probabilities and, for a `multi_proportion` trait when only one prediction run is returned, component proportions. See `predict.pigauto_fit()` for the per-type shapes.

`evaluation` For diagnostic routes, validation- and test-set evaluation metrics from the underlying imputation fit. This is `NULL` when there are no test cells or when evaluation fails.

Prediction row order On diagnostic routes, `se`, the bound matrices, and probability rows follow the fitted model's internal row order. `data$input_row_order[k]` gives the original input row placed at internal row `k`; `NA` marks a tree tip without an input row; use `order(data$input_row_order, na.last = NA)` to restore input-row order. `datasets` and `pooled_point` use original input row order. Posterior- route `se` uses original input row order.

`mi_workflow` Method-dependent provenance. Prediction-diagnostic conformal and MC-dropout draws use `"pigauto_diagnostic_mi"`. Eligible posterior draws with parameter uncertainty use `"pigauto_posterior_mi_v1"` and can enter `with_imputations()` and `pool_mi()` under the documented posterior-route conditions. Posterior draws with `param_uncertainty = "none"` use `"pigauto_posterior_plugin_diagnostic"`: covariance matrices are fixed while trait means and missing cells are drawn, and these draws remain ineligible for downstream pooling.

`imputed_mask` Logical matrix aligned to the input rows and trait columns; `TRUE` marks an originally missing target in a row included in the fit, not necessarily a successful fill. A marked cell can remain `NA` if no prediction was available. When `species_col` is supplied, its column is excluded.

`fit` The underlying `pigauto_fit` object, retained for diagnostics and calls to `predict()` on the fitted data. New-species prediction is not implemented.

`data` The `pigauto_data` object.

`tree` The input phylogeny.

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

`posterior` `draws_method = "posterior"` only. A list with `cell_interval` (data.frame: `row` (row of `traits`), `trait`, `lower`, `upper`, `median`; 95% posterior predictive interval on the original scale), `diagnostics` (data.frame: `parameter`, `rhat`, `ess_bulk`, with attributes `"converged"`, `"n_extensions"` (the number of automatic extensions made) and `"sweeps_per_chain"` (the sweeps after burn-in per chain that the diagnostics use)), `params` (list: `Sigma_P` and `Sigma_E` as `K x K x draws` arrays, `lambda` and `mu` as `draws x K` matrices, on the latent scale), `converged`, `control`, `hyper` (priors), `start` (REML lambda and chain-1 start lambda, before clamping to [0.02, 0.98]), `draw_index` (kept sweeps used for the `m` datasets), `wall_s` and `sweeps` (all sweeps, burn-in included, summed over chains). For this method `fit` is `NULL`, `se` holds the posterior predictive SD of each imputed cell, the class is `c("pigauto_posterior_mi", "pigauto_mi", "list")` and `mi_workflow` is `"pigauto_posterior_mi_v1"` (`"pigauto_posterior_plugin_diagnc` with `posterior_control$param_uncertainty = "none"`, which `with_imputations()` and `pool_mi()` refuse).

`posterior_improper` Only with `posterior_control$param_uncertainty = "both"` (validation only). A list with `datasets` (`m` data.frames drawn with the covariance matrices fixed at their posterior means, with the same missing-cell limits described above, from the same chain run as `datasets`), `cell_interval` (same columns as `posterior$cell_interval`), and the fixed `Sigma_P` and `Sigma_E`. Not for downstream inference: it ignores parameter uncertainty.

The print method returns `x` invisibly.

When to use this

With `draws_method = "conformal"` or `"mc_dropout"`, this function is useful for comparing stochastic prediction behavior from one tree; those draws are not for downstream inference. With `draws_method`

= "posterior" (continuous traits), it returns posterior predictive draws that the documented software route permits analysts to fit and pool under the conditions in "Posterior draws". This software eligibility does not establish model compatibility or inferential performance across regimes; the registered validation had five tested slope regimes fail and three remain indeterminate. with_imputations() fits eligible analyses and pool_mi() pools those fits. multi_impute_analysis() is the separate analysis-aware backend for its own documented regime.

multi_impute_trees() provides an experimental posterior-tree sensitivity check. Its outputs are diagnostic and are not accepted by with_imputations() or pool_mi(), so tree-aware downstream inference is not currently supported.

Posterior draws

With draws_method = "posterior", the latent (z-scored, optionally log-transformed) traits follow the multivariate phylogenetic mixed model

$$\begin{aligned}\mathrm{vec}(Y) \sim \mathrm{N}(&\mathrm{vec}(1_n \mu^\top),\\ &\Sigma_P \otimes R + \Sigma_E \otimes I_n),\end{aligned}$$

where Y is the n-species by K-trait matrix, μ is the vector of K trait means, and vec stacks matrix columns. Here 1_n is an n-vector of ones, R is the phylogenetic correlation matrix of the tree, and I_n is the $n \times n$ identity matrix. The model uses full $K \times K$ phylogenetic (Σ_P) and residual (Σ_E) covariance matrices, with a flat prior on μ. The implied Pagel's lambda of trait k is $\Sigma_P[k,k]/(\Sigma_P[k,k] + \Sigma_E[k,k])$.

The phylogenetic covariance uses a parameter-expanded inverse-Wishart prior on its standardized covariance matrix and Normal priors on the expansion parameters. The default residual covariance prior uses a separation strategy: residual standard deviations have half-Cauchy priors scaled by the observed latent-trait standard deviations, and the full residual-correlation matrix has an LKJ(1) prior. Before applying the numerical cutoff, the LKJ(1) prior does not give each pairwise correlation a uniform distribution when there are more than two traits. For numerical stability, the sampler gives zero prior density if the smallest eigenvalue of the residual covariance matrix, standardized by observed latent-trait standard deviations, is below 0.000001. In plain language, on this standardized scale, variance along every unit-length direction must be at least 0.000001. This cutoff narrows the prior support. The legacy posterior_control$residual_prior = "iw" option instead uses an inverse-Wishart prior on the residual covariance. Prior settings are returned in mi$posterior$hyper.

The sampler draws the phylogenetic effects, the trait means and the missing cells as one block using the implementation's sparse phylogenetic node-precision Cholesky factor, and updates the covariance matrices by parameter-expanded Gibbs steps plus Metropolis moves with the phylogenetic effects integrated out. Mixing is monitored with rank-normalised split R-hat and bulk effective sample size (Vehtari et al. 2021); mi$posterior$diagnostics reports them. Vehtari et al. recommend R-hat below 1.01. Pigauto's R-hat cutoff of 1.05 is less strict than the paper's recommendation; its bulk-ESS cutoff is 400. When any R-hat is at least 1.05 or any ESS is at most 400, the chains are extended automatically (posterior_control$auto_extend and max_extend), and a warning is raised if the rule still fails after the last extension. These diagnostics cover the monitored elements of Sigma_P, Sigma_E and lambda; they do not establish adequate precision for every posterior quantile or validate the sampler.

Before fitting a downstream model, check attr(mi$posterior$diagnostics, "converged"). If it is not TRUE, inspect the per-parameter diagnostics and sampling controls, then refit and check

again. Do not interpret or pool the draws while the flag is false or unavailable. Passing this package-specific convergence rule does not establish calibrated downstream inference.

Evidence scope. The earlier Rubin-study campaign used pigauto b565cad and subsequent separation-prior refits, 200 datasets per cell, n = 100, 300 or 1000, lambda = 0.3, 0.7 or 1, trait correlation 0 or 0.5, and 30% MCAR missingness. Its downstream targets were PGLS slopes and phylogenetic correlations; the coverage claims above belong to that study and distinguish the legacy IW from the separation prior. The separate posterior acceptance sweep used 40 regimes and its own gate rules. The later registered validation froze a different source and used the default separation prior. Its eight posterior slope regimes used Brownian GLS, n = 300 or 1000, lambda = 0.5 or 1, trait correlation 0.7, 30% missingness under MCAR or phylogenetic MAR, and 500 replicates per regime. Five failed and three were indeterminate; empirical 95% coverage ranged from 86.2% to 95.2%. No inferential scenario met all prespecified support criteria. These campaigns should be compared within their stated models, priors, missingness and acceptance rules. Bayesian properness under the stated model does not by itself establish that an applied analysis is calibrated.

The `m` datasets are posterior predictive draws spaced evenly across the kept sweeps of all chains. Per-cell 95% intervals in `mi$posterior$cell_interval` come from all kept sweeps (`keep_draws` of them by default), never from the `m` datasets.

These draws come from a linear-Gaussian model of the imputed traits on the scale where they were imputed (the log scale for traits log-transformed by `log_transform`). The documented software route permits downstream fitting only under the stated eligibility conditions: analyses must be linear in the imputed traits on that scale and use variables among the imputed traits. This fitting eligibility does not establish model compatibility or inferential performance across analyses or tested regimes. In the registered validation, five posterior slope regimes failed and three were indeterminate; see the registered inferential results. Each imputed trait must also have at least two observed values among the rows being fitted; otherwise posterior mode stops and names the affected traits. This minimum is a software requirement, not evidence that two observations are enough for reliable estimation. Not covered: covariates from outside the imputed traits, because the imputation model does not contain them; nonlinear terms (such as squares) or interactions among imputed traits; and analysing a log-transformed trait on its raw scale. The GNN is not used, and `gnn`, `epochs`, `missing_frac`, `lambda_mode` and other fitting arguments are ignored (a message lists any that were supplied). Non-continuous traits, multiple observations per species (or any `species_col`), `covariates`, and input with no missing cells in the rows of `traits` (tree tips absent from `traits` do not count) are errors.

Optional safety floor

When `fit_pigauto()` is called with `safety_floor = TRUE` (opt-in; the default is `FALSE`), the 3-way blend `r_BM * BM + r_GNN * GNN + r_MEAN * MEAN` propagates through prediction-diagnostic draws through `predict.pigauto_fit()`. The posterior sampler uses its own model and does not use this blend. 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 independent baseline perturbations and, when enabled, 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.

Printing

Prints diagnostic or posterior/convergence summaries. Cell counts refer to targeted cells; a targeted cell can remain missing if a prediction is unavailable. Displaying an object does not authorize

downstream pooling. The ... argument is currently unused.

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.

Gelman A (2006). "Prior distributions for variance parameters in hierarchical models." *Bayesian Analysis* 1(3): 515-534.

Hadfield JD (2010). "MCMC methods for multi-response generalized linear mixed models: the MCMCglmm R package." *Journal of Statistical Software* 33(2): 1-22.

Hadfield JD, Nakagawa S (2010). "General quantitative genetic methods for comparative biology: phylogenies, taxonomies and multi-trait models for continuous and categorical characters." *Journal of Evolutionary Biology* 23(3): 494-508.

Vehtari A, Gelman A, Simpson D, Carpenter B, Buerkner P-C (2021). "Rank-normalization, folding, and localization: an improved R-hat for assessing convergence of MCMC." *Bayesian Analysis* 16(2): 667-718.

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.

The references provide methodological context for priors, phylogenetic models and MCMC diagnostics. They do not establish pigauto's exact prior specification, sampler correctness or package-specific diagnostic thresholds. The "Posterior draws" section describes the implementation and retained campaign evidence separately.

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$Mass[seq_len(3L)] <- NA_real_
mi <- multi_impute(df, tree, m = 2L, draws_method = "conformal",
                   epochs = 5L, verbose = FALSE)
print(mi)
lapply(mi$datasets, head)
```

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

Description

Generate multiple imputations conditional on a specified downstream regression model. All routes rely on missing at random (MAR) and a correctly specified conditional imputation model. The `glm` and `lmer` routes use SMCFCS and jomo, respectively, so their sampler convergence must also be checked. The `lm` route uses a separate Bayesian Normal-regression sampler. 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()
)

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

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 double-precision continuous covariate to impute. Integer storage is rejected: for a scientifically continuous whole-number measurement, convert it first with `data$x <- as.numeric(data$x)`. It must occur as a main-effect predictor in `formula` and be the only column 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).
x	An analysis-aware MI object.
...	Ignored. Included for compatibility with the base print() generic.

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. For the "glm" and "lmer" routes, any engine warning aborts the call instead of returning an object. The "lm" route does not apply this rule. mi_workflow is a stable analysis-aware provenance marker propagated by with_imputations() and checked by pool_mi().

The print method returns x invisibly.

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.

Printing

Prints the dataset count, engine, formula and metadata. This display does not validate an analysis. The ... argument is currently unused.

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` datasets per tree, either as point predictions or stochastic completions. This returns `T * m_per_tree` 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,
  draws_method = c("mc_dropout", "conformal"),
  gnn = FALSE,
  ...,
  ordinal_levels = NULL
)

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

Arguments

`traits`	data.frame. Same format as `multi_impute()` and `impute()`.
`trees`	list of `phylo` objects (class `multiPhylo` or plain list). For the default shared-reference workflow, supply trees with a common tip set, including the reference tree if supplied. With `share_gnn = FALSE`, tip sets may differ. Input species

absent from the modeled tree, or from the reference tree in the shared workflow, remain in returned datasets but are not imputed there. Modeled tree tips without trait rows are represented internally as all-missing rows. Posterior samples from BirdTree.org (Jetz et al. 2012) are an example; the bundled trees300 dataset provides 50 posterior trees for avonet300.

`m_per_tree` integer. Number of completed datasets returned per tree (default `1`). Total datasets = `length(trees) * m_per_tree`. With `draws_method = "mc_dropout"` and the default of one, the prediction is deterministic conditional on each fitted model; this setting does not estimate within-tree variation.

`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. Log-transform every continuous trait whose observed values are all positive (default `TRUE`), then back-transform the imputations. Set it to `FALSE` when traits are already on a log scale (for example log body mass): otherwise they are logged a second time and imputed on a log(log) scale, which can bias analyses done on the log scale. With a mix of raw and already-logged traits, log the raw ones yourself and set `FALSE`. In a real-data check (PanTHERIA, 4,027 mammal species, four already-logged traits, posterior draws), the default shifted Rubin-pooled regression slopes by up to 3.9 standard errors of the complete-data slope, in a direction that depended on which values were missing; with `FALSE` the largest shift was 1.8.

`missing_frac` numeric. Fraction held out for validation/test during training (default `0.25`).

`covariates` data.frame or matrix of environmental covariates (fully observed), passed through to `impute()`. Numeric and integer columns are standardized; factor and ordered columns are one-hot encoded. Follow the row-alignment rules in `impute()` and provide covariates for modeled species or observations, including tree tips without trait rows. 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. How it is used depends on `share_gnn` and `draws_method`; see the "Reproducibility" section. The default `NULL` uses the current RNG stream.

`share_gnn` logical. With `gnn = TRUE`, `TRUE` (the default) trains one GNN on a reference tree and reuses it while recomputing the baseline for each posterior tree. Set `FALSE` to refit the GNN on each tree. With the default `gnn = FALSE`, no GNN is trained in either mode. See the "Share-GNN" section below.

`reference_tree` optional `phylo` used for the reference fit 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.

draws_method	character. How missing cells are completed. One of "mc_dropout" (default) or "conformal". The methods differ in whether they add stochastic variation; see the section below. A single mc_dropout completion is a deterministic prediction, while "conformal" runs its sampling step even when m_per_tree = 1.
gnn	logical. Passed through to impute() / fit_pigauto() for every tree. When TRUE, the usual GNN correction is trained (once, on the reference tree, when share_gnn = TRUE). When FALSE, no GNN is used anywhere in the pipeline: baseline-only fits throughout (the default since 0.11.0.9001; see impute()). With share_gnn = TRUE, each tree's baseline is fit without held-out splits (splits = NULL), matching the production baseline_full semantics fit_pigauto() uses.
...	additional arguments forwarded to fit_pigauto() via impute().
ordinal_levels	named list giving the scientific order of ordinal levels, coerced to ordered factors before fitting. See impute() / read_traits(). Default NULL.
x	A posterior-tree prediction-sensitivity object.

Details

This is an experimental prediction-sensitivity check. Use it to see whether point predictions change across the trees you supply. It does not provide inferential uncertainty for downstream model estimates. See vignette("tree-uncertainty") for a step-by-step example.

Each dataset reflects the tree assigned to it. Comparing datasets across trees shows prediction sensitivity to the supplied tree sample. When multiple draws are requested per tree, they can also vary within a tree, depending on the selected draw method. These results are diagnostic, not analysis-ready imputations. The downstream fitting and pooling helpers reject them, so do not use them for inference. This workflow does not combine tree-to-tree variation with imputation variation into an inferential estimate.

For each tree the function runs preprocessing, baseline fitting and prediction. A GNN is also fitted only when gnn = TRUE; with share_gnn = TRUE, that GNN is trained once on a reference 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. The following historical planning estimates apply to gnn = TRUE, with one shared GNN fit plus T baseline passes when share_gnn = TRUE. They are not timings for the current GNN-off default; estimate the intended tree and sample setting before a campaign.

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` data.frames. Observed cells are preserved. Missing cells are filled with tree-conditional predictions or draws where available. A cell marked for imputation can remain `NA` if no prediction is available. These datasets are for sensitivity diagnostics.

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

`n_trees` Number of posterior trees used.

`m_per_tree` Number of prediction or completion datasets 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 numeric trait columns across all `T * m_per_tree` datasets; an `NA` in a numeric trait cell propagates to its average. Factor and ordered-factor columns are taken from the first dataset. Compare the `datasets` directly to describe nonnumeric prediction sensitivity. For reporting, not inference.

`se` On the shared-GNN path, the arithmetic average of the per-tree prediction uncertainty summaries (see `predict.pigauto_fit()`). Descriptive reporting only, not a pooled inferential SE. `NULL` when unavailable, including the separate-fit path.

`imputed_mask` Logical matrix aligned to the input rows and trait columns; `TRUE` marks an originally missing target in a row included in the fit, not necessarily a successful fill. A marked cell can remain `NA` if no prediction was available. When `species_col` is supplied, its column is excluded.

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

`draws_method` Character; `"mc_dropout"` or `"conformal"`, echoing the argument this call used.

`mi_workflow` `"pigauto_tree_sensitivity_diagnostic"`, recording that these are prediction-sensitivity completions and cannot be passed to `with_imputations()` or `pool_mi()`.

`fit` Single `pigauto_fit` fitted 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 the shared fit when `share_gnn = TRUE`; `NULL` otherwise.

`trees` The input posterior trees.

`species_col` Passed-through species column name.

`lambda_per_trait_by_tree` Returned only when `share_gnn = TRUE`. A list indexed in the same order as `trees`; element `i` is the named `lambda_per_trait` vector returned by `fit_baseline()` for `trees[[i]]`. Values can be fixed rather than estimated. Categorical one-vs-rest lambda estimates are not included in this vector, so a categorical column's default value of 1 is not an estimated class-specific lambda.

The print method returns `x` invisibly.

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 to examine prediction sensitivity. Its results cannot be pooled for downstream inference.

Reproducibility

Pigauto routes the base seed separately to tree fits and sampling steps, so the offset depends on the fitting mode. Here, t is a tree's position in the supplied list. With `share_gnn = FALSE`, the separate fit for tree t receives `seed + t - 1`. With `share_gnn = TRUE`, the reference-tree fit receives `seed`, and each tree's baseline refit receives `seed + t`. Stochastic predictions reuse the stored seed from the reference fit; deterministic single-pass predictions do not set a new seed. If `draws_method = "conformal"`, the tree-specific completion sampling step receives `seed + t - 1`; each completion within a tree also receives its own seed offset. With `seed = NULL`, these steps use the current random-number stream.

Share-GNN (tree-sharing) mode

When `gnn = TRUE, share_gnn = TRUE`, neural weights, the reference graph, spectral features, and calibrated blend weights are reused from the reference fit (MCC by default). The baseline is recomputed for each posterior tree and passed into prediction again. Because baseline values also enter the neural inputs, shared weights do not imply a constant neural prediction across trees. No direction or magnitude of tree-variance bias has been established for this approximation.

Set `share_gnn = FALSE` with `gnn = TRUE` to fit a separate GNN for each tree. With the default `gnn = FALSE`, there is no GNN channel.

How the completion methods vary (and their limitations)

With `draws_method = "mc_dropout"` (the default), one completion per tree is a deterministic prediction conditional on that tree's fitted model. When `m_per_tree > 1`, pigauto adds independent Gaussian perturbations to latent cells, scaled by their baseline standard deviations. If `gnn = TRUE`, dropout can add further variation. These perturbations are not joint Brownian-motion posterior draws. Variation can be zero when a source has no uncertainty or its contribution is weighted out; decoding can also map different latent values to the same category.

With `draws_method = "conformal"`, missing-value sampling occurs even when `m_per_tree = 1`. For continuous, count, ordinal and proportion traits, it uses conformal widths where available and reported uncertainty. When no finite conformal score is available for these traits, the route uses the reported `pred$se` as the fallback scale input. For log-transformed continuous traits, log1p-transformed counts, and logit-transformed proportions, `pred$se` is reported on the original scale, but the draw is made on the transformed scale. Pigauto does not fully convert `pred$se` between these scales, so diagnostic draw spread may be too wide or too narrow. Fitted point predictions are unchanged. These tree-based conformal completions use global trait scores; they do not carry per-cell near/far Mondrian scores into the draws, even when prediction intervals use Mondrian calibration. Binary and categorical traits are sampled from predicted probabilities; zero-inflated counts combine a Bernoulli gate with a magnitude draw; compositions use latent standard deviations. A fixed seed makes the draws reproducible. Neither method validates downstream inference.

When `share_gnn = TRUE`, conformal scores and calibrated blend weights come from the reference fit and are reused across trees. With `share_gnn = FALSE`, each tree uses scores and weights from its own fit.

Safety floor with a shared fit

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 reference fit and are not recalibrated for each topology. Each posterior tree recomputes the baseline. The three weights stay fixed, while the GNN delta can change with its per-tree baseline inputs. This preserves prediction sensitivity to the per-tree baseline without re-calibrating the safety floor; it does not establish downstream inferential validity.

Printing

Prints tree, row, trait, and targeted-cell counts and the diagnostic-only scope. A targeted cell can remain missing when the prediction is unavailable. The draw-method label does not imply dropout ran with GNN off or with one completion per tree. The `...` argument is currently unused.

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 or, with `draws_method = "posterior"`, single-tree inference on continuous traits; `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 imputation report from a pigauto fit*

Description

Produces an HTML file with charts. Baseline-only figures are embedded as SVG and work offline; GNN charts load Chart.js from its public CDN. The report compares the fitted baseline with the calibrated blend, which may include the neural component and other baseline contributions. This is not a comparison isolating the neural component. With `gnn = FALSE`, no neural network is fitted. Per-trait metrics, gate values, and held-out diagnostic conformal coverage are included when available.

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`, or reconstructed from a fit that retains its scaled observations. When supplied it must match the fit (species and traits).
`splits`	Optional splits object from `make_missing_splits`. Extracted automatically from the result or fit when available. When supplied it must match the fit.
`output_path`	Character of length 1. File path for the HTML report (default `"pigauto_report.html"` in the working directory).
`title`	Character of length 1. Report title (HTML-escaped on write).
`open`	Logical of length 1. 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 when a GNN was trained): 2x2 panel of training loss components (reconstruction, shrinkage, gate regularisation) and validation loss over epochs. A fit with `gnn = FALSE` (the default) has no training history, so `plot(fit)` then shows `"gates"` instead. `"gates"`: bar plot of calibrated gate values per trait, coloured by trait type (green = continuous, blue = count, orange = ordinal, red = binary, purple = categorical). When `gnn = FALSE`, gate values are zero, so no bars may be visible; that is expected for a baseline-only fit. `"conformal"`: bar plot of conformal prediction scores per trait with a dashed median line and its value labelled above the bars.
`...`	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)
# The package default disables the optional GNN, so this example works
# without torch or its Lantern runtime.
fit <- fit_pigauto(data, tree, splits = splits, epochs = 5L,
                   eval_every = 1L, verbose = FALSE, gnn = FALSE)
plot(fit)
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. Rows must match the predictions in `x` one-for-one and in the same order; rows are paired by position, not matched by row names or species identifiers. Trait columns are selected by name. Values are overlaid on scatter and interval plots. For probability plots, class labels are used to group predictions; rows with missing or unrecognized labels are omitted from these groups.
`splits`	Optional list (output of `make_missing_splits`). Currently unused by this plotting method. All finite pairs in `data` and `x$imputed` are plotted, so the scatter is not a held-out performance estimate. To display held-out cells, obtain `predict(fit, evaluation = TRUE)`, and supply a truth frame with every cell outside the desired validation/test subset set to NA.
`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, the probability of the second class level is shown in boxplots grouped by observed class when usable labels are present; otherwise a histogram shows the probabilities. For categorical traits, usable labels give stacked bars of mean predicted probabilities by observed class; without usable labels, stacked bars show up to 100 observations, ordered by their most probable class. These plots are descriptive: grouping by observed class does not itself assess calibration. Unless the predictions are for cells held out from fitting and `data` supplies their truth, they are not held-out performance assessments. To inspect held-out predictions, use `predict(fit, evaluation = TRUE)` and provide a truth frame with all other cells set to `NA`. Select only traits with at least one usable held-out class label; otherwise the fallback plot uses predictions without restricting them to held-out cells.
`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$Mass[seq_len(3L)] <- NA_real_
result <- impute(traits, tree, epochs = 5L, verbose = FALSE)
pred <- result$prediction
plot(pred)
plot(pred, type = "scatter", data = traits) # descriptive all-finite pairs
plot(pred, type = "intervals", trait = "Mass")
# These two continuous traits each occupy one latent column.
evaluation_pred <- predict(result$fit, evaluation = TRUE)
heldout_truth <- as.matrix(traits)
heldout_truth[] <- NA_real_
idx <- result$fit$splits$test_idx
heldout_truth[idx] <- as.matrix(traits)[idx]
plot(evaluation_pred, type = "scatter", data = heldout_truth)
```

`plot.pigauto_result` *Plot diagnostics for a pigauto result*

Description

S3 `plot` method for `pigauto_result` objects. Delegates to `plot.pigauto_fit` on `x$fit`.

Usage

```
## S3 method for class 'pigauto_result'
plot(x, ...)
```

Arguments

`x`	An object of class `"pigauto_result"`.
`...`	Passed to `plot.pigauto_fit`.

Value

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

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("baseline", "pigauto"),
  ...
)
```

Arguments

results	A data.frame with columns `trait`, `type`, `metric`, `method`, and `value`. Typically the output of `compare_methods`, `evaluate`, 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 pigauto method name. Default `c("baseline", "pigauto")`, matching the labels returned by `compare_methods` and `evaluate`.
...	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("baseline", "pigauto"),
  value = c(1, 0.8)
)
```

```
plot_comparison(results)
plot_comparison(results, metric = "rmse")
```

`plot_history_gg` *Plot training history (ggplot2, deprecated)*

Description

Deprecated. If the fit has GNN training history, use the base-R S3 method with `type = "history"`. The default `gnn = FALSE` fit has no training history; for that fit, use `plot(fit)` to show its gates or request `type = "gates"` explicitly.

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_missingness` *Plot observed missingness patterns*

Description

Display missingness by trait, observation and phylogenetic tip order, or display the fraction of observations missing both traits. Patterns can suggest structure to investigate; they cannot establish whether data are missing completely at random. Trait groups expanded into multiple latent columns count as missing when all their columns are missing.

Usage

```
plot_missingness(
  x,
  type = c("summary", "matrix", "phylo", "co_missing"),
  tree = NULL
)
```

Arguments

x	A preprocessed pigauto_data object.
type	One of "summary", "matrix", "phylo", or "co_missing".
tree	A phylo tree, required for type = "phylo". Rows are ordered by its tip order; repeated observations appear together.

Value

Invisibly, a logical observation-by-trait missingness matrix.

Examples

```
data(avonet300, tree300)
tree <- ape::keep.tip(tree300, tree300$tip.label[seq_len(30)])
d <- avonet300[match(tree$tip.label, avonet300$Species_Key),
               c("Mass", "Wing.Length"), drop = FALSE]
rownames(d) <- tree$tip.label
d$Mass[1:3] <- NA
x <- preprocess_traits(d, tree)
plot_missingness(x, "summary")
plot_missingness(x, "phylo", tree)
```

---

plot_uncertainty *Plot predictions and uncertainty displays for imputed trait values*

Description

For continuous, count, and ordinal traits, plots conformal prediction intervals when present; otherwise it shows an approximate prediction +/- 1.96 * SE ribbon, sorted by predicted value. If truth is supplied, observed values are overlaid as points. For binary traits, the centre line is the predicted probability when available. Binary predictions have no conformal bounds here. If bounds are absent, this function still draws a ribbon using min(p, 1-p), a probability-of-error score, as if it were an SE. This binary ribbon is only a visual score display: it is not a calibrated interval and has no stated coverage. Interpret the centre line as the probability; do not interpret the ribbon as an uncertainty interval. See predict.pigauto_fit for type-specific uncertainty meanings.

Usage

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

Arguments

pred_result	list with imputed and se components (output of predict.pigauto_fit), or a pigauto_result from impute (its $prediction is used).
truth	matrix or data.frame of true values (same scale as pred_result$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. To pool an analysis, first use with_imputations() to fit one eligible downstream model to each completed dataset, then pass those fits here. Start with the analysis-aware multi_impute_analysis() workflow or, for eligible continuous-trait data, multi_impute() with draws_method = "posterior". Check those help topics for input restrictions and inferential limits. This pooling function 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
)
```

```
## S3 method for class 'pigauto_pooled'
print(x, digits = 4, ...)
```

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. Bayesian fits (brms, MCMCglmm) are rejected; see Details. Bare user-supplied fit lists are accepted with an explicit warning because their imputation provenance cannot be verified. pigauto workflow classes and markers are caller-controlled metadata; they do not authenticate the fits or their history. Failed try-error and pigauto_mi_error elements are removed with a warning; at least two retained fits are required. Inspect these failures before interpreting results: dropping analyses can change the imputation sample.
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. For recognized drmTMB and gllvmTMB_multi fits, pigauto still requires convergence code zero and a positive-definite Hessian, even when a custom extractor is supplied.
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. Called once on the first retained fit after failures are removed; it must return one finite positive number assumed appropriate for the common analysis. Supplying it selects the Barnard & Rubin (1999) correction for finite complete-data degrees of freedom. 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.
x	A pooled coefficient table.
digits	Number of significant digits used for the displayed table.
...	Ignored. Included for compatibility with the base print() generic.

Details

Let $\hat{\theta}_i$ be the coefficient vector from fit i and $U_i = \mathrm{vcov}(\mathrm{fit}_i)$, for $i = 1, \ldots, M$. The implementation uses the following Rubin-style pooling equations:

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

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

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

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

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

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

Here `nu` is the pooled degrees of freedom returned in `df`: the classical value when `df_fun = NULL`, or the finite-complete-data adjustment below when `df_fun` is supplied. When `df_fun` returns finite complete-data df `nu_com`, pigauto applies a finite-df adjustment motivated by Barnard and Rubin (1999). The equations below specify pigauto's implementation. It uses $\lambda = (1 + 1/M)B/T$, the proportion of total variance from between-imputation uncertainty, which differs from the finite-df `fmi` above. It 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. For a coefficient with `W = 0` and `B > 0`, all total variance is between imputations: the pooled coefficient estimates differ across imputed datasets, while each fit reports zero variance. This implementation returns `df = M - 1` and `fmi = 1`, including when `df_fun` returns a finite positive complete-data df. The observed-data df term is zero in this limit, so the harmonic adjustment is skipped. When both variance components are zero (`W = B = 0`), it returns `df = Inf` and `fmi = 0`.

Only fixed-effect coefficients are pooled. 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. `drmTMB` and `gllvmTMB` need to be installed and loadable only when their automatic adapters are used. For either recognized backend class, pooling also requires the fit to report scalar convergence code `0` and `pdHess = TRUE`; missing or invalid status is rejected before coefficient extraction, including with custom extractors.

Supported classes. `lm`, `glm`, `gls`, `lme`, `merMod` (lme4), `glmmTMB` (conditional component), `drmTMB`, and `gllvmTMB_multi` are handled automatically. Other classes are pooled through their `coef()` / `vcov()` methods, or through `coef_fun` / `vcov_fun` / `tidy_fun`.

Bayesian fits (brms, MCMCglmm) are rejected because Rubin's rules are not the right tool for posterior samples. Posterior-draw combination is a separate procedure, not a blanket recommendation: use it only with an eligible imputation method and a compatible downstream model, check

convergence for every imputation-specific fit, and give each imputed dataset equal representation in the combined draws. For brms, use `brms::brm_multiple(formula, data = mi$datasets)`, or `brms::combine_models()` on a list of existing fits. For MCMCglmm, concatenate equally represented `Sol` samples only after checking each fit. Combined draws do not establish calibrated inference for pigauto. See the multiple-imputation guide for workflow limits and the current brms diagnostics guidance. No Bayesian downstream workflow is supported by the analysis-aware backend.

The references below provide general multiple-imputation and ecological context. The exact book passages for these equations and the ecological review passages have not been checked against the primary full texts; these citations do not validate pigauto's imputation workflows.

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.

The `m` attribute records the retained fit count used in all pooling calculations; it can be smaller than the supplied list.

The print method returns `x` invisibly.

Printing

Prints coefficients, retained-fit count, confidence level and an FMI-based imputation-count suggestion. That suggestion is a stability heuristic, not a validity check. The `...` argument is currently unused.

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_
# Workflow example only: m = 2L is not an analysis recommendation.
# Choose enough imputations to make Monte Carlo error small relative to
# the standard errors you report.
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

Returns imputed trait values on the original scale for data used to fit the model. Supports continuous, binary, categorical, ordinal, count, proportion, zero-inflated count, and multi-proportion traits.

Multiple draws are diagnostic only. With `n_imputations > 1`, returned draws are not supported for downstream inference. For multi-proportion traits, pooled point values and simplex probabilities are omitted; use a single-run prediction when you need the simplex output.

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,
  evaluation = FALSE,
  ...
```

```
)

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

Arguments

object	object of class "pigauto_fit".
newdata	Must be NULL: predictions use the fitted data. Any non-NULL value is rejected; new-species prediction is not implemented.
return_se	logical. Return type-dependent uncertainty summaries? Default TRUE; see the se return description below.
n_imputations	integer. Number of stochastic prediction-diagnostic draws (default 1L). With multiple draws, independent Gaussian perturbations are added per latent cell using the selected baseline SD; cells with zero selected baseline SD receive no perturbation. GNN dropout also varies across runs when the GNN is enabled. These are not joint phylogenetic BM posterior draws or validated analysis-aware imputations.
baseline_override	optional list(mu, se) with the same shape as object$baseline. When supplied, predictions use this baseline instead of the one saved in the fit. Used internally by multi_impute_trees() to reuse a trained GNN across posterior trees. Most users can ignore this. Default NULL (use the fit's own baseline).
pool_method	character. One of "median" (default), "mean", or "mode". Controls pooling when n_imputations > 1; see "Pooling decoded draws" below. Some types use the same rule for every setting.
clamp_outliers	logical. When TRUE, cap back-transformed predictions for log-transformed continuous, count, and zi_count magnitude traits at tm$obs_max * clamp_factor, where tm$obs_max is the observed maximum on the original scale, recorded during preprocessing. This optional cap limits extrapolation after exp() or expm1(); it is not an uncertainty bound. Default FALSE leaves these predictions uncapped.
clamp_factor	numeric scalar (>= 1). User-selected multiplier of the observed maximum for the upper cap when clamping is enabled. Default 5. Ignored when clamp_outliers = FALSE.
match_observed	character, one of c("none", "pmm"). When "pmm", uses predictive Mean Matching (Little 1988; see References) 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 mc_dropout prediction-diagnostic 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.

Donor matching can reduce extrapolation but also replaces accurate predictions with donor values, so it is not a general RMSE improvement.

Discrete-class types (binary / categorical / ordinal / multi_proportion) and unlog continuous: no-op. Default `"none"` leaves predictions unmatched.

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

`evaluation` Logical. Use `TRUE` to predict validation and test cells using the fit's held-out baseline and masked input context. Requires stored splits and `newdata = NULL`. Default `FALSE` uses all observed data for production imputation. Use `evaluation = TRUE` before passing predictions to `evaluate_imputation`.

`...` Internal compatibility arguments. Unrecognised arguments are an error, including misspelled argument names.

`x` A pigauto prediction object.

Details

With `n_imputations > 1`, predictions include stochastic diagnostic draws, not joint phylogenetic BM posterior draws. On each run, independent Gaussian perturbations are added per latent cell to the selected baseline component, scaled by that component's standard deviation (the second parameter in `N(mu, sd)` is an SD, not a variance). A route that supplies zero uncertainty placeholders adds no Gaussian perturbation for those cells; a zero placeholder does not indicate certainty. When the GNN is enabled, dropout also varies across runs. Because predictions use the phylogenetic graph, predictions across species need not be independent. The calibrated latent prediction combines the selected baseline, GNN prediction and optional mean baseline using their calibrated weights; an applicable direct covariate contribution is then added. In stochastic runs the selected baseline term uses its perturbed value. A zero GNN weight removes only the neural contribution. The output equals the baseline draw only when the baseline weight is one and every other contribution is absent. Point predictions and these stochastic diagnostic draws are distinct outputs. Point estimates follow the type-specific "Pooling decoded draws" table below. 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.

`gnn = FALSE` **fits (traditional-stats mode):** when the fit was produced with `gnn = FALSE`, no GNN was ever trained, so this function makes no `torch::` calls at all – prediction is the calibrated baseline blend `r_cal_bm * baseline + r_cal_mean * mean_baseline_per_col` (`r_cal_gnn` is always zero) computed on plain R matrices. Two baselines are available: in production mode (the default, `evaluation = FALSE`) predictions use `baseline_full` – fit with `splits = NULL`, so every observed cell can inform every other cell; in evaluation mode (`evaluation = TRUE`) predictions use the held-out `baseline` instead, so scored cells never see their own value. `n_imputations > 1` adds independent Gaussian perturbations per latent cell, scaled by the selected baseline standard deviation. Cells with zero selected baseline SD receive no Gaussian perturbation. These are stochastic prediction diagnostics, not joint phylogenetic BM posterior draws. The `gnn = FALSE` path has no GNN-dropout contribution.

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

Conformal intervals: `conformal_lower` / `conformal_upper` use whatever `conformal_method` the fit was trained with (`fit_pigauto`). For `"mondrian"`, the half-width is not constant across rows: each target cell's phylogenetic locality (mean cophenetic distance to its 5 nearest training-observed species for that trait) picks the near- or far-stratum score stored on the fit. These are stratum-specific intervals; the far-stratum score is not necessarily larger than the near-stratum score. See `fit_pigauto`'s `conformal_method` argument for details.

`zi_count` **conformal intervals are conditional on non-zero.** `conformal_lower[, nm]` / `conformal_upper[, nm]` for a `zi_count` trait back-transform the magnitude latent (log1p-z of the count *given it is non-zero*) plus/minus the conformal half-width that was calibrated on that same magnitude column (`fit_pigauto`'s `compute_conformal_scores()` scores zi_count on `latent_cols[2]`, never on the decoded `E[X]`). They are **not** an interval for the expected value `E[X] = P(nonzero) * E[count | nonzero]` – constructing a naive ±quantile interval directly on `E[X]` would conflate gate and magnitude uncertainty into one number with no valid coverage interpretation. The gate probability `P(nonzero)` is available separately in `pred$probabilities[[nm]]`; combine it with the conditional interval yourself if you need an `E[X]`-scale bound.

Value

A list of class `"pigauto_pred"` with:

imputed data.frame of model predictions for every fitted row (observed and missing cells), on the original scale with proper R types (numeric, integer, factor, ordered). Observed values are not retained in this component. Use `impute()` and its `$completed` component for a data.frame that preserves observed values and fills missing cells when predictions are available.

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. See the "What `$se` means per trait type" section below. Single-run and multi-draw constructions differ. These are diagnostic summaries, not calibrated total predictive SDs or valid inputs to downstream pooling. `NULL` if `return_se = FALSE`.

se_latent Numeric matrix on the latent scale, with one column per latent trait column. Returned when `return_se = TRUE` for current trait-map fits; otherwise `NULL`. Binary and categorical entries are `NA`; for `zi_count`, only the magnitude column has a summary.

probabilities Named list. Binary traits: numeric probability vector; categorical traits: n x K class-probability matrix; zi_count traits: probability of nonzero. A single-run multi_proportion prediction includes its n x K simplex matrix. Compositional entries are absent from pooled multi-draw output.

conformal_lower, conformal_upper Matrices of lower and upper conformal limits when the fitted model has conformal scores. Values are on the original trait scale except for ordinal traits, whose bounds are rounded zero-based indices. The lower limit is floored at 0 and the upper limit capped at K - 1; these one-sided limits do not guarantee both endpoints are in range or

that the interval is non-empty. Check bounds before mapping an index to a label using the corresponding ordered factor levels (index + 1 in R). Entries can be NA for unsupported traits or unavailable scores; both are NULL when scores are absent.

conformal_scores Named numeric vector of the scores stored in the fitted model, or NULL when the fit has no scores.

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.

evaluation_safe_idx Latent indices excluded from fitting and prediction input, used to check held-out scoring.

The print method returns x invisibly.

Pooling decoded draws

These rules apply to n_imputations > 1, on the decoded scale.

Type	median	mean	mode
continuous, unlogged	mean	mean	mean
continuous, logged	median	median	median
count / proportion / zi_count	median	mean	median
ordinal	rounded mean class index	rounded mean class index	modal class
binary	mean probability, then threshold 0.5	same	same
categorical	mean class-probability vector, then argmax	same	same
multi_proportion	pooled output unavailable	unavailable	unavailable

Counts and zi_count values are rounded and clipped to nonnegative integers; the zi_count nonzero probability is always averaged separately. Single-run compositional output is a simplex vector; its pooled MC-dropout counterpart is not populated, although individual decoded compositions are returned in imputed_datasets. Do not interpret this omission as simplex averaging.

What $se means per trait type

For fits with a trait_map, pred$se is an n x n_original_traits matrix of type-dependent diagnostic summaries. Its construction also depends on n_imputations. It is not a uniformly Gaussian standard error, a calibrated total predictive SD, or an input to Rubin pooling.

continuous/count/ordinal/proportion Single run: selected-baseline latent uncertainty scaled/back-transformed to the original scale. Multiple draws: square root of transformed baseline variance plus between-decoded-draw variance; logged continuous, count, and proportion transformations use the first latent run as their anchor.

binary min(p, 1-p), using the mean probability across draws when n_imputations > 1.

categorical 1 - max(p_k), using the mean class-probability vector across draws when n_imputations > 1.

zi_count Single run: a delta-method product summary combining gate and magnitude terms. Multiple draws: between-draw SD of decoded expected counts, with no additional baseline-variance term.

multi_proportion NA for either setting.

For a BM Gaussian baseline, the single-run continuous-family source is a model-dependent BM/joint conditional SD. It is not total final blended-prediction uncertainty: it describes uncertainty under the selected baseline, not prediction error in the final baseline–GNN blend. Other route summaries do not inherit that interpretation automatically. Discrete scores measure distance from a certain class prediction, not a Gaussian SE. For compositions, inspect the separate CLR latent summary in `se_latent`. The multi-draw construction can contain baseline-draw variation with `gnn = FALSE`; it should not generally be described as an MC-dropout SE.

Older saved fits without a `trait_map` use a separate continuous-only legacy prediction path and are not described by this table. For supported downstream multiple-imputation routes, consult `multi_impute_analysis` or `multi_impute` with `draws_method = "posterior"`, then use `with_imputations` and `pool_mi` under their documented eligibility conditions.

Printing

Prints dimensions, trait types, draw count and available probability/bounds components. Component presence does not establish interval coverage. The `...` argument is currently unused.

References

Little RJA (1988). "Missing-Data Adjustments in Large Surveys." *Journal of Business & Economic Statistics* 6(3): 287-296. doi:10.1080/07350015.1988.10509663. This is a source for predictive mean matching, not validation of pigauto's implementation.

van Buuren S, Groothuis-Oudshoorn K (2011). "mice: Multivariate Imputation by Chained Equations in R." *Journal of Statistical Software* 45(3), 1-67. doi:10.18637/jss.v045.i03. The official mice PMM documentation describes that software's method; it does not validate pigauto's PMM.

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,
  ordinal_levels = NULL
)

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

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. Traits already on a log scale should not be logged again: set `log_transform = FALSE` or leave them out of `log_cols` (see `impute`).

`center` logical. Subtract column means for continuous traits, log1p counts, ordinal scores, logit proportions, nonzero zero-inflated-count magnitudes, and each compositional CLR component. Binary/categorical indicators and the zero-inflation gate are unchanged. Default `TRUE`.

`scale` logical. Divide those same transformed values by their column SDs; binary/categorical indicators and the zero-inflation gate are unchanged. 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). Rows are aligned to the same species / observation order as `X_scaled` after traits are matched to the tree. Single-obs: rownames should match `tree$tip.label` (they are reordered to tip order). Multi-obs: include the same `species_col` and keep observations in the same order within each species in both frames. Repeated species labels alone do not identify individual observations: matching takes the next unused covariate row for each species. Align by an explicit observation ID before calling if within-species rows were shuffled independently. Unnamed covariates are assumed to share the input-row order of `traits` and receive the same drop / reorder. Matching `nrow` alone is not enough — unmatched species names are an error.

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).

`ordinal_levels` named list giving the scientific order of ordinal levels, e.g. `list(status = c("LC", "NT", "VU", "EN", "CR"))`. Each named column is coerced to an ordered factor with exactly those levels before type detection, so no `trait_types` entry is needed: the coerced column detects as `"ordinal"`, or as `"binary"` (order preserved) when exactly two levels are declared — the same detection an `ordered()` column with those levels receives. Observed values absent from the declared levels are an error. Same semantics as the `ordinal_levels` argument of `read_traits`. Default `NULL`.

`x` A preprocessed trait-data object.

`...` Ignored. Included for compatibility with the base `print()` generic.

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`, all nonnegative	count
`integer`, any negative value	continuous
`factor` with 2 levels	binary
`factor` (unordered) with >2 levels	categorical
`ordered` with 2 levels	binary
`ordered` with >2 levels	ordinal
`character`	converted to `factor`, then binary or categorical
`logical`	binary (FALSE = 0, TRUE = 1)

Whole-number measurements can still be scientifically continuous. Use `trait_types = c(measurement = "continuous")` to override count detection.

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")`.

Species and composition alignment: trait rows whose species are absent from the tree are dropped with a warning. Tree tips without trait rows remain in the aligned data with all trait values missing; no shared species is an error. For a `multi_proportion_groups` composition, a row is used only

when every component is present, so a partly missing composition is treated as missing as a whole. Complete compositions must be non-negative and sum to 1 within 1 percent; zero components are replaced by a small positive value for the log-ratio transform.

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 scalar means used for centering continuous, count, ordinal, proportion, and zero-inflated-count magnitude traits. Multi-proportion component means are vectors stored in the corresponding `trait_map` entry.

sds Named numeric vector of scalar SDs for those same traits. Multi-proportion component SDs are vectors stored in the corresponding `trait_map` entry.

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` whenever `species_col` is supplied, including with exactly one observation per species.

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).

The print method returns `x` invisibly.

Printing

Reports species and trait counts, and observation counts plus the observation-per-species range and median in multi-observation mode. With `trait_map`, it reports type counts and the latent-column count. Without `trait_map`, it reports column names and log-transform status. The latent-cell missing fraction is printed only when greater than zero and can differ from original-cell missingness. Covariates are stored in `x$covariates`, not displayed by this method. The dots argument is unused.

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
```

---

print.pigauto_check *Print a pigauto input check*

Description

Print a pigauto input check

Usage

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

Arguments

`x`	A `pigauto_check` object.
`...`	Unused.

Value

`x`, invisibly.

Printing

Always prints the status line. When messages are present, also prints their count and at most the first three text/action pairs. With no messages, only the status line is printed. Inspect `x$messages` for the complete structured message table; `...` is unused.

print.summary_pigauto_result *Print a pigauto result summary*

Description

Print a pigauto result summary

Usage

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

Arguments

x	A summary_pigauto_result object.
...	Unused.

Details

Prints the four aggregate counts and, when available, a generic output-role notice. It does not print the per-trait table, check details, or evaluation metrics. Inspect xper_trait, xcheck, and x$evaluation for those fields. The dots argument is unused.

Value

x, invisibly.

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 latitude / longitude centroid. The returned data.frame contains species identifiers, centroid coordinates, and occurrence counts.

Usage

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

Arguments

species	non-empty character vector of unique species names. Do not include missing or empty names. These names are used for GBIF lookups, output row names, and (when caching) cache-key generation.
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 stores centroid summaries without the raw point list. Requesting points refetches an existing centroid-only GBIF cache when the point list is absent.

Details

With the default cache_dir = NULL, results are not cached, so a later call queries GBIF again for each species. Set cache_dir to a writable directory to reuse stored results; the example below uses "script/data-cache/gbif".

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.

For cache files written by pigauto, refresh_cache = FALSE reuses a readable entry when its saved species name exactly matches the request. A live GBIF query is needed when no matching cache entry is available or refresh_cache = TRUE. GBIF is also queried again for that species when store_points = TRUE and the saved points field is absent or NULL. With cache_dir = NULL, each call queries GBIF instead of using a cache. These live-query paths require the optional rgbif package.

A cached result can still contain an NA centroid. That can mean pigauto could not match the species name, GBIF returned no records, or all records were excluded by pigauto's filters. Older cache entries may also reflect failed requests, including incomplete non-NA centroids or points when a later occurrence page failed. Set refresh_cache = TRUE to retry affected entries; this requires a live GBIF query and the optional rgbif package. A failed name lookup or occurrence request now stops with an error without caching a partial result or replacing an existing cache entry. Occurrence requests are retried before stopping.

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`. For geographic covariates, select only `centroid_lat` and `centroid_lon`. With one row per species, keep species row names so coordinates match the species in the traits and tree. With repeated observations, repeat each species' coordinates in trait-row order. Resolve missing or non-finite coordinates before fitting.

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 species-level summaries. For one row per species, select the numeric `bio*` columns for climate predictors and keep row names matching species names in the trait data and tree. For repeated observations, see `preprocess_traits` for observation-level covariate alignment. Species with no usable GBIF points can have missing climate values. This helper does not fill missing or non-finite climate values: `preprocess_traits` rejects covariates containing `NA`, `Inf`, or `-Inf`, so handle them before fitting. The `species` column is an identifier and `n_extracted` is a diagnostic, so do not include them as climate predictors.

Rasters are checked and loaded for the requested `resolution`; if its cache lacks the completion marker or any of the 19 raster files, the function attempts to download the stack. Per-species extracts are cached by species name, not resolution. When changing resolution, use a separate `worldclim_cache_dir` or set `refresh_cache = TRUE` to avoid reusing extracts made at another resolution. Refreshing extracts does not force raster downloads. If loading the stack fails, existing cached extracts can still be used; a new extraction at usable GBIF locations needs a readable raster stack.

Extraction uses the raw per-occurrence point list when the GBIF cache contains it (request it with `store_points = TRUE` in `pull_gbif_centroids`). A centroid-only cache falls back to the species centroid. Requesting points refetches a centroid-only GBIF cache, but an existing WorldClim extract remains cached until `refresh_cache = TRUE` is used here.

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	Supply a non-empty character vector of unique species names. Do not include missing or empty names; names are used to create cache keys and output row names.
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

```
# With fresh caches, this example contacts GBIF and may download the 19
# global WorldClim rasters. It needs internet access and the optional
# rgbif and terra packages.
sp <- c("Quercus alba", "Pinus taeda", "Acer saccharum")
# Use temporary caches for this R session. Choose a persistent directory
# if you want to reuse downloaded data after R restarts.
cache_root <- file.path(tempdir(), "pigauto-worldclim-example")
gbif_cache <- file.path(cache_root, "gbif")
worldclim_cache <- file.path(cache_root, "worldclim")
# store_points = TRUE refetches centroid-only GBIF entries when needed.
gbif_df <- pull_gbif_centroids(sp,
  cache_dir = gbif_cache, store_points = TRUE)
# Refresh old climate extracts. Occurrence points are used when available;
# centroid-only GBIF cache entries fall back to the species centroid.
wc_df <- pull_worldclim_per_species(
  species = sp,
  gbif_cache_dir = gbif_cache,
  worldclim_cache_dir = worldclim_cache,
  refresh_cache = TRUE)
# For one row per species, select climate columns and align by species.
wc_cov <- wc_df[, grep("^bio", names(wc_df)), drop = FALSE]
rownames(wc_cov) <- wc_df$species
gbif_cov <- gbif_df[match(wc_df$species, gbif_df$species),
                    c("centroid_lat", "centroid_lon"), drop = FALSE]
rownames(gbif_cov) <- wc_df$species
cov <- cbind(gbif_cov, wc_cov)
# This helper does not fill missing values. Resolve NA/Inf/-Inf before
# passing cov to impute().
```

---

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`. All species identifiers must be present and unique because row names identify one trait row per species. For repeated observations, use `read.csv()` and pass the resulting data frame with `species_col` to `check_pigauto()` and `impute()`. 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,
  ordinal_levels = 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. Without ordinal_levels, levels follow R's default sorting order.
ordinal_levels	named list specifying the scientific order of ordinal levels, for example list(status = c("LC", "NT", "VU", "EN", "CR")). Each named column is converted to an ordered factor. Unlisted observed values are an error, so a misspelling cannot silently become missing.

Value

A data.frame with unique 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.

Details

For bird phylogenies, BirdTree provides a species-subset download tool for up to 2,500 species and full-tree distributions for larger sets. Download and unzip the trees, then read a Newick tree file here. BirdTree documents its distributions as Newick trees; this function also accepts NEXUS files. A file with one tree returns a `"phylo"` object; a file with several trees returns a `"multiPhylo"` object.

pigauto does not download BirdTree data automatically. Pass a single-tree result to `impute(traits, tree)`. A multi-tree result can be passed to `multi_impute_trees(traits, trees)` for experimental prediction- sensitivity checks only; this path has not been validated for downstream inference.

BirdTree's subset tool defaults to at least 100 draws and recommends a reasonable sample (more than 100) for analyses using its full-tree distributions. pigauto's `multi_impute_trees()` remains experimental: it supports descriptive checks of prediction sensitivity only, and its stochastic datasets are not validated for downstream inference. See `vignette("tree-uncertainty")`.

BirdTree asks researchers using full or partial tree data to cite Jetz et al. (2012). If you use the BirdTree web tool, also credit BirdTree.org.

Value

An object of class `"phylo"` when the file contains one tree, or `"multiPhylo"` when it contains multiple trees.

References

Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). The global diversity of birds in space and time. *Nature*, 491, 444-448. doi:10.1038/nature11631. BirdTree.org. Downloads, phylogeny subsets, and FAQ. https://birdtree.org/downloads/ https://birdtree.org/subsets/ https://birdtree.org/faq/.

See Also

`multi_impute_trees`

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, for later loading with `load_pigauto()`. Matching pigauto versions alone do not guarantee cross-version or cross-platform compatibility.

Usage

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

Arguments

fit	pigauto_fit object, or a pigauto_result from impute (its $fit is saved).
path	character. File path to save to (recommended extension: .pigauto).
compress	logical. Use gzip compression (default TRUE).

Details

Use save_pigauto() and load_pigauto() as the package's save/load interface. For GNN fits containing torch tensors, native saveRDS() / readRDS() does not preserve a usable tensor state; this function serialises that state before saving. pigauto checks its package version after loading the tensors, but does not check R, torch/LibTorch, or operating-system compatibility. Retain the software environment used to save the fit and verify loading and predictions in the destination environment. GNN-off fits have an empty model state and do not use this tensor serialization path.

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")
suppressMessages(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 the Brownian motion baseline and a pigauto model, and compares performance. GNN training in the pigauto model is controlled by the gnn argument. Use it for exploratory comparisons and workflow checks. Its default of three replicates, and the one-replicate example, do not by themselves establish performance. For consequential performance claims, pre-specify a design, use replication suited to the question, and account for every requested attempt, including failures. The result summary alone

does not preserve that denominator. The baseline is fitted first with `fit_baseline()`'s defaults: estimated continuous-family and discrete-trait lambda, `joint_solver = "inhouse"`, and automatic exact/per-column routing where the joint path is eligible. Arguments in `...` are forwarded only to `fit_pigauto()` after the baseline is fitted; they do not change its settings. To compare another baseline configuration, construct that baseline explicitly in a custom evaluation workflow.

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,
  gnn = TRUE,
  ...
)

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

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

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.
`gnn`	logical. Whether the pigauto arm trains the GNN. Default `TRUE`, so this function keeps comparing the baseline with the GNN-corrected model even though `fit_pigauto` and `impute` default to `gnn = FALSE`.
`...`	additional arguments passed to `fit_pigauto`.
`x`	A benchmark object.
`object`	A benchmark object to summarize.

Details

Available scenarios:

`"BM"` Independent-trait Brownian-motion control. Each trait is generated separately; no cross-trait covariance is imposed. The fitted baseline estimates parameters rather than using known generating values. Finite-sample changes in GNN performance are measured outcomes, not a promised benefit from cross-trait correlation.

`"OU"` Ornstein-Uhlenbeck traits with a specified stabilising-selection parameterisation. This differs from the BM control; the scenario definition alone does not establish which method predicts better.

`"regime_shift"` Two-regime BM with opposite additive offsets for two clades. The resulting distribution depends on the tree and offset; bimodality is not guaranteed.

`"nonlinear"` Traits are generated with non-linear inter-trait relationships. Any difference in predictive performance is an empirical outcome, not a guaranteed advantage for either method.

`"mixed"` Mixed trait types: 2 continuous + 1 binary + 1 categorical trait (3 levels). This scenario exercises these three types together; it does not cover every supported trait type.

Value

An object of class `"pigauto_benchmark"` with:

results data.frame with columns: `scenario`, `rep`, `method`, `trait`, `type`, `metric`, `value`, `n_test`. Failed attempts are caught and omitted; their messages are suppressed when `verbose = FALSE`. There is no returned attempt/failure table.

summary data.frame averaged over surviving metric rows. Its `n_reps` column counts non-missing values for that metric, rather than all requested attempts.

scenarios character vector of scenarios run.

n_reps integer: requested replicates per scenario.

n_species integer.

For scientific method comparisons, retain a separate attempt/failure table and reconcile every requested replicate before interpreting the returned summaries; this function alone does not preserve that denominator.

The print method returns `x` invisibly.

The summary method returns `object$summary` invisibly.

Printing

Prints requested replicates and aggregate metrics from surviving result rows. The `...` argument is currently unused.

Summary method

`summary()` prints scenario metrics and relative RMSE changes using the surviving rows. Relative changes can be undefined for a zero baseline RMSE. Its `...` argument is unused.

Examples

```
bench <- simulate_benchmark(n_species = 20L, n_traits = 2L,
                            scenarios = "BM", epochs = 5L, n_reps = 1L,
                            verbose = FALSE, gnn = 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). Use at least 2 for `scenario = "nonlinear"`: the current generator always returns `trait1` and `trait2`, even if a smaller number is requested.
`scenario`	Character. One of `"OU"`, `"regime_shift"`, `"nonlinear"`.
`alpha`	Numeric. OU pull strength (only for `"OU"`).
`theta`	Numeric. Base OU optimum (only for `"OU"`). Trait j uses `theta + 0.5 * (j - 1)` for both its optimum and root value.
`sigma`	Numeric. BM diffusion rate.
`shift_magnitude`	Numeric. Raw additive regime offset (only for `"regime_shift"`). Trait j uses `shift_magnitude * (1 + 0.2 * (j - 1))`, added to one clade and subtracted from the other. It is not multiplied by a trait SD; `sigma` separately controls the underlying BM diffusion.
`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

Rank currently missing cells or species for follow-up observation

---

Description

Use this function with a fitted `pigauto_result` (returned by `impute`) to shortlist currently missing species-trait cells or species for follow-up measurement. It estimates expected BM variance reduction for continuous-family traits and label-propagation entropy reduction for discrete traits. These model-based scores can guide a shortlist, but they do not guarantee a gain in a field study or downstream analysis and have not been validated against field-sampling outcomes. They use the BM or label-propagation baseline formulas, not predictions from the fitted GNN.

Usage

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

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

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. With `by = "cell"`, rows are sorted globally by decreasing `delta` and truncated to the first `top_n` rows after applying the `types` filter. Set `top_n` large enough to retain the candidates you need before applying any further filters, then compare candidates within the same trait and metric; cross-trait or cross-metric scores do not represent a common biological sampling value. With `by = "species"`, species are ranked by summed variance reduction when positive, otherwise by summed entropy reduction.

by	character, one of "cell" (default) or "species". "cell" returns individual (species, trait) pairs. "species" returns species, delta_var_total, delta_entropy_total, and n_traits_missing, with the two reductions summed separately across 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, and multi_proportion.
x	A pigauto_active object returned by suggest_next_observation().
...	Further arguments passed to the data-frame print method when rankings are nonempty.

Details

A practical reading guide

1. Choose by = "cell" for species-trait candidates or by = "species" for a species-level shortlist.
2. For cell results, compare scores only within the same trait and metric. Variance and entropy use different units, and variance scores for different traits may use different transformed scales. Species totals are approximate across-trait rankings.
3. Set top_n large enough before filtering; the function truncates the ranking first. Use the scores to shortlist candidates, then choose field measurements using feasibility and biological priorities. The scores have not been validated against field outcomes.

Two metrics are supported, dispatched by trait type:

Continuous-family traits (continuous, count, ordinal, proportion) use the BM variance-reduction formula. This 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. The equation below gives the expected total variance reduction when candidate cell (s, t) is observed.

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

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. Within one trait, a larger ΔV means the model expects a greater reduction in the remaining BM prediction variance across missing cells if that candidate is measured.

These variance scores use the untransformed tree correlation (Pagel's $\lambda = 1$), even when the fitted baseline used another value, so their candidate ranking may differ from one calculated under the fitted baseline. Treat them as screening scores.

Discrete traits (binary, categorical) use a label-propagation expected-entropy-reduction formula. Here sim contains phylogenetic similarity weights between species; observed classes with larger weights contribute more to the LP probabilities. 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).

Scores require at least one missing cell. Continuous-family and multi-proportion variance scores require at least five observed cells. Binary and categorical entropy scores require at least one observed class. For `zi_count`, gate entropy requires at least one observed gate value; its magnitude variance contribution is zero when fewer than five non-zero magnitudes are observed. Traits that do not meet these conditions are omitted, so the result can be empty.

Variance and entropy are not directly comparable. For `by = "cell"`, rows are sorted globally by `delta` and then truncated to `top_n`, including across metrics. Compare candidates within the same trait and metric. If you plan to filter the returned rows, set `top_n` large enough to retain the candidates you want; filtering cannot recover rows already removed by the limit. For `by = "species"`, output has four columns: `species`, `delta_var_total`, `delta_entropy_total`, and `n_traits_missing`; the two totals are summed separately across that species' included missing traits. Species are ranked by `delta_var_total` when it is positive, otherwise by `delta_entropy_total`; this is only an approximate cross-metric ranking because the measures have different units.

Variance reductions across traits can also be measured on different transformed scales, so variance-only species totals are not a common measure of biological value. Treat these outputs as model-based screening scores: compare cell candidates within the same trait and metric, and use cross-trait species rankings to shortlist candidates rather than to claim equal sampling value. Choose among shortlisted candidates using field feasibility and biological priorities. The package does not validate these rankings against field-sampling outcomes.

`zi_count`: 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 = $p_{\text{gate}} \times \Delta V_{\text{mag}}$) 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. The current magnitude calculation also treats structurally missing magnitudes for species with an observed zero gate as targets, although those values cannot be observed. Thus the total can overstate variance reduction across magnitudes that could actually be measured.

`multi_proportion`: 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 entropy-only rows), and `delta_entropy_total` (NA for variance-only rows), sorted globally by decreasing `delta`, including across metrics. Cross-metric order is approximate because the metrics have different units. When `by = "species"` and rows are returned, columns are: `species`, `delta_var_total`, `delta_entropy_total`, `n_traits_missing`, sorted by `delta_var_total` when positive and otherwise by `delta_entropy_total`. If no candidates remain, the empty result instead has the seven cell-mode columns and does not include `n_traits_missing`.

Invisibly returns `x`. For nonempty rankings, the printed object is converted to a plain data frame, so the returned object no longer has the `"pigauto_active"` class or its `"by"` attribute. An empty result returns the original object unchanged.

Printing

The print method always reports the ranking mode and number of rows. Nonempty rankings are printed as a plain data frame, with ... forwarded to its print method, followed by the proxy-ranking caution. An empty result means no eligible candidates under the selected trait types and scoring requirements. Missing cells may remain when their trait does not meet a minimum-observation requirement. The console message reports candidate eligibility; it does not mean that the whole dataset is complete. The nonempty branch invisibly returns the plain data frame without the by attribute; the empty branch invisibly returns the original x.

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)
# Request enough rows before filtering to one trait.
ranked <- suggest_next_observation(res, top_n = 50L)
subset(ranked, trait == "Mass")
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. For a trait represented by multiple latent columns, the displayed gate uses its first latent column. 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, the fit has no test cells, or evaluation fails. If evaluation fails, the method prints a message and returns NULL.

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)
# This small test set illustrates the output; it is too small for a
# stable performance conclusion. Use a held-out design sized for your
# question. Repeated splits or seeds assess split/training variability,
# but do not by themselves establish generalisation.
summary(fit, data = pd)
```

---

summary.pigauto_result

Summarise a pigauto result

Description

Summarise a pigauto result

Usage

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

Arguments

object	A pigauto_result.
...	Unused.

Details

The summary object is returned visibly. Its print method shows only the four aggregate counts and a generic role notice. Inspect the corresponding fields for per-trait counts, check details, and evaluation metrics; these are not printed. The dots argument is unused.

Value

Visibly returns a `summary_pigauto_result` list with:

`check` The stored input check, or a notice when unavailable.

`counts` `filled`: successfully filled target trait cells; `observed`: observed target trait cells; `unresolved`: target trait cells still missing; `not_modeled`: missing trait cells in excluded data-only rows.

`per_trait` A data.frame of per-trait successful fill counts.

`output_roles` Descriptions of completed data, prediction, uncertainty, conformal, and inference roles.

`evaluation` The stored evaluation results.

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

Description

An object of class `'phylo'` from the **ape** package. It is posterior sample member 69 from the Hackett-backbone half of BirdTree's 100-tree bird collection, retrieved through the **megatrees** package and pruned to the 300 species in `avonet300`. The MIT software licence for **megatrees** does not establish rights for the underlying BirdTree data; see the installed '`NOTICE`'. This object is included for examples, not as an MCC or consensus tree or a recommended tree for substantive analysis.

Usage

```
tree300
```

Format

An object of class `phylo` with 300 tips.

Source

Example phylogeny drawn from the BirdTree Hackett-backbone collection. The underlying BirdTree distribution is described by Jetz et al. (2012); see the installed '`NOTICE`' for data provenance and reuse information.

References

Hackett SJ, Kimball RT, Reddy S, et al. (2008). A phylogenomic study of birds reveals their evolutionary history. *Science*, 320(5884), 1763-1768. doi:10.1126/science.1157704.

Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). The global diversity of birds in space and time. *Nature*, 491, 444-448. doi:10.1038/nature11631.

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

Description

A `multiPhylo` list of 50 phylogenetic trees sampled from the 100-tree BirdTree bird collection retrieved through **megatrees**. The collection's first 50 trees use the Ericson backbone and its next 50 use the Hackett backbone. The fixed sample generated with seed 42 contains 28 Ericson and 22 Hackett trees, each pruned to the 300 species in `avonet300`. Topologies and branch lengths vary across this posterior sample. The MIT software licence for **megatrees** does not establish rights for the underlying BirdTree data; see the installed '`NOTICE`'.

Usage

```
trees300
```

Format

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

Details

Use with `multi_impute_trees` to compare point predictions across posterior-tree choices. This is a prediction-sensitivity check only; the resulting datasets cannot be pooled for downstream inference.

Source

Example phylogenies drawn from the BirdTree collection. The source collection and its Ericson and Hackett backbones are described by Ericson et al. (2006), Hackett et al. (2008), and Jetz et al. (2012).

References

Ericson PGP, Anderson CL, Britton T, et al. (2006). Diversification of Neoaves: integration of molecular sequence data and fossils. *Biology Letters*, 2(4), 543-547. doi:10.1098/rsbl.2006.0523.

Hackett SJ, Kimball RT, Reddy S, et al. (2008). A phylogenomic study of birds reveals their evolutionary history. *Science*, 320(5884), 1763-1768. doi:10.1126/science.1157704.

Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). The global diversity of birds in space and time. *Nature*, 491, 444-448. doi:10.1038/nature11631.

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. It is posterior sample member 69 from the Hackett-backbone half of BirdTree's 100-tree bird collection, pruned to the species present in `avonet_full` rather than the 300-species example subset. The MIT software licence for **megatrees** does not establish rights for the underlying BirdTree data; see the installed 'NOTICE'. This object is included for examples, not as an MCC or consensus tree or a recommended tree for substantive analysis.

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

Example phylogeny drawn from the BirdTree Hackett-backbone collection. The underlying BirdTree distribution is described by Jetz et al. (2012); see the installed 'NOTICE' for data provenance and reuse information.

References

Hackett SJ, Kimball RT, Reddy S, et al. (2008). A phylogenomic study of birds reveals their evolutionary history. *Science*, 320(5884), 1763-1768. doi:10.1126/science.1157704.

Jetz W, Thomas GH, Joy JB, Hartmann K, Mooers AO (2012). The global diversity of birds in space and time. *Nature*, 491, 444-448. doi:10.1038/nature11631.

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 an analysis-aware MI object and return the list of fits. This is the documented software fitting entry for `multi_impute_analysis()` and for `multi_impute()` with `draws_method = "posterior"`. Prediction-diagnostic (`"conformal"`, `"mc_dropout"`) and posterior-tree sensitivity completions stop with an actionable error before any model is fitted. The workflow is experimental and limited to fixed-effect coefficients and their covariance matrices. Passing software eligibility checks means the package permits fitting; it does not establish inferential performance across tested regimes or compatibility with every downstream model.

Usage

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

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

Arguments

`mi`	A `pigauto_analysis_mi` object returned by `multi_impute_analysis()`, or a `pigauto_posterior_mi` object returned by `multi_impute()` with `draws_method = "posterior"` (and `param_uncertainty = "full"`, the default, or `"both"`). Legacy `pigauto_mi` objects, diagnostic completions, tree-sensitivity completions, plug-in posterior draws, and bare dataset lists are not supported fitting inputs.
`.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.
`...`	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.
`x`	A list of downstream imputation fits.

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.

The print method returns `x` invisibly.

Posterior draws and congeniality

The `draws_method = "posterior"` completions come from a linear-Gaussian phylogenetic mixed model of the imputed traits only, on the scale where they were imputed (the log scale for traits log-transformed by `log_transform`). The documented software route permits fitting analyses that are linear in the imputed traits on that scale and whose variables are among the imputed traits. This eligibility does not establish that an analysis model is compatible with the imputation model or that inference is calibrated. In the registered validation, five tested posterior slope regimes failed and three were indeterminate; see the registered inferential results. The route has not established support outside its measured regimes. Not covered: covariates from outside the imputed traits, nonlinear terms (such as squares) or interactions among imputed traits, and analysing a log-transformed trait on its raw scale. The formula passed to `.f` is not checked; this is the analyst's responsibility. Plug-in draws from `posterior_control$param_uncertainty = "none"` (validation only) are refused.

Printing

Prints successful and failed fit counts and the first successful fit class. Failed elements remain in the list until pooling removes them. The `...` argument is currently unused.

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_
# Workflow example only: m = 2L is not an analysis recommendation.
# Choose enough imputations to make Monte Carlo error small relative to
# the standard errors you report.
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

∗ **datasets**
avonet300, 3
avonet_full, 4
ctmax_sim, 15
tree300, 103
tree_full, 105
trees300, 104

avonet300, 3, *4, 5*, *59*, *103, 104*
avonet_full, 4, *105*

build_phylo_graph, 5, *19*, *26*

calibration_df, 7
check_pigauto, 7
compare_methods, 9, *69*
completed_data, 11
completed_data(), *33*
confusion_matrix, 12
cross_validate, 13
cross_validate(), *9*
ctmax_sim, 15

evaluate, 16, *18*, *69*
evaluate_imputation, *14*, 17, *69*, *78*

fit_baseline, *6*, 19, *21*, *24*, *26*, *30*, *33*, *36, 37*
fit_baseline(), *61*
fit_pigauto, *10*, *22*, 24, *33*, *38*, *79*, *95*
fit_pigauto(), *38*, *48–50*, *60*

get_imputed_datasets, 32
get_imputed_datasets(), *11*

impute, *10*, *15*, 33, *64*, *72*, *83*, *89, 90*, *94, 95*, *98*, *101*
impute(), *7, 8*, *11*, *33*, *46*, *48–50*, *55*, *58–60*

load_pigauto, 42

make_missing_splits, *17*, *19*, *26*, 43, *64*, *67*
mask_missing, 45
multi_impute, *35*, *39*, 45, *81*
multi_impute(), *11*, *32, 33*, *57, 58*, *61*, *63*, *72*, *106*
multi_impute_analysis, 56, *81*
multi_impute_analysis(), *32*, *50*, *53*, *55*, *63*, *72*, *76*, *106, 107*
multi_impute_trees, 58, *93*, *104*
multi_impute_trees(), *32, 33*, *53*, *61*, *77*

name_backbone, *88*

occ_search, *88*

pigauto_data, *52*
pigauto_fit, *52*
pigauto_report, 64
plot.pigauto_benchmark, 65
plot.pigauto_fit, 66, *68*
plot.pigauto_pred, 67
plot.pigauto_result, 68
plot_comparison, 69
plot_history_gg, 70
plot_missingness, 70
plot_uncertainty, 71
pool_mi, 72, *81*
pool_mi(), *33*, *47*, *49–53*, *57*, *61*, *106, 107*
predict(), *52*
predict.pigauto_fit, *17*, *38, 39*, *71*, 76
predict.pigauto_fit(), *51*, *61*
preprocess_traits, *3*, *13*, *33*, *43*, 82, *89*
preprocess_traits(), *8*, *48*, *59*
print.pigauto_active (suggest_next_observation), 98
print.pigauto_analysis_mi (multi_impute_analysis), 56
print.pigauto_benchmark (simulate_benchmark), 94
print.pigauto_check, 86
print.pigauto_cv (cross_validate), 13

print.pigauto_data *(*preprocess_traits*)*, 82
print.pigauto_fit *(*fit_pigauto*)*, 24
print.pigauto_mi *(*multi_impute*)*, 45
print.pigauto_mi_fits *(*with_imputations*)*, 106
print.pigauto_mi_trees *(*multi_impute_trees*)*, 58
print.pigauto_pooled *(*pool_mi*)*, 72
print.pigauto_pred *(*predict.pigauto_fit*)*, 76
print.pigauto_result *(*impute*)*, 33
print.summary_pigauto_result, 87
pull_gbif_centroids, 87, *89, 90*
pull_worldclim_per_species, *88*, 89

read_traits, *39*, *83*, 91
read_traits(), *8*, *50*, *60*
read_tree, 92

save_pigauto, *42*, 93
simulate_benchmark, 94
simulate_non_bm, 97
stats::glm, *73*
stats::lm, *73*
suggest_next_observation, 98
summary.pigauto_benchmark *(*simulate_benchmark*)*, 94
summary.pigauto_cv *(*cross_validate*)*, 13
summary.pigauto_fit, 101
summary.pigauto_result, 102

tree300, *15*, 103, *104, 105*
tree_full, *5*, 105
trees300, *59*, *63*, 104

with_imputations, *81*, 106
with_imputations(), *33*, *47*, *49–53*, *57*, *61*, *72*, *73*, *76*