Package 'deepSTRAPP'

October 29, 2026

Title Test for Differences in Diversification Rates over Time

Version 1.1.1

Description Employ time-calibrated phylogenies and trait/range data to test for differences in diversification rates over evolutionary time. Extend the STRAPP test from BAMMtools::traitDependentBAMM() to any time step along phylogenies. See inst/COPYRIGHTS for details on third-party code.

License MIT + file LICENSE

Encoding UTF-8

RoxygenNote 7.3.2

URL https://github.com/MaelDore/deepSTRAPP, https://maeldore.github.io/deepSTRAPP/

BugReports https://github.com/MaelDore/deepSTRAPP/issues

Imports ape (>= 5.8.0), BAMMtools (>= 2.1.12), cladoRcpp (>= 0.15.1), coda (>= 0.19-4.1), cowplot (>= 1.1.3), dplyr (>= 1.1.4), dunn.test (>= 1.3.6), geiger (>= 2.0.11), ggplot2 (>= 3.5.0), methods (>= 4.4.2), phytools (>= 2.0.0), plyr (>= 1.8.9), purrr (>= 1.0.2), qpdf (>= 1.4.1), RColorBrewer (>= 1.1-3), Rcpp (>= 1.0.14), rlang (>= 1.1.6), scales (>= 1.3.0), stringr (>= 1.5.1), tidyr (>= 1.3.1)

Depends R (>= 4.4)

LazyData true

LazyDataCompression xz

Suggests knitr, maps (>= 3.4.2), parallel (>= 4.4.2), BioGeoBEARS (>= 1.1.3), rmarkdown, contsimmap (>= 0.1.0)

Additional_repositories https://maeldore.github.io/drat

LinkingTo Rcpp

VignetteBuilder knitr

NeedsCompilation yes

Author Maël Doré [aut, cre, cph] (ORCID:
<https://orcid.org/0000-0003-3242-9242>),
Dan Rabosky [ctb],
Mike Grundler [ctb],
Huateng Huang [ctb],
Pascal Title [ctb],
Liam Revell [ctb],
Nicholas J. Matzke [ctb]

Maintainer Maël Doré <mael.dore@gmail.com>

Repository CRAN

Date/Publication 2026-10-29 11:00:02 UTC

Contents

aggregate_contMaps . . . 3
BAMM_template_diversification . . . 6
build_BAMM_object . . . 7
compute_STRAPP_test_for_focal_time . . . 11
convert_BSM_to_simmap . . . 20
convert_contsimmap_to_contMaps . . . 24
convert_simmaps_to_densityMaps . . . 26
cut_contMaps_for_focal_time . . . 29
cut_contMap_for_focal_time . . . 32
cut_densityMaps_for_focal_time . . . 34
cut_densityMap_for_focal_time . . . 37
cut_phylo_for_focal_time . . . 40
cut_simmaps_for_focal_time . . . 42
cut_simmap_for_focal_time . . . 46
eel_biogeo_data . . . 49
eel_cat_3lvl_data . . . 50
extract_all_trait_values_for_focal_time . . . 51
extract_diversification_data_melted_df_for_focal_time . . . 58
extract_most_likely_trait_values_for_focal_time . . . 59
extract_trait_data_melted_df_for_focal_time . . . 66
mammals . . . 68
plot_BAMM_rates . . . 69
plot_contMap . . . 73
plot_densityMaps_overlay . . . 75
plot_histograms_STRAPP_tests_over_time . . . 77
plot_histogram_STRAPP_test_for_focal_time . . . 83
plot_rates_through_time . . . 89
plot_rates_vs_trait_data_for_focal_time . . . 94
plot_rates_vs_trait_data_over_time . . . 100
plot_STRAPP_pvalues_over_time . . . 107
plot_traits_vs_rates_on_phylogeny_for_focal_time . . . 109
plot_traits_vs_rates_on_phylogeny_over_time . . . 117
Ponerinae_binary_range_table . . . 124

Ponerinae_biogeo_data_old_calib 125
Ponerinae_cat_2lvl_data_old_calib 126
Ponerinae_cat_3lvl_data_old_calib 127
Ponerinae_trait_cont_tip_data_10My 128
Ponerinae_trait_tip_data 129
Ponerinae_tree 129
Ponerinae_tree_old_calib 130
prepare_diversification_data 131
prepare_trait_data 139
prune_BAMM_object 149
run_deepSTRAPP_for_focal_time 156
run_deepSTRAPP_over_time 169
select_best_model_from_BioGeoBEARS 186
select_best_trait_model_from_geiger 189
subset_BAMM_object 192
update_rates_and_regimes_for_focal_time 195
whale_BAMM_object 201

Index **204**

`aggregate_contMaps` *Aggregate a list of contMaps into a unique mean/median contMap*

Description

Aggregate a list of contMaps into a unique mean/median contMap as produced by the `phytools::contMap()` function.

Usage

```
aggregate_contMaps(
  contMaps,
  fun = "mean",
  display_plot = TRUE,
  ...,
  color_scale = NULL,
  verbose = TRUE
)
```

Arguments

`contMaps`	List of objects of class `"contMap"` that represent independent simulations of the evolution of a continuous trait (i.e., continuous stochastic maps).
`fun`	Character string. Select the aggregating function. Available options are `"mean"` and `"median"`. Default = `"mean"`.
`display_plot`	Logical. Whether to plot the resulting aggregated contMap. Default = `TRUE`.
`...`	List of named arguments. Additional arguments to be passed down to `deepSTRAPP::plot_contMap()`.

`color_scale` Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` showing the evolution of the continuous trait. From lowest values to highest values. Default (`color_scale = NULL`) is using the color palette recorded in the `contMap$cols` item. If none was provided, the `rainbow()` palette is used. Provide the full argument name to avoid ambiguity with the `col` graphical argument.

`verbose` Logical. Whether to display progress every 100 edges.

Details

The function is primarily designed to average trait values across multiple continuous stochastic maps produced from the same simulation, typically with `contsimmap::make.contsimmap()` and then converted to a list of contMaps (phytools format) with the `convert_contsimmap_to_contMaps()` function.

The result is a unique `contMap` which should be consistent with the `contMap` produced by `phytools::contMap()` when interpolating maximum likelihood estimates of ancestral trait values between nodes.

Value

Returns a unique `"contMap"` object representing the average trait evolutionary history recorded across all continuous stochastic maps provided as input in `contMaps`.

The resulting aggregated `"contMap"` is a list with three items:

- `$tree` A list of classes `"simmap"` and `"phylo"`. The mapped phylogeny including:
  - `$maps` A list of named numeric vectors. Provides the mapping of aggregated trait values along each edge (scaled between 0 and 1000).
  - `$mapped.edge` A numeric matrix. Provides the evolutionary time spent across trait values (columns) along the edges (rows).
- `$cols` A named character vector mapping values to colors used to display trait evolution along the branches.
- `$lims` A numeric vector. Minimum and maximum aggregated trait values recorded.

Author(s)

Maël Doré

References

For continuous stochastic mapping: Martin, B. S., & Weber, M. G. (2026). Stochastic character mapping of continuous traits on phylogenies. Systematic Biology, syag031. doi:10.1093/sysbio/syag031.

See Also

`contsimmap::make.contsimmap()` `phytools::contMap()` `plot_contMap()`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## The R package 'contsimmap' is needed for this example to work.
 # Please install it manually from: https://github.com/bstaggmartin/contsimmap.

 # ----- Prepare data ----- #

 # Load eel phylogeny and tip data from the R package phytools
 # Source: Collar et al., 2014; DOI: 10.1038/ncomms6505

 data("eel.tree", package = "phytools")
 data("eel.data", package = "phytools")

 # Extract body size
 eel_tip_data <- stats::setNames(eel.data$Max_TL_cm,
                                 rownames(eel.data))

 # ----- Run continuous stochastic mapping ----- #

  # (May take several seconds to run)
 eel_contsimmap <- contsimmap::make.contsimmap(
    tree = eel.tree,
    trait.data = eel_tip_data ,
    nsim = 100, res = 100,
    verbose = TRUE)

 dim(eel_contsimmap) # 121 edges, 1 trait, 100 simulations

 # ----- Convert to a list of contMaps ----- #

 eel_contMaps <- convert_contsimmap_to_contMaps(
    contsimmap = eel_contsimmap, verbose = TRUE)

 # ----- Explore contMaps ----- #

 # Plot contMap from simulation n°1
 plot_contMap(eel_contMaps[[1]])
 # Plot contMap from simulation n°50
 plot_contMap(eel_contMaps[[50]])

 # ----- Aggregate all simulations ----- #

 eel_aggregated_contMap <- aggregate_contMaps(
    contMaps = eel_contMaps,
    verbose = TRUE, display_plot = FALSE)

 plot_contMap(eel_aggregated_contMap)

 # ----- Compare to interpolated contMap from phytools ----- #

 # Produce interpolated contMap with phytools
```

```
eel_contMap <- phytools::contMap(tree = eel.tree, x = eel_tip_data,
                                 res = 100, # Number of time steps
                                 plot = FALSE)

# Plot the interpolated contMap mapping Maximum Likelihood estimates
# of ancestral trait values interpolated between nodes
plot_contMap(eel_contMap)
# Plot the aggregated contMap representing average trait values
# from true continuous stochastic mapping simulations
plot_contMap(eel_aggregated_contMap)
# Both should be visually similar

}
```

`BAMM_template_diversification`

Template file for BAMM diversification analyses

Description

Template file for BAMM diversification analyses provided as character strings.

Source: bamm-2.5.0 References: http://bamm-project.org/; https://github.com/macroevolution/bamm

Usage

```
data(BAMM_template_diversification)
```

Format

A vector of 260 character strings.

Details

A vector of 260 character strings that can be displayed with `print(BAMM_template_diversification)`. This is the template used to generate the 'config_file.txt' controlling settings used for a BAMM run. It provides detailed explanations of the `additional_BAMM_settings` that can be used in `prepare_diversification_data()` to customize the BAMM run. It is called internally by `prepare_diversification_da` to produce the custom 'config_file' used in the subsequent BAMM run.

References

Authors: Daniel Rabosky (BAMM) & Pascal Title (`{BAMMtools}`). Modified by Maël Doré for deepSTRAPP.

See Also

BAMM software website: `http://bamm-project.org/`

`build_BAMM_object` *Build a BAMM object for a deepSTRAPP run*

Description

Build a BAMM object of class `bammdata` based on the output file of a BAMM run that contains a phylogenetic tree and associated diversification rates mapped along branches across BAMM posterior samples.

The `BAMM_object` output is typically used as input to run deepSTRAPP with `run_deepSTRAPP_for_focal_time()` or `run_deepSTRAPP_over_time()`.

This is a wrapper of the original `BAMMtools::getEventData()` function that additionally provides information on:

- the Marginal Shift Probability (MSP) = the probability of a regime shift to occur along each branch.
- the Maximum A Posteriori probability (MAP) configurations among the posterior samples = the configurations of regime shifts that were sampled most frequently (See `BAMMtools::getBestShiftConfiguration`
- the Maximum Shift Credibility (MSC) configurations among the posterior samples = the configurations of regime shift location with the highest product of marginal probabilities across branches (See `BAMMtools::maximumShiftCredibility()`).

Those additional elements are used by `plot_BAMM_rates()` to display regime shift probabilities and locations.

This function enables users to integrate the results of their own BAMM analyses into the deepSTRAPP framework. Alternatively, a full BAMM analysis starting from a time-calibrated phylogeny alone can be carried out within deepSTRAPP with `prepare_diversification_data()`.

Usage

```
build_BAMM_object(
  phylo,
  eventdata,
  burn_in = 0.25,
  nb_posterior_samples = NULL,
  seed = NULL,
  expectedNumberOfShifts,
  MAP_odds_ratio_threshold = 5,
  verbose = FALSE
)
```

Arguments

`phylo`	Object of class `"phylo"` as defined in R package `{ape}`. Time-calibrated phylogeny that was used to produce the BAMM run. The phylogeny must be rooted and fully resolved.

`eventdata` Character string specifying the path to a BAMM event-data file. Alternatively, an object of class data.frame that includes the event data from a BAMM run.

`burn_in` Numeric. Proportion of posterior samples removed from the BAMM output to ensure that the remaining samples were drawn once the equilibrium distribution was reached. Default is `0.25`

`nb_posterior_samples`
Integer. Number of posterior samples to extract, after removing the burn-in, in the final `BAMM_object` to use for downstream analyses. If set to `NULL` (default), all samples remaining after removing the burn-in will be kept.

`seed` Integer. Set the seed to ensure reproducibility when drawing random posterior samples. Default is `NULL` (a random seed is used).

`expectedNumberOfShifts`
Integer. The expected number of regime shifts set during the BAMM run as a hyperparameter controlling the exponential prior distribution used to modulate reversible jumps across model configurations in the rjMCMC run. This is needed to compute priors for regime shift along branches.

`MAP_odds_ratio_threshold`
Numeric. Controls the definition of 'core-shifts' used to distinguish across configurations when fetching the MAP samples. Shifts that have an odds ratio of marginal posterior probability / prior lower than `MAP_odds_ratio_threshold` are ignored. See `BAMMtools::getBestShiftConfiguration()`. Default = `5`.

`verbose` Logical. Whether to display progress in the console. Default = `FALSE`.

Value

The function returns a `BAMM_object` of class `"bammdata"` which is a list with at least 23 elements.

Phylogeny-related elements used to plot a phylogeny with `ape::plot.phylo()`:

- `$edge` Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- `$Nnode` Integer. Number of internal nodes.
- `$tip.label` Character vector. Labels of all tips.
- `$edge.length` Numeric vector. Length of edges/branches.
- `$node.label` Character vector. Labels of all internal nodes. (Present only if present in the initial `phylo`)

BAMM internal elements used for tree exploration:

- `$begin` Numeric vector. Absolute time since root of edge/branch start (rootward).
- `$end` Numeric vector. Absolute time since root of edge/branch end (tipward).
- `$downseq` Integer vector. Order of node visits when using a pre-order tree traversal.
- `$lastvisit` ID of the last node visited when starting from the node in the corresponding position in `$downseq`.

BAMM elements summarizing diversification data:

- `$numberEvents` Integer vector. Number of events/macroevolutionary regimes (k+1) recorded in each posterior configuration. k = number of shifts.

- $eventData List of data.frames. One per posterior sample. Records shift events and macroevolutionary regimes parameters. 1st line = Background root regime.
- $eventVectors List of integer vectors. One per posterior sample. Record regime ID per branch.
- $tipStates List of named integer vectors. One per posterior sample. Record regime ID per tip.
- $tipLambda List of named numeric vectors. One per posterior sample. Record speciation rates per tip.
- $tipMu List of named numeric vectors. One per posterior sample. Record extinction rates per tip.
- $eventBranchSegs List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- $meanTipLambda Named numeric vector. Mean tip speciation rates across all posterior configurations of tips.
- $meanTipMu Named numeric vector. Mean tip extinction rates across all posterior configurations of tips.
- $type Character string. Set the type of data modeled with BAMM. Should be "diversification".

Additional elements providing key information for downstream analyses:

- $expectedNumberOfShifts Integer. The expected number of regime shifts used to set the prior in BAMM.
- $MSP_tree Object of class phylo. List of 4 elements duplicating information from the Phylogeny-related elements above, except MSP_treeedge.length is recording the Marginal Shift Probability of each branch (i.e., the probability of a regime shift to occur along each branch)
- $MAP_indices Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- $MAP_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.
- $MSC_indices Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.
- $MSC_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.

Note on Bayesian Analysis of Macroevolutionary Mixtures (BAMM)

BAMM is a model of diversification for time-calibrated phylogenies that explores complex diversification dynamics by allowing multiple regime shifts across clades without a priori hypotheses on the location of such shifts. It uses reversible jump Markov chain Monte Carlo (rjMCMC) to automatically explore a vast range of models with different speciation and extinction rates, and different number and location of regime shifts.

BAMM is one option among others for modeling diversification on phylogenies. You may wish to explore alternative models such as LSBDS model in RevBayes (Höhna et al., 2016), the MTBD

model (Barido-Sottani et al., 2020), or the ClaDS2 model (Maliet et al., 2019) for your own data. However, you will need Bayesian models that infer regime shifts to be able to perform STRAPP tests (Rabosky & Huang, 2016). Additionally, you need to format the model output such as in `BAMM_object`, so it can be used in a deepSTRAPP workflow.

Author(s)

Maël Doré

References

For BAMM: Rabosky, D. L. (2014). Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. PloS one, 9(2), e89543. doi:10.1371/journal.pone.0089543. Website: http://bamm-project.org/.

For {BAMMtools}: Rabosky, D. L., Grundler, M., Anderson, C., Title, P., Shi, J. J., Brown, J. W., ... & Larson, J. G. (2014). BAMM tools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. Methods in Ecology and Evolution, 5(7), 701-707. doi:10.1111/2041210X.12199

See Also

Initial functions in BAMMtools: `BAMMtools::getEventData()` `BAMMtools::getBestShiftConfiguration()` `BAMMtools::maximumShiftCredibility()`

Associated functions in deepSTRAPP: `prepare_diversification_data()` `plot_BAMM_rates()` `run_deepSTRAPP_for_focal_time()` `run_deepSTRAPP_over_time()`

For a guided tutorial, see this vignette: `vignette("model_diversification_dynamics", package = "deepSTRAPP")`

Examples

```
# The key output from a BAMM is the 'event_data.txt' file
# It can be loaded in R to use as input for deepSTRAPP

# The 'whale_event_data.txt' file used for example here is provided
# within deepSTRAPP in the 'extdata' directory

library(phytools)
data(whale.tree)

BAMM_object <- build_BAMM_object(
   phylo = whale.tree,
   eventdata = system.file("extdata", "whale_event_data.txt", package = "deepSTRAPP"),
   burn_in = 0.25, # Remove 25% as burn-in
   nb_posterior_samples = 1000, # Retain 1000 samples
   expectedNumberOfShifts = 1,
   verbose = TRUE)
str(BAMM_object, 1)
```

`compute_STRAPP_test_for_focal_time` *Compute STRAPP to test for a relationship between diversification rates and trait data*

Description

Carries out the appropriate statistical method to test for a relationship between diversification rates and trait data for a given point in the past (i.e. the `focal_time`). Tests are based on block-permutations: rates data are randomized across tips following blocks defined by the diversification regimes identified on each tip (typically from a BAMM).

Such tests are called STructured RAte Permutations on Phylogenies (STRAPP) as described in Rabosky, D. L., & Huang, H. (2016). A robust semi-parametric test for detecting trait-dependent diversification. Systematic biology, 65(2), 181-193. doi:10.1093/sysbio/syv066.

The function is an extension of the original `BAMMtools::traitDependentBAMM()` function used to carry out STRAPP test on extant time-calibrated phylogenies.

Tests can be carried out on speciation, extinction and net diversification rates.

`deepSTRAPP::compute_STRAPP_test_for_focal_time()` can handle three types of statistical tests depending on the type of trait data provided:

Continuous trait data:

Tests for correlations between trait and rates carried out with `deepSTRAPP::compute_STRAPP_test_for_continuous_dat` The associated test is the Spearman's rank correlation test (See stats::cor.test).

Binary trait data:

For categorical and biogeographic trait data that have only two states (ex: 'Nearctic' vs. 'Neotropics'). Tests for differences in rates between states are carried out with `deepSTRAPP::compute_STRAPP_test_for_binary_` The associated test is the Mann-Whitney-Wilcoxon rank-sum test (See stats::wilcox.test).

Multinominal trait data:

For categorical and biogeographic trait data with more than two states (ex: 'No leg' vs. 'Two legs' vs. 'Four legs'). Tests for differences in rates between states are carried out with `deepSTRAPP::compute_STRAPP_test_fo` The associated test for all states is the Kruskal-Wallis H test (See stats::kruskal.test). If `posthoc_pairwise_tests = TRUE`, post hoc pairwise tests between pairs of states will be carried out too. The associated test for post hoc pairwise tests is Dunn's post hoc pairwise rank-sum test (See dunn.test::dunn.test).

Usage

```
compute_STRAPP_test_for_focal_time(
  BAMM_object,
  trait_data_list,
  rate_type = "net_diversification",
  uncertainty_strategy = "paired",
  trait_maps_vs_BAMM_samples_list = NULL,
  seed = NULL,
  nb_permutations = NULL,
```

```
  alpha = 0.05,
  two_tailed = TRUE,
  one_tailed_hypothesis = NULL,
  posthoc_pairwise_tests = FALSE,
  p.adjust_method = "none",
  trait_data_type_for_stats = NULL,
  return_perm_data = FALSE,
  nthreads = 1,
  print_hypothesis = TRUE
)
```

Arguments

BAMM_object	Object of class "bammdata", typically generated with update_rates_and_regimes_for_focal_time(), that contains a phylogenetic tree and associated diversification rates across selected posterior samples updated to a specific time in the past (i.e. the focal_time).
trait_data_list	List obtained from extract_most_likely_trait_values_for_focal_time() or extract_all_trait_values_for_focal_time() that contains at least a $trait_data element, a $focal_time element, and a $trait_data_type. $trait_data is a named vector with the trait data found on the phylogeny at focal_time. $focal_time informs on the time in the past at which the trait and rates data will be tested. $trait_data_type informs on the type of trait data: 'continuous', 'categorical', or 'biogeographic'.
rate_type	A character string specifying the type of diversification rates to use. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).
uncertainty_strategy	Character string. To select the strategy used to account for uncertainty in estimates. • "rates_only": Only accounts for diversification-rate uncertainty across BAMM posterior samples. Uses ML estimates for continuous traits and the most frequent state/range observed across stochastic maps for categorical and biogeographic data. • "paired": Default option. Accounts for both diversification-rate and ancestral trait/range reconstruction uncertainty by pairing BAMM posterior samples with stochastic maps. When the number of BAMM samples and stochastic maps differ, random pairing with replacement from the smaller set is used so that all posterior samples and stochastic maps contribute to the analysis. • "full": Exhaustive option that accounts for trait/range- and rate- uncertainty by crossing all BAMM posterior samples with all stochastic maps. Accounts for both diversification-rate and ancestral reconstruction uncertainty by evaluating every combination of BAMM posterior sample and stochastic map. WARNING: This exhaustive approach can substantially increase computation time and memory requirements and is therefore recommended only for moderate-sized analyses.

`trait_maps_vs_BAMM_samples_list`
: (Optional) Data.frame of two variables manually providing the names to associate stochastic maps (`$trait_map_ID`) with BAMM samples (`$BAMM_posterior_sample_ID`). This is typically used to ensure the same stochastic maps and BAMM samples are used to test across multiple time-steps. Values are the names of the objects such as "Map_X" and "BAMM_X". Default = `NULL`. * For uncertainty_strategy == 'rates_only', the `$trait_map_ID` must be "Map_ML" as only the ML estimates of trait values/states/ranges are used. * For uncertainty_strategy == 'paired', each pair of stochastic map and BAMM sample will be used once. * For uncertainty_strategy == 'full', all stochastic maps will be matched with all BAMM samples. Those may partly differ from the actual maps and BAMM samples used for the test, as recorded in `$perm_data_df`, because invalid maps with not enough states/ranges are discarded.

`seed`
: Integer. Set the seed to ensure reproducibility. Default is `NULL` (a random seed is used).

`nb_permutations`
: Integer. To select the number of random permutations to perform during the tests. If NULL (default), all BAMM posterior samples will be used once.

`alpha`
: Numeric. Significance level to use to compute the `estimate` corresponding to the values of the test statistic used to assess significance of the test. This does NOT affect p-values. Default is `0.05`.

`two_tailed`
: Logical. To define the type of tests. If `TRUE` (default), tests for correlations/differences in rates will be carried out with a null hypothesis that rates are not correlated with trait values (continuous data) or equal between trait states (categorical and biogeographic data). If `FALSE`, one-tailed tests are carried out.

  - For continuous data, it involves defining a `one_tailed_hypothesis` testing for either a "positive" or "negative" correlation under the alternative hypothesis.
  - For binary data (two states), it involves defining a `one_tailed_hypothesis` indicating which states have higher rates under the alternative hypothesis.
  - For multinomial data (more than two states), it defines the type of post hoc pairwise tests to carry out between pairs of states. If `posthoc_pairwise_tests = TRUE`, all two-tailed (if `two_tailed = TRUE`) or one-tailed (if `two_tailed = FALSE`) tests are automatically carried out.

`one_tailed_hypothesis`
: A character string specifying the alternative hypothesis in the one-tailed test. For continuous data, it is either "negative" or "positive" correlation. For binary data, it lists the trait states with states ordered in increasing rates under the alternative hypothesis, separated by a greater-than such as c('A > B').

`posthoc_pairwise_tests`
: Logical. Only for multinomial data (with more than two states). If `TRUE`, all possible post hoc pairwise (Dunn) tests will be computed across all pairs of states. This is a way to detect which pairs of states have significant differences in rates if the overall test (Kruskal-Wallis) is significant. Default is `FALSE`.

`p.adjust_method`
: A character string. Only for multinomial data (with more than two states). It specifies the type of correction to apply to the p-values in the post hoc pair-

wise tests to account for multiple comparisons. See `stats::p.adjust()` for the available methods. Default is `none`.

`trait_data_type_for_stats`
: (Optional) Character string. One of `"continuous"`, `"binary"`, or `"multinomial"`. Imposes the statistical method to use, instead of deriving it from the type of traits and the number of states/ranges observed at this `focal_time`. This is set automatically by `run_deepSTRAPP_over_time()`, which fixes the method once for the whole run from the complete trait mapping, so that the same test is used at every time step and the p-values along the trajectory remain comparable. Default is `NULL`, in which case the method is selected from the number of states/ranges in `trait_data_list$trait_data` observed at this `focal_time`.

`return_perm_data`
: Logical. Whether to return the stats data computed from the posterior samples for observed and permuted data in the output. This is needed to plot the histogram of the null distribution used to assess significance of the test with `plot_histogram_STRAPP_test_for_focal_time()`. Default is `FALSE`.

`nthreads`
: Integer. Number of threads to use for parallel computing of the tests across the permutations. The R package `parallel` must be loaded for `nthreads > 1`. Default is `1`.

`print_hypothesis`
: Logical. Whether to print information on what test is carried out, detailing the null and alternative hypotheses, what significance level is used to reject or not the null hypothesis, and how uncertainty in trait estimates is handled. Default is `TRUE`.

Details

This set of functions carries out the STructured RAte Permutations on Phylogenies (STRAPP) test as defined in Rabosky, D. L., & Huang, H. (2016). A robust semi-parametric test for detecting trait-dependent diversification. Systematic biology, 65(2), 181-193.

It is an extension of the original `BAMMtools::traitDependentBAMM()` function used to carry out STRAPP test on extant time-calibrated phylogenies, but allowing here to test for differences/correlations at any point in the past (i.e. the `focal_time`).

It takes an object of class `"bammdata"` (`BAMM_object`) that was updated such that its diversification rates (`$tipLambda` and `$tipMu`) and regimes (`$tipStates`) are reflecting values observed at a specific time in the past (i.e. the `$focal_time`). Similarly, it takes a list (`trait_data_list`) that provides `$trait_data` as observed on branches at the same `focal_time` as the diversification rates and regimes.

A STRAPP test is carried out by drawing a random set of posterior samples from the `BAMM_object`, then randomly permuting rates across blocks of tips defined by the macroevolutionary regimes. Test statistics are then computed across the initial observed data and the permuted data for each sample. In a two-tailed test, the p-value is the proportion of posterior samples in which the test stats is more extreme in the permuted than in the observed data. In a one-tailed test, the p-value is the proportion of posterior samples in which the test stats is higher in the permuted than in the observed data.

———- Major changes compared to `BAMMtools::traitDependentBAMM()` ———-

- Allow to account for uncertainty in trait estimates by pairing/mapping multiple trait data extracted from stochastic maps across the BAMM samples. See the `uncertainty_strategy` argument for details.
- Add post hoc pairwise tests (Dunn test) for multinomial data. Use `posthoc_pairwise_tests = TRUE`.
- Provide outputs tailored for histogram plots `plot_histogram_STRAPP_test_for_focal_time()` and p-value time-series plots `plot_STRAPP_pvalues_over_time()`.
- Add prints detailing what test is carried out, what are the null and alternative hypotheses, and what significance level is used to reject or not the null hypothesis. (Enabled with `print_hypothesis = TRUE`).
- Split the function in multiple sub-functions according to the type of data (`$trait_data_type`).
- Prevent using Pearson's correlation tests and applying log-transformation for continuous data. The rationale is that there is no reason to assume that tip rates are distributed normally or log-normally. Thus, a Spearman's rank correlation test is favored.

Value

The function returns a list with at least eleven elements.

Summary elements for the main test:

- `$estimate` Named numeric. Value of the test statistic used to assess significance of the test according to the significance level provided (`alpha`). The test is significant if `$estimate` is higher than zero.
- `$stats_median` Numeric. Median value of the distribution of test statistics.
- `$nb_test_stats` Integer. Number of test stats in the distribution.
- `$p-value` Numeric. P-value of the test. The test is considered significant if `$p-value` is lower than `alpha`.
- `$method` Character string. The statistical method used to carry out the test.
- `$rate_type` Character string. The type of diversification rates tested. One of 'speciation', 'extinction' or 'net_diversification'.
- `$trait_data_type` Character string. The type of trait data as found in 'trait_data_list$trait_data_type'. One of 'continuous', 'categorical', or 'biogeographic'.
- `$trait_data_type_for_stats` Character string. The type of trait data used to select statistical method. One of 'continuous', 'binary', or 'multinomial'.
- `$focal_time` The time in the past at which the trait and rates data were tested.
- `$uncertainty_strategy` Character string. The strategy used to account for uncertainty in estimates.
- `$trait_maps_vs_BAMM_samples_list` List of two elements recording the stochastic maps (`$trait_map_ID`) and BAMM samples (`$BAMM_posterior_sample_ID`) chosen for testing. Those may partly differ from the actual maps and BAMM samples used for the test as recorded in `$perm_data_df` because invalid maps with not enough states/ranges are discarded.
- `$states_observed` (Only for categorical and biogeographic data) Character string vector of the states/ranges actually found in `trait_data_list$trait_data` at this `focal_time`. This can be a subset of the states/ranges described in the complete trait mapping, since a state/range may be absent from the deeper time steps.

- $nb_states_observed (Only for categorical and biogeographic data) Integer. Number of states/ranges actually found in trait_data_list$trait_data at this focal_time.

If using continuous or binary data:

- $two-tailed Logical. Record the type of test used: two-tailed if TRUE, one-tailed if FALSE. If one_tailed_hypothesis is provided (only for continuous and binary trait data):
- $one_tailed_hypothesis Character string. Record of the alternative hypothesis used for the one-tailed tests.

If posthoc_pairwise_tests = TRUE (only for multinomial trait data):

- $posthoc_pairwise_tests List of at least 3 sub-elements:
  - $summary_df Data.frame of six variables providing the summary results of post hoc pairwise tests
  - $method Character string. The statistical method used to carry out the test. Here, "Dunn".
  - $two-tailed Logical. Record the type of post hoc pairwise tests used: two-tailed if TRUE, one-tailed if FALSE.

If return_perm_data = TRUE, the stats data computed from the posterior samples for observed and permuted data are provided. This is needed to plot the histogram of the null distribution used to assess significance of the test with plot_histogram_STRAPP_test_for_focal_time().

- $perm_data_df A data.frame with four variables summarizing the data generated during the STRAPP test:
  - $trait_map_ID Integer. ID of the stochastic map from which trait data were extracted and used for the STRAPP test.
  - $BAMM_posterior_sample_ID Integer. ID of the posterior samples randomly drawn and used for the STRAPP test.
  - $*_obs Numeric. Test stats computed from the observed data in the posterior samples. Name depends on the test used.
  - $*_perm Numeric. Test stats computed from the permuted data in the posterior samples. Name depends on the test used.
  - $delta_* OR $abs_delta_* Numeric. Test stats computed for the STRAPP test comparing observed stats and permuted stats. Name depends on the test used and the type of tests (two-tailed compare absolute values; one-tailed compare raw values). Combined with posthoc_pairwise_tests = TRUE, the stats data are also provided for the post hoc pairwise tests:
- $posthoc_pairwise_tests$perm_data_array A 3D array containing stats data for all post hoc pairwise tests in a similar format to $perm_data_df.

If no STRAPP test was performed in the case of categorical/biogeographic data with a single state/range at focal_time, only the $trait_data_type, $trait_data_type_for_stats = "none", $focal_time, $trait_maps_vs_BAMM_samples_list, $states_observed, and $nb_states_observed are returned.

Author(s)

Maël Doré

References

For STRAPP: Rabosky, D. L., & Huang, H. (2016). A robust semi-parametric test for detecting trait-dependent diversification. Systematic biology, 65(2), 181-193. doi:10.1093/sysbio/syv066.

For STRAPP in deep times: Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B., (2025), Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity, Nature Communications. doi:10.1038/s41467025637093

See Also

Associated functions in deepSTRAPP: `extract_most_likely_trait_values_for_focal_time()` `update_rates_and_regimes_for_focal_time()`

Original function in BAMMtools: `BAMMtools::traitDependentBAMM()`

Statistical tests: `stats::cor.test()` `stats::wilcox.test()` `stats::kruskal.test()` `dunn.test::dunn.test()`

For a guided tutorial, see this vignette: `vignette("explore_STRAPP_test_types", package = "deepSTRAPP")`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ------ Prepare data ------ #

 ## Load the BAMM_object summarizing 1000 posterior samples of BAMM with diversification rates
 # for ponerine ants extracted for 10My ago.
 data(Ponerinae_BAMM_object_10My, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 # Plot the associated phylogeny with mapped rates
 plot_BAMM_rates(Ponerinae_BAMM_object_10My)

 ## Load the object containing head width trait data for ponerine ants extracted for 10My ago.
 data(Ponerinae_trait_cont_tip_data_10My, package = "deepSTRAPP")

 # Plot the associated contMap (continuous trait stochastic map)
 plot_contMap(Ponerinae_trait_cont_tip_data_10My$contMap)

 # Check that objects are ordered in the same fashion
 identical(names(Ponerinae_BAMM_object_10My$tipStates[[1]]),
           names(Ponerinae_trait_cont_tip_data_10My$trait_data))

 # Save continuous data
 trait_data_continuous <- Ponerinae_trait_cont_tip_data_10My

 ## Transform trait data into binary and multinomial data

 # Binarize data into two states
 trait_data_binary <- trait_data_continuous
```

```
trait_data_binary$trait_data[trait_data_continuous$trait_data < 0.5] <- "state_A"
trait_data_binary$trait_data[trait_data_continuous$trait_data >= 0.5] <- "state_B"
trait_data_binary$trait_data_type <- "categorical"

table(trait_data_binary$trait_data)

# Categorize data into three states
trait_data_multinomial <- trait_data_continuous
trait_data_multinomial$trait_data[trait_data_continuous$trait_data < 0.6] <- "state_B"
trait_data_multinomial$trait_data[trait_data_continuous$trait_data < 0.4] <- "state_A"
trait_data_multinomial$trait_data[trait_data_continuous$trait_data >= 0.6] <- "state_C"
trait_data_multinomial$trait_data_type <- "categorical"

table(trait_data_multinomial$trait_data)

## Duplicate trait data as if extracted from multiple stochastic maps

trait_data_continuous_multimaps <- Ponerinae_trait_cont_tip_data_10My
trait_data_initial <- trait_data_continuous_multimaps$trait_data

trait_data_multimaps <- list()
for (i in 1:10)
{
  # Add a bit of randomness across all stochastic maps data
  trait_data_multimaps[[i]] <- trait_data_initial +
      rnorm(n = length(trait_data_initial), mean = 0, sd = sd(trait_data_initial) / 10)
}
names(trait_data_multimaps) <- paste0("Map_", 1:10)
trait_data_continuous_multimaps$trait_data <- trait_data_multimaps


 # (May take several minutes to run)
# ------ Compute STRAPP test for continuous data ------ #

plot(x = trait_data_continuous$trait_data, y = Ponerinae_BAMM_object_10My$tipLambda[[1]])

# Compute STRAPP test under the alternative hypothesis of a "negative" correlation
# between "net_diversification" rates and trait data
STRAPP_results <- compute_STRAPP_test_for_focal_time(
   BAMM_object = Ponerinae_BAMM_object_10My,
   trait_data_list = trait_data_continuous,
   uncertainty_strategy = "rates_only",
   two_tailed = FALSE,
   one_tailed_hypothesis = "negative",
   return_perm_data = TRUE)
str(STRAPP_results, max.level = 2)
# Data from the posterior samples is available in STRAPP_results$perm_data_df
head(STRAPP_results$perm_data_df)

# ------ Compute STRAPP test for binary data ------ #

# Compute STRAPP test under the alternative hypothesis that "state_A" is associated
# with higher "net_diversification" that "state_B"
```

```
STRAPP_results <- compute_STRAPP_test_for_focal_time(
   BAMM_object = Ponerinae_BAMM_object_10My,
   trait_data_list = trait_data_binary,
   uncertainty_strategy = "rates_only",
   two_tailed = FALSE,
   one_tailed_hypothesis = c("state_A > state_B"))
str(STRAPP_results, max.level = 1)

# Compute STRAPP test under the alternative hypothesis that "state_B" is associated
# with higher "net_diversification" that "state_A"
STRAPP_results <- compute_STRAPP_test_for_focal_time(BAMM_object = Ponerinae_BAMM_object_10My,
   trait_data_list = trait_data_binary,
   uncertainty_strategy = "rates_only",
   two_tailed = FALSE,
   one_tailed_hypothesis = c("state_B > state_A"))
str(STRAPP_results, max.level = 1)

# ------ Compute STRAPP test for multinomial data ------ #

# Compute STRAPP test between all three states, and compute post hoc tests
# for differences in rates between all possible pairs of states
# with a p-value adjusted for multiple comparison using Bonferroni's correction
STRAPP_results <- compute_STRAPP_test_for_focal_time(
   BAMM_object = Ponerinae_BAMM_object_10My,
   trait_data_list = trait_data_multinomial,
   uncertainty_strategy = "rates_only",
   posthoc_pairwise_tests = TRUE,
   two_tailed = TRUE,
   p.adjust_method = "bonferroni",
   return_perm_data = TRUE)
str(STRAPP_results, max.level = 2)
# All post hoc pairwise test summaries are available in $summary_df
STRAPP_results$posthoc_pairwise_tests$summary_df

# ------ Compute STRAPP test with the 'paired' strategy ------ #

# Account for uncertainty in trait estimates
# by pairing stochastic maps with BAMM posterior samples

# Compute STRAPP test under the alternative hypothesis of a "negative" correlation
# between "net_diversification" rates and trait data
STRAPP_results <- compute_STRAPP_test_for_focal_time(
  BAMM_object = Ponerinae_BAMM_object_10My,
  trait_data_list = trait_data_continuous_multimaps,
  uncertainty_strategy = "paired",
  two_tailed = FALSE,
  one_tailed_hypothesis = "negative",
  return_perm_data = TRUE)
str(STRAPP_results, max.level = 2)
# Data from the paired stochastic maps and BAMM posterior samples
# is available in STRAPP_results$perm_data_df
head(STRAPP_results$perm_data_df)
# Tests were performed across 1000 iterations since the stochastic maps
```

```
# have been paired with the 1000 BAMM samples
nrow(STRAPP_results$perm_data_df)
# Each of the 10 maps was paired ca. 100 times with a unique BAMM sample
table(STRAPP_results$perm_data_df$trait_map_ID)

# ------ Compute STRAPP test with the 'full' strategy ------ #

# Account for uncertainty in trait estimates
# by crossing stochastic maps with BAMM posterior samples

# Compute STRAPP test under the alternative hypothesis of a "negative" correlation
# between "net_diversification" rates and trait data
STRAPP_results <- compute_STRAPP_test_for_focal_time(
  BAMM_object = Ponerinae_BAMM_object_10My,
  trait_data_list = trait_data_continuous_multimaps,
  uncertainty_strategy = "full",
  two_tailed = FALSE,
  one_tailed_hypothesis = "negative",
  return_perm_data = TRUE)
str(STRAPP_results, max.level = 2)
# Data from the combined stochastic maps X BAMM posterior samples
# is available in STRAPP_results$perm_data_df
head(STRAPP_results$perm_data_df)
# Tests were performed across 10000 iterations since the 10 stochastic maps
# have been combined with the 1000 BAMM samples
nrow(STRAPP_results$perm_data_df)
# Each of the 10 maps was paired with all 1000 BAMM samples
table(STRAPP_results$perm_data_df$trait_map_ID)

}
```

`convert_BSM_to_simmap` *Convert Biogeographic Stochastic Map (BSM) to phytools SIMMAP stochastic map (SM) format*

Description

These functions convert a Biogeographic Stochastic Map (BSM) output from BioGeoBEARS into a `simmap` object from R package `{phytools}` (See `phytools::make.simmap()`).

They require a model fit with `BioGeoBEARS::bears_optim_run()` and the output of a Biogeographic Stochastic Mapping performed with `BioGeoBEARS::runBSM()` to produce `simmap` objects as phylogenies with the associated mapping of range evolution along branches across simulations.

- `convert_BSM_to_simmap()`: Produce one `simmap` for the required simulation (index of the simulation provided with `sim_index`).
- `convert_BSMs_to_simmaps()`: Produce all `simmap` objects for all simulations stored in a unique `multiSimmap` object.

Initial functions in R package BioGeoBEARS by Nicholas J. Matzke:

- `BioGeoBEARS::BSM_to_phytools_SM()`
- `BioGeoBEARS::BSMs_to_phytools_SMs()`

Usage

```
convert_BSM_to_simmap(model_fit, phylo, BSM_output, sim_index)

convert_BSMs_to_simmaps(model_fit, phylo, BSM_output)
```

Arguments

`model_fit`	A BioGeoBEARS results object, produced by ML inference via `BioGeoBEARS::bears_optim_run()`.
`phylo`	Time-calibrated phylogeny used in the BioGeoBEARS analyses to produce the historical biogeographic inference and run the Biogeographic Stochastic Mapping. Object of class `"phylo"` as defined in `{ape}`.
`BSM_output`	A list with two objects, a cladogenetic events table and an anagenetic events table, as the result of Biogeographic Stochastic Mapping conducted with `BioGeoBEARS::runBSM()`.
`sim_index`	Integer. Index of the biogeographic simulation targeted to produce the `simmap` with `convert_BSM_to_simmap()`.

Details

- `BioGeoBEARS::BSM_to_phytools_SM()`
- `BioGeoBEARS::BSMs_to_phytools_SMs()`

These functions are slight adaptations of original functions from the R Package `BioGeoBEARS` by N. Matzke.

Initial functions: `BioGeoBEARS::BSM_to_phytools_SM()` `BioGeoBEARS::BSMs_to_phytools_SMs()`

Changes:

- Solves issue with differences in ranges allowed across time-strata.
- Requires directly the output of `BioGeoBEARS::runBSM()` instead of separated cladogenetic and anagenetic event tables.
- Resolve paths to local tree and tip range files.
- Restore the S4 class 'BioGeoBEARS_model' if needed.
- Update the documentation.

Value

The `convert_BSM_to_simmap()` function returns a list with two elements:

- `$simmap` A unique `simmap` for a given biogeographic simulation as an object of classes `c("simmap", "phylo")`. This is a modified `{ape}` tree with additional elements to report range mapping, model parameters and likelihood.
  - `$maps` A list of named numeric vectors. Provides the mapping of ranges along each remaining edge. Names are the ranges. Values are residence times in each state across segments

  - `$mapped.edge` A numeric matrix. Provides the evolutionary time spent across ranges (columns) along the edges (rows). row.names() are the node ID at the rootward and tipward ends of each edge.
  - `$Q` Numerical matrix. The transition rates across ranges calculated from the ML parameter estimates of the model.
  - `$logL` Numeric. The log-likelihood of the data under the ML model.
- `$residence_times` Data.frame with two rows. Summarizes the residence time spent in each range along all branches, in (raw) evolutionary time (i.e., branch lengths), and in percentage (perc).

The `convert_BSMs_to_simmaps()` function loops around the `convert_BSM_to_simmap()` function to aggregate all `simmaps` from all biogeographic simulations in a unique list of classes `c("multiSimmap", "multiPhylo")`.

- Each element is the `$simmap` of a biogeographic simulation obtained with `convert_BSM_to_simmap()`.
- `$residence_times` summary data.frames are not preserved.

Notes on BioGeoBEARS

The R package `BioGeoBEARS` is needed for this function to work with biogeographic data. Please install it manually from: https://github.com/nmatzke/BioGeoBEARS.

Notes on using the resulting simmap object in phytools (adapted from Nicholas J. Matzke)

The phytools functions, like `phytools::countSimmap()`, will only count the anagenetic events (range transitions occurring along branches) as they were written assuming purely anagenetic models.

It remains possible to extract cladogenetic events (range transitions occurring at speciation) by comparing the last-state-below-a-node with the descendant-pairs-above-a-node. However, it is recommended to use the built-in functions from BioGeoBEARS to summarize the biogeographic history based on the tables of cladogenetic and anagenetic events obtained from `BioGeoBEARS::runBSM()`. `simmap` objects should primarily be considered as a tool for visualization.

Associated functions in R package `BioGeoBEARS`:

- `BioGeoBEARS::simulate_source_areas_ana_clado()`: To select randomly a unique area source for transition from a multi-area state to a single area.
- `BioGeoBEARS::get_dmat_times_from_res()`: To generate matrices of range expansion from source area to destination area.
- `BioGeoBEARS::count_ana_clado_events()`: To count the number and type of events from BSM tables.
- `BioGeoBEARS::hist_event_counts()`: To plot histograms of event counts across BSM tables.

Please note carefully that area-to-area dispersal events are not identical with the state transitions. For example, a state can be a geographic range with multiple areas, but under the logic of DEC-type models, a range-expansion event like ABC->ABCD actually means that a dispersal happened from some specific area (A, B, or C) to the new area. BSMs track this area-to-area sourcing in their cladogenetic and anagenetic event tables, at least if `BioGeoBEARS::simulate_source_areas_ana_clado()` has been run on the output of `BioGeoBEARS::runBSM()`.

Author(s)

Nicholas J. Matzke. Contact: <matzke@berkeley.edu>

Changes by Maël Doré (see Details)

References

For BioGeoBEARS: Matzke, Nicholas J. (2018). BioGeoBEARS: BioGeography with Bayesian (and likelihood) Evolutionary Analysis with R Scripts. version 1.1.1, published on GitHub on November 6, 2018. doi:10.5281/zenodo.1478250. Website: http://phylo.wikidot.com/biogeobears.

See Also

phytools::countSimmap() phytools::make.simmap() BioGeoBEARS::simulate_source_areas_ana_clado() BioGeoBEARS::get_dmat_times_from_res() BioGeoBEARS::count_ana_clado_events() BioGeoBEARS::hist_event_

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## Run only if you have R package 'BioGeoBEARS' installed.
 # Please install it manually from: https://github.com/nmatzke/BioGeoBEARS")

 ## Load phylogeny and tip data
 library(phytools)
 data(eel.tree)

  # (May take several minutes to run)
 ## Load directly output of prepare_trait_data() run on biogeographic data
 data(eel_biogeo_data, package = "deepSTRAPP")

 ## Convert BSM output into a unique simmap, including residence times
 simmap_1 <- convert_BSM_to_simmap(model_fit = eel_biogeo_data$best_model_fit,
                                   phylo = eel.tree,
                                   BSM_output = eel_biogeo_data$BSM_output,
                                   sim_index = 1)
 # Explore output
 str(simmap_1, max.level = 1)
 # Print residence times in each range
 simmap_1$residence_times
 # Plot simmap
 plot(simmap_1$simmap)

 ## Convert BSM output into all simmaps in a multiSimmap/multiPhylo object
 all_simmaps <- convert_BSMs_to_simmaps(model_fit = eel_biogeo_data$best_model_fit,
                                        phylo = eel.tree,
                                        BSM_output = eel_biogeo_data$BSM_output)
 # Explore output
 str(all_simmaps, max.level = 1)
 # Plot simmap n°1
 plot(all_simmaps[[1]])
}
```

`convert_contsimmap_to_contMaps` *Convert a contsimmap object into a list of contMaps*

Description

Convert a contsimmap object generated with `contsimmap::make.contsimmap()` into a list of `contMaps` as produced by the `phytools::contMap()` function.

Usage

```
convert_contsimmap_to_contMaps(contsimmap, verbose = FALSE)
```

Arguments

`contsimmap`	List of class `"contsimmap"`, typically generated with `contsimmap::make.contsimmap()`, that summarizes a set of continuous stochastic maps. Each map represents a simulation of the evolution of a continuous trait compatible with the associated evolutionary model fit on observed trait data.
`verbose`	Logical. Whether to display conversion progress every 10 maps.

Details

A contsimmap object is typically produced by the `contsimmap::make.contsimmap()` function. It stores results of a continuous stochastic mapping simulation that represent independent evolutionary history of a continuous trait mapped along a time-calibrated phylogeny, and compatible with the associated evolutionary model fit on observed trait data. The method is described in Martin & Weber, 2026.

A typical `"contsimmap"` object stores simulated trait values across maps in a 3D array, recording values for all time-points used to represent the continuous evolution. While `"contsimmap"` objects can record evolution of multivariate traits, deepSTRAPP currently works only with simulation of univariate trait evolution. See the `contsimmap::make.contsimmap()` help section for a detailed description of the structure of the object.

A `"contMap"` object typically produced by the `phytools::contMap()` function also records the evolution of a continuous trait alongside branches of a time-calibrated phylogeny. Trait values are stored in `$maps` as list of named vectors recording trait values (as names scaled between 0 and 1000) and length of the associated branch segment between two time-points (as values). Therefore, a unique `"contsimmap"` object is converted into a list of `"contMaps"`, with each item being a `"contMap"` that represents a unique evolutionary history simulation conditioned on the observed trait data and model fit.

Value

Returns a list of "contMap" objects. Each "contMap" represents a unique evolutionary history simulation conditioned on the observed trait data and model fit (i.e., a continuous stochastic map).

A "contMap" typically contains:

- $tree A list of classes "simmap" and "phylo". The mapped phylogeny including:
  - $maps A list of named numeric vectors. Provides the mapping of trait values along each edge (scaled between 0 and 1000).
  - $mapped.edge A numeric matrix. Provides the evolutionary time spent across trait values (columns) along the edges (rows).
- $cols A named vector mapping values to colors used to display trait evolution along the branches.
- $lims A numeric vector. Minimum and maximum trait values recorded during the simulation.

Author(s)

Maël Doré

References

For continuous stochastic mapping: Martin, B. S., & Weber, M. G. (2026). Stochastic character mapping of continuous traits on phylogenies. Systematic Biology, syag031. doi:10.1093/sysbio/syag031.

See Also

contsimmap::make.contsimmap() phytools::contMap() prepare_trait_data()

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## The R package 'contsimmap' is needed for this example to work.
 # Please install it manually from: https://github.com/bstaggmartin/contsimmap.

 # ----- Prepare data ----- #

 # Load eel phylogeny and tip data from the R package phytools
 # Source: Collar et al., 2014; DOI: 10.1038/ncomms6505

 data("eel.tree", package = "phytools")
 data("eel.data", package = "phytools")

 # Extract body size
 eel_tip_data <- stats::setNames(eel.data$Max_TL_cm,
                                 rownames(eel.data))

 # ----- Infer evolutionary rate ----- #

 # Use phytools::anc.ML() to extract the evolutionary rate under a BM
```

```
BM_fit <- phytools::anc.ML(tree = eel.tree,
                           x = eel_tip_data,
                           model = "BM")
# Extract evolutionary rate = sigma²
sigma2 <- BM_fit$sig2

# ----- Run continuous stochastic mapping ----- #

 # (May take several seconds to run)
eel_contsimmap <- contsimmap::make.contsimmap(
   tree = eel.tree,
   trait.data = eel_tip_data,
   Xsig2 = list(sigma2), # Evolutionary rate
   nsim = 100, res = 100,
   verbose = TRUE)

dim(eel_contsimmap) # 121 edges, 1 trait, 100 simulations

# ----- Convert to a list of contMaps ----- #

eel_contMaps <- convert_contsimmap_to_contMaps(
   contsimmap = eel_contsimmap, verbose = TRUE)

# ----- Explore contMaps ----- #

# Plot contMap from simulation n°1
plot_contMap(eel_contMaps[[1]])
# Plot contMap from simulation n°50
plot_contMap(eel_contMaps[[50]])

}
```

`convert_simmaps_to_densityMaps` *Convert simmaps into densityMaps suitable for deepSTRAPP*

Description

Convert `simmaps` representing independent simulations of evolutionary history into a `densityMaps` object suitable to use as input for a deepSTRAPP run.

`simmaps` are mapping discrete character/geographic evolutionary history as transitions in character states/geographic ranges across nodes and branches of a phylogeny.

`densityMaps` are recording the posterior probability/frequencies of being in a given state/range along branches. In deepSTRAPP format, `densityMaps` is a list of objects of class `"densityMap"` where each object corresponds to the frequency of presence/absence of a state/range.

Usage

```
convert_simmaps_to_densityMaps(
  simmaps,
  colors_per_levels = NULL,
  tol = 1e-05,
  verbose = TRUE
)
```

Arguments

simmaps	List of objects of class "simmap", typically generated with prepare_trait_data() or phytools::make.simmap(), that represent discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches.
colors_per_levels	Named character string. To set the colors to use to map each state/range posterior probabilities. Names = states/ranges; values = colors. If NULL (default), the rainbow() color scale will be used.
tol	Positive numeric. To set the tolerance used to match node ages and time steps (i.e., consider them equal). Default = 1e-5.
verbose	Logical. Whether to display progress every 100 edges. Default = TRUE.

Details

The function is a wrapper of the original phytools::densityMap() by Liam Revell. However, it does not produce a single densityMap for binary states, but rather can handle simmaps mapping any number of states/ranges, and produce a densityMaps list that summarizes the frequency of presence/absence of each state/range in subsequent densityMap objects.

Both simmaps and densityMaps objects can be used as inputs for a deepSTRAPP run with run_deepSTRAPP_for_focal_tim or run_deepSTRAPP_over_time().

simmaps retain the identity of each simulated history, allowing deepSTRAPP to keep track of which simulation generated each set of trait values (i.e., states/ranges). However, retaining all simulated histories can require substantial RAM, particularly for large phylogenies or many simulations.

Alternatively, densityMaps summarize the frequency of states/ranges across simulations. They require substantially less RAM and can be used to visualize the overall uncertainty in trait evolution with plot_densityMaps_overlay().

The trade-off is that densityMaps discard the identity of individual simulations and therefore cannot be used to track which simulated history generated a given set of trait values.

Value

The function returns a densityMaps as a list of objects of class "densityMap", where each object is mapping the frequency of presence/absence of a state/range.

The number of objects depends on the number of states/ranges observed across the simmaps provided for conversion. Each densityMap is named as "Density_map_X" with X being each of the state/range recorded across the simmaps.

Each densityMap is a list of three elements:

- `$tree` List of classes `"simmap"` and `"phylo"` that contains the phylogeny in ape format and the mapping of states/ranges frequencies along branches in `$tree$maps`.
- `$cols` Named character strings. Colors mapped to the 0 to 1000 scale used to record frequencies in `$tree$maps`.
- `$states` Character string with two values. First entry is the absence of the state/range recorded as "Not X". Second entry is the presence of the state/range recorded as "X", X being the state/range name.

Author(s)

Maël Doré

See Also

`phytools::densityMap()` `plot_densityMaps_overlay()` `prepare_trait_data()`

Examples

```
## Load data

# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
## Produce densityMaps using stochastic character mapping based on an ER Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
    tip_data = Ponerinae_cat_3lvl_tip_data,
    phylo = Ponerinae_tree_old_calib,
    trait_data_type = "categorical",
    colors_per_levels = colors_per_states,
    evolutionary_models = "ER", # Use ER model
    run_stochastic_maps = TRUE,
    nb_simulations = 100, # Reduce number of simulations to save time
    seed = 1234, # Set seed for reproducibility
    return_simmaps = TRUE, # Return simmaps in the output
    plot_map = FALSE)

# Note that densityMaps are already produced by the [deepSTRAPP::prepare_trait_data()] function,
# but for the sake of example, we can convert the simmaps stored in the output into densityMaps,
# using [deepSTRAPP::convert_simmaps_to_densityMaps()].
```

```
## Convert simmaps to densityMaps as input format for deepSTRAPP
Ponerinae_densityMaps <- convert_simmaps_to_densityMaps(
   simmaps = Ponerinae_cat_3lvl_data_old_calib$simmaps)

# Plot densityMaps one by one
plot(Ponerinae_densityMaps[[1]]) # densityMap for state n°1 ("arboreal")
plot(Ponerinae_densityMaps[[2]]) # densityMap for state n°1 ("subterranean")
plot(Ponerinae_densityMaps[[3]]) # densityMap for state n°1 ("terricolous")

# Plot overlay of all densityMaps
plot_densityMaps_overlay(densityMaps = Ponerinae_densityMaps)
```

`cut_contMaps_for_focal_time` *Cut the phylogenies and continuous trait mappings of a list of contMaps for a given focal time in the past*

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise, remove continuous trait mapping for the cut off branches by updating the `$tree$maps` and `$tree$mapped.edge` elements.

Usage

```
cut_contMaps_for_focal_time(contMaps, focal_time, keep_tip_labels = TRUE)
```

Arguments

`contMaps`	List of objects of class `"contMap"`, typically generated with `phytools::contMap()`, that contains a phylogenetic tree and associated continuous trait mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

The phylogenetic trees are cut for a specific time in the past (i.e. the `focal_time`).

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The continuous trait mappings are updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns an updated list of objects as cut contMaps of class `"contMap"`.

Each contMap object contains three elements:

- `$tree`. A list of classes `"simmap"` and `"phylo"`. This function updates and adds multiple useful sub-elements to the `$tree` element.
  - `$maps` An updated list of named numeric vectors. Provides the mapping of trait values along each remaining edge.
  - `$mapped.edge` An updated matrix. Provides the evolutionary time spent across trait values (columns) along the remaining edges (rows).
  - `$root_age` Integer. Stores the age of the root of the tree.
  - `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
  - `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
  - `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
  - `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.
- `$cols` A named character vector mapping trait values with colors to display on the plot.
- `$lims` A numeric vector of length 2 storing the initial min and max values of the trait (trait values are scaled between 0 and 1000 in `$tree$maps`)

Author(s)

Maël Doré

See Also

`cut_phylo_for_focal_time()` `extract_most_likely_trait_values_for_focal_time()` `extract_all_trait_values`

For a guided tutorial, see this vignette: `vignette("cut_phylogenies", package = "deepSTRAPP")`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## The R package 'contsimmap' is needed for this example to work.
 # Please install it manually from: https://github.com/bstaggmartin/contsimmap.
```

```
# ----- Prepare data ----- #

# Load phylogeny and tip data
# Load eel phylogeny and tip data from the R package phytools
# Source: Collar et al., 2014; DOI: 10.1038/ncomms6505
data("eel.tree", package = "phytools")
data("eel.data", package = "phytools")

# Extract body size
eel_data <- stats::setNames(eel.data$Max_TL_cm,
                            rownames(eel.data))

 # (May take several seconds to run)
## Map multiple trait evolution histories on the phylogeny using continuous stochastic mapping
eel_mapped_trait_data <- prepare_trait_data(
  tip_data = eel_data,
  trait_data_type = "continuous",
  phylo = eel.tree,
  evolutionary_models = "BM",
  run_stochastic_maps = TRUE,
  nb_simulations = 10,
  plot_map = FALSE,
  verbose = TRUE)

# Extract list of stochastic maps
eel_contMaps <- eel_mapped_trait_data$contMaps

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut contMaps to 20 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_contMaps <- cut_contMaps_for_focal_time(contMaps = eel_contMaps,
                                                focal_time = 20,
                                                keep_tip_labels = TRUE)

# Plot node labels on stochastic map n°1 with cut-off
plot_contMap(eel_contMaps[[1]], lwd = 2, fsize = c(0.5, 1))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(eel_contMaps[[1]]$tree)[,2]) - 20,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map n°1
plot_contMap(updated_contMaps[[1]], fsize = c(0.8, 1))
ape::nodelabels(cex = 0.8, text = updated_contMaps[[1]]$tree$initial_nodes_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut contMap to 20 Mya while NOT keeping tip.label.
updated_contMaps <- cut_contMaps_for_focal_time(contMaps = eel_contMaps,
                                                focal_time = 20,
                                                keep_tip_labels = FALSE)

# Plot node labels on stochastic map n°1 with cut-off
```

```
plot_contMap(eel_contMaps[[1]], fsize = c(0.5, 1))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(eel_contMaps[[1]]$tree)[,2]) - 20,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
plot_contMap(updated_contMaps[[1]], fsize = c(0.8, 1))
ape::nodelabels(cex = 0.8, text = updated_contMaps[[1]]$tree$initial_nodes_ID)

}
```

`cut_contMap_for_focal_time` *Cut the phylogeny and continuous trait mapping for a given focal time in the past*

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise, remove continuous trait mapping for the cut off branches by updating the `$tree$maps` and `$tree$mapped.edge` elements.

Usage

```
cut_contMap_for_focal_time(contMap, focal_time, keep_tip_labels = TRUE)
```

Arguments

`contMap`	Object of class `"contMap"`, typically generated with `phytools::contMap()`, that contains a phylogenetic tree and associated continuous trait mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

The phylogenetic tree is cut for a specific time in the past (i.e. the `focal_time`).

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The continuous trait mapping is updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns the cut contMap as an object of class `"contMap"` as a list of three elements:

- `$tree`. A list of classes `"simmap"` and `"phylo"`. This function updates and adds multiple useful sub-elements to the `$tree` element.
  - `$maps` An updated list of named numeric vectors. Provides the mapping of trait values along each remaining edge.
  - `$mapped.edge` An updated matrix. Provides the evolutionary time spent across trait values (columns) along the remaining edges (rows).
  - `$root_age` Integer. Stores the age of the root of the tree.
  - `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
  - `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
  - `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
  - `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.
- `$cols` A named character vector mapping trait values with colors to display on the plot.
- `$lims` A numeric vector of 2 storing the initial min and max values of the trait (trait values are scaled between 0 and 1000 in `$tree$maps`)

Author(s)

Maël Doré

See Also

`cut_phylo_for_focal_time()` `extract_most_likely_trait_values_for_focal_time()`

For a guided tutorial, see this vignette: `vignette("cut_phylogenies", package = "deepSTRAPP")`

Examples

```
# ----- Prepare data ----- #

# Load mammals phylogeny and data from the R package motmot (data included in deepSTRAPP)
# Initial data source: Slater, 2013; DOI: 10.1111/2041-210X.12084

data(mammals, package = "deepSTRAPP")

mammals_tree <- mammals$mammal.phy
mammals_data <- setNames(object = mammals$mammal.mass$mean,
```

```
                        nm = row.names(mammals$mammal.mass))[mammals_tree$tip.label]

# Run a stochastic mapping based on a Brownian Motion model
# for Ancestral Trait Estimates to obtain a "contMap" object
mammals_contMap <- phytools::contMap(mammals_tree, x = mammals_data,
                                     res = 100, # Number of time steps
                                     plot = FALSE)

# Set focal time
focal_time <- 80

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut contMap to 80 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_contMap <- cut_contMap_for_focal_time(contMap = mammals_contMap,
                                              focal_time = focal_time,
                                              keep_tip_labels = TRUE)

# Plot node labels on initial stochastic map with cut-off
plot_contMap(mammals_contMap, lwd = 2, fsize = c(0.5, 1))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_contMap$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
plot_contMap(updated_contMap, fsize = c(0.8, 1))
ape::nodelabels(cex = 0.8, text = updated_contMap$tree$initial_nodes_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut contMap to 80 Mya while NOT keeping tip.label.
updated_contMap <- cut_contMap_for_focal_time(contMap = mammals_contMap,
                                              focal_time = focal_time,
                                              keep_tip_labels = FALSE)

# Plot node labels on initial stochastic map with cut-off
plot_contMap(mammals_contMap, fsize = c(0.5, 1))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_contMap$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
plot_contMap(updated_contMap, fsize = c(0.8, 1))
ape::nodelabels(cex = 0.8, text = updated_contMap$tree$initial_nodes_ID)
```

`cut_densityMaps_for_focal_time`

Cut phylogenies and posterior probability mapping of each state for a given focal time in the past

---

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise, remove posterior probability mapping of the categorical trait for the cut off branches by updating the `$tree$maps` and `$tree$mapped.edge` elements.

Usage

```
cut_densityMaps_for_focal_time(densityMaps, focal_time, keep_tip_labels = TRUE)
```

Arguments

`densityMaps`	List of objects of class `"densityMap"`, typically generated with `prepare_trait_data()`. Each densityMap (see `phytools::densityMap()`) contains a phylogenetic tree and associated posterior probability mapping of a categorical trait. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

The phylogenetic tree is cut for a specific time in the past (i.e. the `focal_time`).

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The continuous trait mapping is updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns an updated list of objects as cut densityMaps of class `"densityMap"`.

Each densityMap object contains three elements:

- `$tree` element of classes `"simmap"` and `"phylo"`. This function updates and adds multiple useful sub-elements to the `$tree` element.
  - `$maps` An updated list of named numeric vectors. Provides the mapping of posterior probability of the state along each remaining edge.
  - `$mapped.edge` An updated matrix. Provides the evolutionary time spent across posterior probabilities (columns) along the remaining edges (rows).
  - `$root_age` Integer. Stores the age of the root of the tree.

  - $nodes_ID_df Data.frame with two columns. Provides the conversion from the new_node_ID to the initial_node_ID. Each row is a node.
  - $initial_nodes_ID Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with ape::nodelabels().
  - $edges_ID_df Data.frame with two columns. Provides the conversion from the new_edge_ID to the initial_edge_ID. Each row is an edge/branch.
  - $initial_edges_ID Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with ape::edgelabels().
- $col element describes the colors used to map each possible posterior probability value from 0 to 1000.
- $states element provide the name of the states. Here, the first value is the absence of the state labeled as "Not X" with X being the state. The second value is the name of the state.

High posterior probability reflects high likelihood to harbor the state. Low probability reflects high likelihood to NOT harbor the state.

Author(s)

Maël Doré

See Also

cut_phylo_for_focal_time() cut_densityMap_for_focal_time() extract_most_likely_trait_values_for_focal
extract_all_trait_values_for_focal_time()

Examples

```
# ----- Prepare data ----- #

# Load mammals phylogeny and data from the R package motmot, and implemented in deepSTRAPP
# Data source: Slater, 2013; DOI: 10.1111/2041-210X.12084
data("mammals", package = "deepSTRAPP")

# Obtain mammal tree
mammals_tree <- mammals$mammal.phy
# Convert mass data into categories
mammals_mass <- setNames(object = mammals$mammal.mass$mean,
                         nm = row.names(mammals$mammal.mass))[mammals_tree$tip.label]
mammals_data <- mammals_mass
mammals_data[seq_along(mammals_data)] <- "small"
mammals_data[mammals_mass > 5] <- "medium"
mammals_data[mammals_mass > 10] <- "large"
table(mammals_data)

 # (May take several minutes to run)
# Produce densityMaps using stochastic character mapping based on an equal-rates (ER) Mk model
mammals_cat_data <- prepare_trait_data(tip_data = mammals_data, phylo = mammals_tree,
                                       trait_data_type = "categorical",
                                       evolutionary_models = "ER",
                                       nb_simulations = 100,
                                       plot_map = FALSE)
```

```
# Set focal time
focal_time <- 80

# Extract the density maps
mammals_densityMaps <- mammals_cat_data$densityMaps

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut densityMaps to 80 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_mammals_densityMaps <- cut_densityMaps_for_focal_time(
    densityMaps = mammals_densityMaps,
    focal_time = focal_time,
    keep_tip_labels = TRUE)

## Plot density maps as overlay of all state posterior probabilities
# ?plot_densityMaps_overlay

# Plot initial density maps
plot_densityMaps_overlay(densityMaps = mammals_densityMaps, fsize = 0.5)
abline(v = max(phytools::nodeHeights(mammals_densityMaps[[1]]$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated/cut density maps
plot_densityMaps_overlay(densityMaps = updated_mammals_densityMaps, fsize = 0.8)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut densityMap to 80 Mya while NOT keeping tip.label
updated_mammals_densityMaps <- cut_densityMaps_for_focal_time(
    densityMaps = mammals_densityMaps,
    focal_time = focal_time,
    keep_tip_labels = FALSE)

# Plot initial density maps
plot_densityMaps_overlay(densityMaps = mammals_densityMaps, fsize = 0.5)
abline(v = max(phytools::nodeHeights(mammals_densityMaps[[1]]$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated/cut density maps
plot_densityMaps_overlay(densityMaps = updated_mammals_densityMaps, fsize = 0.8)
```

`cut_densityMap_for_focal_time`

Cut the phylogeny and posterior probability mapping of a categorical trait for a given focal time in the past

---

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise, remove posterior probability mapping of the categorical trait for the cut off branches by updating the `$tree$maps` and `$tree$mapped.edge` elements.

Usage

```
cut_densityMap_for_focal_time(densityMap, focal_time, keep_tip_labels = TRUE)
```

Arguments

`densityMap`	Object of class `"densityMap"`, typically generated with `phytools::densityMap()`, that contains a phylogenetic tree and associated posterior probability mapping of a categorical trait. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

The phylogenetic tree is cut for a specific time in the past (i.e. the `focal_time`).

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The posterior probability mapping of a categorical trait is updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns the cut densityMap as an object of class `"densityMap"` with three elements.

It contains a `$tree` element of classes `"simmap"` and `"phylo"`. This function updates and adds multiple useful sub-elements to the `$tree` element. * `$maps` An updated list of named numeric vectors. Provides the mapping of posterior probability of the state along each remaining edge. * `$mapped.edge` An updated matrix. Provides the evolutionary time spent across posterior probabilities (columns) along the remaining edges (rows). * `$root_age` Integer. Stores the age of the root of the tree. * `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node. * `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`. * `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch. * `$initial_edges_ID`

Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.

The `$col` element describes the colors used to map each possible posterior probability value from 0 to 1000.

The `$states` element provides the name of the states. Here, the first value is the absence of the state labeled as "Not X" with X being the state. The second value is the name of the state.

High posterior probability reflects high likelihood to harbor the state. Low probability reflects high likelihood to NOT harbor the state.

Author(s)

Maël Doré

See Also

`cut_phylo_for_focal_time()` `extract_most_likely_trait_values_for_focal_time()` `extract_all_trait_values`

For a guided tutorial, see this vignette: `vignette("cut_phylogenies", package = "deepSTRAPP")`

Examples

```
# ----- Prepare data ----- #

# Load mammals phylogeny and data from the R package motmot included within deepSTRAPP
# Data source: Slater, 2013; DOI: 10.1111/2041-210X.12084
data("mammals", package = "deepSTRAPP")

# Obtain mammal tree
mammals_tree <- mammals$mammal.phy
# Convert mass data into categories
mammals_mass <- setNames(object = mammals$mammal.mass$mean,
                         nm = row.names(mammals$mammal.mass))[mammals_tree$tip.label]
mammals_data <- mammals_mass
mammals_data[seq_along(mammals_data)] <- "small"
mammals_data[mammals_mass > 5] <- "medium"
mammals_data[mammals_mass > 10] <- "large"
table(mammals_data)

 # (May take several minutes to run)
# Produce densityMaps using stochastic character mapping based on an equal-rates (ER) Mk model
mammals_cat_data <- prepare_trait_data(tip_data = mammals_data, phylo = mammals_tree,
                                       trait_data_type = "categorical",
                                       evolutionary_models = "ER",
                                       nb_simulations = 100,
                                       plot_map = FALSE)

# Set focal time
focal_time <- 80

# Extract the density map for small mammals (state 3 = "small")
mammals_densityMap_small <- mammals_cat_data$densityMaps[[3]]
```

```
# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut densityMap to 80 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_mammals_densityMap_small <- cut_densityMap_for_focal_time(
     densityMap = mammals_densityMap_small,
     focal_time = focal_time,
     keep_tip_labels = TRUE)

# Plot node labels on initial stochastic map with cut-off
phytools::plot.densityMap(mammals_densityMap_small, fsize = 0.7, lwd = 2)
ape::nodelabels(cex = 0.7)
abline(v = max(phytools::nodeHeights(mammals_densityMap_small$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
phytools::plot.densityMap(updated_mammals_densityMap_small, fsize = 0.8)
ape::nodelabels(cex = 0.8, text = updated_mammals_densityMap_small$tree$initial_nodes_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut densityMap to 80 Mya while NOT keeping tip.label
updated_mammals_densityMap_small <- cut_densityMap_for_focal_time(
     densityMap = mammals_densityMap_small,
     focal_time = focal_time,
     keep_tip_labels = FALSE)

# Plot node labels on initial stochastic map with cut-off
phytools::plot.densityMap(mammals_densityMap_small, fsize = 0.7, lwd = 2)
ape::nodelabels(cex = 0.7)
abline(v = max(phytools::nodeHeights(mammals_densityMap_small$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
phytools::plot.densityMap(updated_mammals_densityMap_small, fsize = 0.8)
ape::nodelabels(cex = 0.8, text = updated_mammals_densityMap_small$tree$initial_nodes_ID)
```

`cut_phylo_for_focal_time` *Cut the phylogeny for a given time in the past*

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`.

Usage

```
cut_phylo_for_focal_time(tree, focal_time, keep_tip_labels = TRUE)
```

Arguments

`tree`	Object of class `"phylo"`. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

Value

The function returns the cut phylogeny as an object of class `"phylo"`. It adds multiple useful elements to the object.

- `$root_age` Integer. Stores the age of the root of the tree.
- `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
- `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
- `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
- `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.

Author(s)

Maël Doré

See Also

`cut_contMap_for_focal_time()` `cut_densityMaps_for_focal_time()`

For a guided tutorial, see this vignette: `vignette("cut_phylogenies", package = "deepSTRAPP")`

Examples

```
# Load eel phylogeny from the R package phytools
# Source: Collar et al., 2014; DOI: 10.1038/ncomms6505
data("eel.tree", package = "phytools")

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut tree to 30 Mya while keeping tip.label on terminal branches with a unique descending tip.
cut_eel.tree <- cut_phylo_for_focal_time(tree = eel.tree, focal_time = 30, keep_tip_labels = TRUE)

# Plot internal node labels on initial tree with cut-off
plot(eel.tree, cex = 0.5)
abline(v = max(phytools::nodeHeights(eel.tree)[,2]) - 30, col = "red", lty = 2, lwd = 2)
nb_tips <- length(eel.tree$tip.label)
nodelabels_in_cut_tree <- (nb_tips + 1):(nb_tips + eel.tree$Nnode)
nodelabels_in_cut_tree[!(nodelabels_in_cut_tree %in% cut_eel.tree$initial_nodes_ID)] <- NA
ape::nodelabels(text = nodelabels_in_cut_tree)

# Plot initial internal node labels on cut tree
plot(cut_eel.tree, cex = 0.8)
ape::nodelabels(text = cut_eel.tree$initial_nodes_ID)

# Plot edge labels on initial tree with cut-off
plot(eel.tree, cex = 0.5)
abline(v = max(phytools::nodeHeights(eel.tree)[,2]) - 30, col = "red", lty = 2, lwd = 2)
edgelabels_in_cut_tree <- 1:nrow(eel.tree$edge)
edgelabels_in_cut_tree[!(1:nrow(eel.tree$edge) %in% cut_eel.tree$initial_edges_ID)] <- NA
ape::edgelabels(text = edgelabels_in_cut_tree)

# Plot initial edge labels on cut tree
plot(cut_eel.tree, cex = 0.8)
ape::edgelabels(text = cut_eel.tree$initial_edges_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut tree to 30 Mya without keeping tip.label on terminal branches with a unique descending tip.
# All tip.labels are converted to their descending/tipward node ID
cut_eel.tree <- cut_phylo_for_focal_time(tree = eel.tree, focal_time = 30, keep_tip_labels = FALSE)
plot(cut_eel.tree, cex = 0.8)
```

`cut_simmaps_for_focal_time`

Cut the phylogenies and categorical trait/range mappings of a list of simmaps for a given focal time in the past

---

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise,

remove continuous trait mapping for the cut off branches by updating the $maps and $mapped.edge elements.

Usage

```
cut_simmaps_for_focal_time(simmaps, focal_time, keep_tip_labels = TRUE)
```

Arguments

simmaps	List of objects with the classes "phylo" and "simmap", typically generated with phytools::make.simmap(), that contains a phylogenetic tree and associated categorical trait/range mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
focal_time	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
keep_tip_labels	Logical. Specify whether terminal branches with a single descendant tip must retain their initial tip.label. Default is TRUE.

Details

The phylogenetic trees are cut for a specific time in the past (i.e. the focal_time).

When a branch with a single descendant tip is cut and keep_tip_labels = TRUE, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and keep_tip_labels = FALSE, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The categorical trait/range mappings are updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns an updated list of objects as cut/updated simmaps of classes "phylo" and "simmap".

Each simmap object represents an updated stochastic map cut/updated simmap as a list of at least six elements:

Initial updated elements:

- $edge A numeric matrix with two columns listing the updated ID of rootward and tipward nodes of the remaining edges
- $edge.length A numeric vector. Updated length of remaining edges.
- $Nnode Integer. Number of remaining internal nodes.
- $tip.label Character vector. Labels of the branches cut at the focal_time.
  - If keep_tip_labels = TRUE, the cut terminal branches are labeled with the tip.label of their unique descendant tip.

  - If `keep_tip_labels = FALSE`, the cut terminal branches are labeled with the node ID of their unique descendant tip.
- `$maps` An updated list of named numeric vectors. Provides the mapping of trait values along each remaining edge.
- `$mapped.edge` An updated matrix. Provides the evolutionary time spent across trait values (columns) along the remaining edges (rows).

Additional elements:

- `$root_age` Integer. Stores the age of the root of the tree.
- `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
- `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
- `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
- `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.

Author(s)

Maël Doré

See Also

`cut_phylo_for_focal_time()` `extract_most_likely_trait_values_for_focal_time()` `extract_all_trait_values`

For a guided tutorial, see this vignette: `vignette("cut_phylogenies", package = "deepSTRAPP")`

Examples

```
# ----- Prepare data ----- #

## Load mammals phylogeny and data from the R package motmot, and implemented in deepSTRAPP
# Data source: Slater, 2013; DOI: 10.1111/2041-210X.12084
data("mammals", package = "deepSTRAPP")

# Obtain mammal tree
mammals_tree <- mammals$mammal.phy
# Convert mass data into categories
mammals_mass <- setNames(object = mammals$mammal.mass$mean,
                         nm = row.names(mammals$mammal.mass))[mammals_tree$tip.label]
mammals_data <- mammals_mass
mammals_data[seq_along(mammals_data)] <- "small"
mammals_data[mammals_mass > 5] <- "medium"
mammals_data[mammals_mass > 10] <- "large"
table(mammals_data)

 # (May take several seconds to run)
# Produce densityMaps using stochastic character mapping based on an equal-rates (ER) Mk model
mammals_cat_data <- prepare_trait_data(tip_data = mammals_data, phylo = mammals_tree,
                                       trait_data_type = "categorical",
```

```
                                        evolutionary_models = "ER",
                                        nb_simulations = 100,
                                        return_simmaps = TRUE,
                                        plot_map = FALSE)

# Extract the simulated evolutionary histories (simmaps)
mammals_simmaps <- mammals_cat_data$simmaps

# Set focal time to 80Mya
focal_time <- 80

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut the simmap to 80 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_simmaps <- cut_simmaps_for_focal_time(simmaps = mammals_simmaps,
                                              focal_time = focal_time,
                                              keep_tip_labels = TRUE)

# Plot node labels on initial stochastic map n°1 with cut-off
plot(mammals_simmaps[[1]], lwd = 2, fsize = 0.5,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_simmaps[[1]])[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map n°1
plot(updated_simmaps[[1]], fsize = 0.8,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.8, text = updated_simmaps[[1]]$initial_nodes_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut the simmap to 80 Mya while NOT keeping tip.label.
updated_simmaps <- cut_simmaps_for_focal_time(simmaps = mammals_simmaps,
                                              focal_time = focal_time,
                                              keep_tip_labels = FALSE)

# Plot node labels on initial stochastic map n°1 with cut-off
plot(mammals_simmaps[[1]], lwd = 2, fsize = 0.5,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_simmaps[[1]])[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map n°1
plot(updated_simmaps[[1]], fsize = 0.8,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.8, text = updated_simmaps[[1]]$initial_nodes_ID)
```

`cut_simmap_for_focal_time` *Cut the phylogeny and categorical trait/range mapping for a given focal time in the past*

Description

Cuts off all the branches of the phylogeny which are younger than a specific time in the past (i.e. the `focal_time`). Branches overlapping the `focal_time` are shortened to the `focal_time`. Likewise, remove mapping for the cut off branches by updating the `$maps` and `$mapped.edge` elements.

Usage

```
cut_simmap_for_focal_time(simmap, focal_time, keep_tip_labels = TRUE)
```

Arguments

`simmap`	Object with the classes `"phylo"` and `"simmap"`, typically generated with `phytools::make.simmap()`, that contains a phylogenetic tree and associated categorical trait/range mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`focal_time`	Numeric. The time, in terms of time distance from the present, for which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.
`keep_tip_labels`	Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label`. Default is `TRUE`.

Details

The mapped phylogenetic tree is cut for a specific time in the past (i.e. the `focal_time`).

When a branch with a single descendant tip is cut and `keep_tip_labels = TRUE`, the leaf left is labeled with the tip.label of the unique descendant tip.

When a branch with a single descendant tip is cut and `keep_tip_labels = FALSE`, the leaf left is labeled with the node ID of the unique descendant tip.

In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The stochastic trait mapping is updated accordingly by removing mapping associated with the cut off branches.

Value

The function returns the cut/updated simmap as a list of at least six elements with the classes `"phylo"` and `"simmap"`:

Initial updated elements:

- $edge A numeric matrix with two columns listing the updated ID of rootward and tipward nodes of the remaining edges
- $edge.length A numeric vector. Updated length of remaining edges.
- $Nnode Integer. Number of remaining internal nodes.
- $tip.label Character vector. Labels of the branches cut at the focal_time.
  - If keep_tip_labels = TRUE, the cut terminal branches are labeled with the tip.label of their unique descendant tip.
  - If keep_tip_labels = FALSE, the cut terminal branches are labeled with the node ID of their unique descendant tip.
- $maps An updated list of named numeric vectors. Provides the mapping of trait values along each remaining edge.
- $mapped.edge An updated matrix. Provides the evolutionary time spent across trait values (columns) along the remaining edges (rows).

Additional elements:

- $root_age Integer. Stores the age of the root of the tree.
- $nodes_ID_df Data.frame with two columns. Provides the conversion from the new_node_ID to the initial_node_ID. Each row is a node.
- $initial_nodes_ID Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with ape::nodelabels().
- $edges_ID_df Data.frame with two columns. Provides the conversion from the new_edge_ID to the initial_edge_ID. Each row is an edge/branch.
- $initial_edges_ID Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with ape::edgelabels().

Author(s)

Maël Doré

See Also

cut_phylo_for_focal_time() extract_most_likely_trait_values_for_focal_time() extract_all_trait_values

For a guided tutorial, see this vignette: vignette("cut_phylogenies", package = "deepSTRAPP")

Examples

```
# ----- Prepare data ----- #

## Load mammals phylogeny and data from the R package motmot, and implemented in deepSTRAPP
# Data source: Slater, 2013; DOI: 10.1111/2041-210X.12084
data("mammals", package = "deepSTRAPP")

# Obtain mammal tree
mammals_tree <- mammals$mammal.phy
# Convert mass data into categories
mammals_mass <- setNames(object = mammals$mammal.mass$mean,
                         nm = row.names(mammals$mammal.mass))[mammals_tree$tip.label]
mammals_data <- mammals_mass
```

```
mammals_data[seq_along(mammals_data)] <- "small"
mammals_data[mammals_mass > 5] <- "medium"
mammals_data[mammals_mass > 10] <- "large"
table(mammals_data)
 # (May take several seconds to run)
# Produce densityMaps using stochastic character mapping based on an equal-rates (ER) Mk model
mammals_cat_data <- prepare_trait_data(tip_data = mammals_data, phylo = mammals_tree,
                                       trait_data_type = "categorical",
                                       evolutionary_models = "ER",
                                       nb_simulations = 100,
                                       return_simmaps = TRUE,
                                       plot_map = FALSE)

# Extract the simulated evolutionary histories (simmaps)
mammals_simmaps <- mammals_cat_data$simmaps
# Extract simulation n°1
mammals_simmap_1 <- mammals_simmaps[[1]]

# Set focal time to 80Mya
focal_time <- 80

# ----- Example 1: keep_tip_labels = TRUE ----- #

# Cut the simmap to 80 Mya while keeping tip.label
# on terminal branches with a unique descending tip.
updated_simmap_1 <- cut_simmap_for_focal_time(simmap = mammals_simmap_1,
                                              focal_time = focal_time,
                                              keep_tip_labels = TRUE)

# Plot node labels on initial stochastic map with cut-off
plot(mammals_simmap_1, lwd = 2, fsize = 0.5,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_simmap_1)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
plot(updated_simmap_1, fsize = 0.8,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.8, text = updated_simmap_1$initial_nodes_ID)

# ----- Example 2: keep_tip_labels = FALSE ----- #

# Cut the simmap to 80 Mya while NOT keeping tip.label.
updated_simmap_1 <- cut_simmap_for_focal_time(simmap = mammals_simmap_1,
                                              focal_time = focal_time,
                                              keep_tip_labels = FALSE)

# Plot node labels on initial stochastic map with cut-off
plot(mammals_simmap_1, lwd = 2, fsize = 0.5,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.5)
abline(v = max(phytools::nodeHeights(mammals_simmap_1)[,2]) - focal_time,
```

```
       col = "red", lty = 2, lwd = 2)

# Plot initial node labels on cut stochastic map
plot(updated_simmap_1, fsize = 0.8,
     colors = c(large = "black", medium = "blue", small = "dodgerblue"))
ape::nodelabels(cex = 0.8, text = updated_simmap_1$initial_nodes_ID)
```

`eel_biogeo_data` *Data summarizing the evolution of geographic ranges in eels*

Description

A list containing (fake) geographic ranges data of eels mapped on the phylogeny, modeled with R package `BioGeoBEARS`. This object was obtained with `prepare_trait_data()`. Initial data based on feeding habits was altered to be transformed into ranges "A" and "B", and then adding arbitrarily multi-area "AB" ranges. This is NOT real biogeographic data. Please refer to the initial article for real data.

Original data source: Collar, D. C., P. C. Wainwright, M. E. Alfaro, L. J. Revell, and R. S. Mehta (2014) Biting disrupts integration to spur skull evolution in eels. Nature Communications, 5, 5505. doi:10.1038/ncomms6505

Usage

```
data(eel_biogeo_data)
```

Format

A list with 9 elements.

Details

A list of 9 elements containing information on the evolution of geographic ranges in eels. This object was obtained with `prepare_trait_data()`.

- `$densityMaps` List of objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given range along branches. The list contains only a `"densityMap"` per unique area because `split_multi_area_ranges` was set to TRUE.
- `$densityMaps_all_ranges` List of objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given range along branches. The list contains one `"densityMap"` per range found along branches during the simulated biogeographic histories.
- `$trait_data_type` Character string. Record the type of trait data. Here: "biogeographic".
- `$ace` Numeric matrix. Record the posterior probabilities of ancestral ranges estimated at internal nodes. Only unique areas are considered among the ranges. Multi-area ranges have been split among unique ranges. Rows are internal nodes. Columns are ranges. Values are posterior probabilities of each range per node.

- `$ace_all_ranges` Numeric matrix. Record the posterior probabilities of ancestral ranges estimated at internal nodes. All ranges observed along branches during the simulated biogeographic histories are present. Rows are internal nodes. Columns are ranges. Values are posterior probabilities of each range per node.
- `$BSM_output` List of two lists that contain summary information on cladogenetic (`$RES_caldo_events_tables`) and anagenetic (`$RES_ana_events_tables`) events recorded across the 1000 simulations of biogeographic histories performed during Biogeographic Stochastic Mapping (BSM). Each element of the list is a data.frame recording events occurring during one simulation.
- `$simmaps` List of 1000 objects of class `"simmap"`. Each simmap object is a phylogeny with one simulated biogeographic history (i.e., transitions in geographic ranges) mapped along branches.
- `$best_model_fit` List that provides the output of the best fitting model (Here: DEC+J model).
- `$model_selection_df` Data.frame that summarizes model comparisons used to select the best fitting model.

References

Collar, D. C., P. C. Wainwright, M. E. Alfaro, L. J. Revell, and R. S. Mehta (2014) Biting disrupts integration to spur skull evolution in eels. Nature Communications, 5, 5505. doi:10.1038/ncomms6505

See Also

`prepare_trait_data()`

`eel_cat_3lvl_data` *Data summarizing the evolution of feeding habits in eels using a 3-level factor as categorical trait*

Description

A list containing feeding habits data of eels mapped on the phylogeny, modeled with geiger::fitDiscrete. This object was obtained with `prepare_trait_data()`. Initial data was altered arbitrarily to create three categories, adding a "kiss" feeding habit to the initial "bite" and "suction" data. This is NOT real biological data. Please refer to the initial article for real data.

Original data source: Collar, D. C., P. C. Wainwright, M. E. Alfaro, L. J. Revell, and R. S. Mehta (2014) Biting disrupts integration to spur skull evolution in eels. Nature Communications, 5, 5505. doi:10.1038/ncomms6505

Usage

```
data(eel_cat_3lvl_data)
```

Format

A list with 6 elements.

Details

A list of six elements containing information on the evolution of feeding habits in eels. This object was obtained with `prepare_trait_data()`.

- `$densityMaps` List of objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given state/range along branches. The list contains one `"densityMap"` per state/range found in the `tip_data`.
- `$trait_data_type` Character string. Record the type of trait data. Here: "categorical".
- `$simmaps` List of 100 stochastic mapping simulations for trait evolution. Each element is a `"simmap"` object (see phytools::make.simmap) representing a possible evolutionary history that fits states observed on tips, inferred ACE at internal nodes, and transition rates as estimated from the best fit model.
- `$ace` Numeric matrix. Record the posterior probabilities of ancestral states/ranges (characters) estimates (ACE) at internal nodes. Rows are internal nodes. Columns are states/ranges. Values are posterior probabilities of each state per node.
- `$best_model_fit` List that provides the output of the best fitting model (Here: ER model).
- `$model_selection_df` Data.frame that summarizes model comparisons used to select the best fitting model.

References

Collar, D. C., P. C. Wainwright, M. E. Alfaro, L. J. Revell, and R. S. Mehta (2014) Biting disrupts integration to spur skull evolution in eels. Nature Communications, 5, 5505. doi:10.1038/ncomms6505

See Also

`prepare_trait_data()`

`extract_all_trait_values_for_focal_time` *Extract all trait data from stochastic maps at a given time in the past*

Description

Extracts all trait values/states/ranges found along branches of multiple stochastic maps at a specific time in the past (i.e. the `focal_time`). Optionally, the function can update the mapped phylogenies (`contMaps` or `densityMaps`) such as branches overlapping the `focal_time` are shortened to the `focal_time`, and the trait mapping for the cut off branches are removed by updating the `$maps` and `$mapped.edge` elements.

Usage

```
extract_all_trait_values_for_focal_time(
  contMaps = NULL,
  densityMaps = NULL,
```

```
  simmaps = NULL,
  nb_simulations = NULL,
  tip_data = NULL,
  trait_data_type,
  focal_time,
  update_Map = FALSE,
  keep_tip_labels = TRUE
)
```

Arguments

contMaps	For continuous trait data. List of objects of class "contMap", typically generated with prepare_trait_data(), each containing a phylogenetic tree and associated continuous trait mapping that represents an independent evolutionary history of ancestral trait evolution, conditioned on observed trait data and model fit (i.e., stochastic maps). The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
densityMaps	For categorical trait or biogeographic data. List of objects of class "densityMap", typically generated with prepare_trait_data(), that contains a phylogenetic tree and associated posterior probability of being in a given state/range along branches. Each object (i.e., densityMap) corresponds to a state/range. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
simmaps	For categorical trait or biogeographic data. List of objects of classes "phylo" and "simmap", typically generated with prepare_trait_data() or phytools::make.simmap(), that represent discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches. This is needed to be able to track which simulated history provided which trait data in downstream analyses employing the "paired" or "full" strategies to account for uncertainty in ancestral trait estimates.
nb_simulations	For categorical trait or biogeographic data. Integer. The number of stochastic maps used to simulate trait evolution. This is needed for the "paired" and "full" strategies to account for trait estimate uncertainty, if only densityMaps summarizing posterior state/range density are provided, but not the simmaps representing all evolutionary histories.
tip_data	(Optional) Named vector of tip values of the trait. • For continuous trait data: Named numeric vector of trait values. • For categorical trait or biogeographic data: Character string vector of states/ranges Names are nodes_ID of the internal nodes. Ensure accurate tip values are used.
trait_data_type	Character string. Specify the type of trait data. Must be one of "continuous", "categorical", "biogeographic".
focal_time	Integer. The time, in terms of time distance from the present, at which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.

update_Map Logical. Specify whether the mapped phylogeny (contMap, densityMaps, and/or simmaps) provided as input should be updated for visualization and returned among the outputs. Default is FALSE. The update consists of cutting off branches and mapping that are younger than the focal_time.

keep_tip_labels Logical. Specify whether terminal branches with a single descendant tip must retain their initial tip.label on the updated contMap. Default is TRUE. Used only if update_Map = TRUE.

Details

The mapped phylogenies (contMaps, densityMaps, or simmaps) are cut at a specific time in the past (i.e. the focal_time) and the associated trait values of the overlapping edges/branches are extracted.

—— Extract trait_data ——

For continuous trait data:

Simulated trait data are extracted from contMaps (i.e. continuous stochastic maps).

For categorical trait and biogeographic data:

If simmaps are provided as input, all states/ranges are extracted directly from the stochastic maps.

If only densityMaps are provided, the posterior probabilities of states/ranges are extracted. Posterior probabilities are multiplied by the nb_simulations to record the distribution of states/ranges across dummy stochastic maps and assigned to each tip and cut branches at focal_time.

True tip states/ranges will be used if tip_data are provided as optional inputs. In practice the discrepancy is negligible.

—— Update the contMap/densityMaps/simmaps ——

To obtain an updated contMap/densityMaps/simmaps alongside the trait data, set update_Map = TRUE. The update consists of cutting off branches and mapping that are younger than the focal_time.

- When a branch with a single descendant tip is cut and keep_tip_labels = TRUE, the leaf left is labeled with the tip.label of the unique descendant tip.
- When a branch with a single descendant tip is cut and keep_tip_labels = FALSE, the leaf left is labeled with the node ID of the unique descendant tip.
- In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The mapping in contMap/densityMaps/simmaps ($maps and $mapped.edge) is updated accordingly by removing mapping associated with the cut off branches.

To extract only the most likely trait value/state/range, see this associated function: extract_most_likely_trait_values_fo

Value

By default, the function returns a list with three elements.

- $trait_data A list of named numeric vectors with simulated trait values found along branches overlapping the focal_time across each of the stochastic maps. Names are the tip.label/tipward node ID. ID of the associated stochastic maps associated with each item of the list can only be

provided if simmaps are provided as inputs. If only densityMaps are provided, the distribution of states/ranges is associated with dummy maps_ID.

- `$focal_time` Integer. The time, in terms of time distance from the present, at which the trait data were extracted.
- `$trait_data_type` Character string. Define the type of trait data as "continuous", "categorical", or "biogeographic". Used in downstream analyses to select appropriate statistical processing.

If `update_Map = TRUE`, the output is a list with four to five elements: `$trait_data`, `$focal_time`, `$trait_data_type`, `$contMaps` or `$densityMaps`, and/or `$simmaps`.

For continuous trait data:

- `$contMaps` A list of objects of class `"contMap"` that contains the updated `contMaps` with branches and mapping that are younger than the `focal_time` cut off. The function also adds multiple useful sub-elements to the `$contMap$tree` elements.
  - `$root_age` Integer. Stores the age of the root of the tree.
  - `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
  - `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
  - `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
  - `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.

For categorical trait and biogeographic data:

- `$densityMaps` A list of objects of class `"densityMap"` that contains the updated `densityMap` of each state/range, with branches and mapping that are younger than the `focal_time` cut off. The function also adds multiple useful sub-elements to the `$densityMaps$tree` elements.
  - `$root_age` Integer. Stores the age of the root of the tree.
  - `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.
  - `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
  - `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
  - `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.
- `$simmaps` A list of objects with the class `"simmap"` that contains the updated `simmap =` stochastic maps representing discrete character/geographic evolutionary history, with branches and mapping that are younger than the `focal_time` cut off. The function also adds multiple useful sub-elements to each `simmap`.
  - `$root_age` Integer. Stores the age of the root of the tree.
  - `$nodes_ID_df` Data.frame with two columns. Provides the conversion from the `new_node_ID` to the `initial_node_ID`. Each row is a node.

- `$initial_nodes_ID` Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with `ape::nodelabels()`.
- `$edges_ID_df` Data.frame with two columns. Provides the conversion from the `new_edge_ID` to the `initial_edge_ID`. Each row is an edge/branch.
- `$initial_edges_ID` Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with `ape::edgelabels()`.

Author(s)

Maël Doré

See Also

`cut_phylo_for_focal_time()` `cut_contMaps_for_focal_time()` `cut_densityMaps_for_focal_time()` `cut_simmaps_for_focal_time()`

Equivalent function to extract the most likely trait value/state/range from mapped phylogenies: `extract_most_likely_trait_values_for_focal_time()`

Examples

```
# ----- Example 1: Continuous trait ----- #

if (deepSTRAPP::is_dev_version())
{
 ## The R package 'contsimmap' is needed for this example to work.
 # Please install it manually from: https://github.com/bstaggmartin/contsimmap.

 ## Prepare data

 # Load eel data from the R package phytools
 # Source: Collar et al., 2014; DOI: 10.1038/ncomms6505

 library(phytools)
 data(eel.tree)
 data(eel.data)

 # Extract body size
 eel_data <- setNames(eel.data$Max_TL_cm,
                      rownames(eel.data))

  # (May take several minutes to run)
 ## Map continuous trait evolution on the phylogeny
 eel_cont_data  <- prepare_trait_data(
    tip_data = eel_data,
    trait_data_type = "continuous",
    phylo = eel.tree,
    # evolutionary_models = c("BM", "OU", "lambda", "kappa"),
    evolutionary_models = c("BM"),
    # Perform stochastic mapping to obtain multiple evolutionary histories
    run_stochastic_maps = TRUE,
    nb_simulations = 100,
```

```
   verbose = TRUE)

# Extract continuous stochastic maps (contMaps)
eel_contMaps <- eel_cont_data$contMaps

# Plot the interpolated map of ML estimates
plot_contMap(contMap = eel_cont_data$contMap)

# Plot several continuous stochastic maps (contMaps)
plot_contMap(contMap = eel_contMaps[[1]])
plot_contMap(contMap = eel_contMaps[[10]])
plot_contMap(contMap = eel_contMaps[[100]])

# Set focal time to 10 Mya
focal_time <- 10

## Extract all trait values for focal time = 10 Mya
extract_trait_data_10My <- extract_all_trait_values_for_focal_time(
   contMaps = eel_contMaps, trait_data_type = "continuous",
   focal_time = 10, update_Map = TRUE)

# Convert in data.frame
 # Rows = Stochastic maps
 # Columns = Cut branches at 10 Mya
trait_data_df <- as.data.frame(do.call(rbind, extract_trait_data_10My$trait_data))
head(trait_data_df)

## Plot updated contMaps

updated_contMaps_10My <- extract_trait_data_10My$contMaps
plot_contMap(contMap = updated_contMaps_10My[[1]])
plot_contMap(contMap = updated_contMaps_10My[[10]])
plot_contMap(contMap = updated_contMaps_10My[[100]])

}


# ----- Example 2: Categorical trait ----- #

## Load categorical trait data mapped on a phylogeny
data(eel_cat_3lvl_data, package = "deepSTRAPP")

# Explore data
str(eel_cat_3lvl_data, 1)
eel_cat_3lvl_data$densityMaps # Three density maps: one per state

# Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Extract trait data and update densityMaps for the given focal_time

# Extract from the densityMaps
```

```
eel_cat_3lvl_data_10My <- extract_all_trait_values_for_focal_time(
   densityMaps = eel_cat_3lvl_data$densityMaps,
   trait_data_type = "categorical",
   nb_simulations = 100,
   focal_time = focal_time,
   update_Map = TRUE)

## Print trait data
str(eel_cat_3lvl_data_10My, 1)
eel_cat_3lvl_data_10My$trait_data[1:2]

# Convert in data.frame
 # Rows = Dummy stochastic maps
 # Columns = Cut branches at 10 Mya
trait_data_df <- as.data.frame(do.call(rbind, eel_cat_3lvl_data_10My$trait_data))
trait_data_df[50:60, ]

# Distributions of states across dummy stochastic maps reflect posterior probabilities
# recorded in densityMaps, but they are not true evolutionary histories.
# If you want to relate trait states with a simulated evolutionary history,
# you need to provide simmaps as input.



# ----- Example 3: Biogeographic ranges ----- #

## Load biogeographic range data mapped on a phylogeny
data(eel_biogeo_data, package = "deepSTRAPP")

# Explore data
str(eel_biogeo_data, 1)
length(eel_biogeo_data$simmaps) # 100 stochastic maps: one per simulated biogeographic history

# Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Extract trait data and update simmaps for the given focal_time

# Extract from the simmaps
eel_biogeo_data_10My <- extract_all_trait_values_for_focal_time(
   simmaps = eel_biogeo_data$simmaps,
   trait_data_type = "biogeographic",
   focal_time = focal_time,
   update_Map = TRUE)

## Print trait data
str(eel_biogeo_data_10My, 1)
eel_biogeo_data_10My$trait_data[1:2]

# Convert in data.frame
 # Rows = Stochastic maps
 # Columns = Cut branches at 10 Mya
```

```
trait_data_df <- as.data.frame(do.call(rbind, eel_biogeo_data_10My$trait_data))
trait_data_df[1:10, ]

# Distributions of ranges are recorded across true stochastic maps
# as you provided simmaps as input.

## Plot updated stochastic maps

# Plot initial stochastic map n°1
plot(eel_biogeo_data$simmaps[[1]], fsize = 0.5)
abline(v = max(phytools::nodeHeights(eel_biogeo_data$simmaps[[1]])[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated stochastic map n°1, cut at 10 Mya
plot(eel_biogeo_data_10My$simmaps[[1]], fsize = 0.7)
```

`extract_diversification_data_melted_df_for_focal_time`
Extract diversification data from a BAMM_object

Description

Extracts regime IDs and tip rates from a `BAMM_object` that has been updated to provide diversification data for a specific time in the past (i.e. the `focal_time`). Use `update_rates_and_regimes_for_focal_time()` to obtain a `BAMM_object` updated for a given `focal_time`.

Usage

```
extract_diversification_data_melted_df_for_focal_time(
  BAMM_object,
  verbose = TRUE
)
```

Arguments

`BAMM_object`	Object of class "`bammdata`", typically generated with `update_rates_and_regimes_for_focal_time()`, that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples.
`verbose`	Logical. Should progression be displayed? A message will be printed for every batch of 100 BAMM posterior samples extracted. Default is `TRUE`.

Value

Returns a data.frame with six columns.

- `$focal_time` Integer. The time, in terms of time distance from the present, at which the trait data were extracted. Should be equal for all rows as a unique BAMM_object updated for a unique `focal_time` is being extracted.

- $BAMM_sample_ID Character string. ID of the posterior samples from which the diversification data are extracted named as 'BAMM_X' where X is the numerical ID.
- $tip_ID Character string. Tip labels of the branches cut-off at focal_time.
  - If keep_tip_labels = TRUE was used in update_rates_and_regimes_for_focal_time(), cut-off branches with a single descendant tip retain their initial tip.label.
  - If keep_tip_labels = FALSE was used in update_rates_and_regimes_for_focal_time(), all cut-off branches are labeled using their tipward node ID.
- $regime_ID Integer. The regime ID on tips at focal_time. IDs are integer. The root process equals "1", then they are incremented from oldest to youngest. Regime IDs are independent across posterior samples.
- $rate_type Character string. Type of rates: "lambda" for speciation rates, "mu" for extinction rates, and "net_diversification" for net diversification rates (lambda - mu).
- $rates Numeric. Rates in \[number of events / branch / evolutionary time\].

Author(s)

Maël Doré

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # Load the BAMM_object summarizing 1000 posterior samples of BAMM updated for a focal_time of 10 My
 data(Ponerinae_BAMM_object_10My)
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

  # (May take several minutes to run)
 # Extract diversification data
 diversification_data_df <- extract_diversification_data_melted_df_for_focal_time(
     BAMM_object = Ponerinae_BAMM_object_10My,
     verbose = TRUE)
 # Print output
 head(diversification_data_df)
}
```

---

extract_most_likely_trait_values_for_focal_time

Extract most likely trait data mapped on a phylogeny at a given time in the past

---

Description

Extracts the most likely trait values/states/ranges found along branches at a specific time in the past (i.e. the `focal_time`). Optionally, the function can update the mapped phylogeny (`contMap` or `densityMaps`) such as branches overlapping the `focal_time` are shortened to the `focal_time`, and the trait mapping for the cut off branches are removed by updating the `$tree$maps` and `$tree$mapped.edge` elements.

Usage

```
extract_most_likely_trait_values_for_focal_time(
  contMap = NULL,
  contMaps = NULL,
  densityMaps = NULL,
  simmaps = NULL,
  ace = NULL,
  tip_data = NULL,
  trait_data_type,
  focal_time,
  update_Map = FALSE,
  keep_tip_labels = TRUE
)
```

Arguments

`contMap`	For continuous trait data. Object of class `"contMap"`, typically generated with `prepare_trait_data()` or `phytools::contMap()`, that contains a phylogenetic tree and associated continuous trait mapping, representing interpolated ML estimates of ancestral trait values. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`contMaps`	For continuous trait data. List of objects of class `"contMap"`, typically generated with `prepare_trait_data()`, each containing a phylogenetic tree and associated continuous trait mapping that represents an independent evolutionary history of ancestral trait evolution, conditioned on observed trait data and model fit (i.e., stochastic maps).
`densityMaps`	For categorical trait or biogeographic data. List of objects of class `"densityMap"`, typically generated with `prepare_trait_data()`, that contains a phylogenetic tree and associated posterior probability of being in a given state/range along branches. Each object (i.e., `densityMap`) corresponds to a state/range. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
`simmaps`	For categorical trait or biogeographic data. List of objects of classes `"phylo"` and `"simmap"`, typically generated with `prepare_trait_data()` or `phytools::make.simmap()`, that represent discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches.
`ace`	(Optional) Ancestral Character Estimates (ACE) at the internal nodes. Obtained with `prepare_trait_data()` as output in the `$ace` slot.

- For continuous trait data: Named numeric vector typically generated with `phytools::fastAnc()`, `phytools::anc.ML()`, or `ape::ace()`. Names are nodes_ID of the internal nodes. Values are ACE of the trait.
- For categorical trait or biogeographic data: Matrix that records the posterior probabilities of ancestral states/ranges. Rows are internal nodes_ID. Columns are states/ranges. Values are posterior probabilities of each state per node. Needed in all cases to provide accurate estimates of trait values.

`tip_data` (Optional) Named vector of tip values of the trait.

- For continuous trait data: Named numeric vector of trait values.
- For categorical trait or biogeographic data: Character string vector of states/ranges Names are nodes_ID of the internal nodes. Needed to provide accurate tip values.

`trait_data_type`
Character string. Specify the type of trait data. Must be one of "continuous", "categorical", "biogeographic".

`focal_time` Integer. The time, in terms of time distance from the present, at which the tree and mapping must be cut. It must be smaller than the root age of the phylogeny.

`update_Map` Logical. Specify whether the mapped phylogeny (`contMap`, `densityMaps`, `simmaps`) provided as input should be updated for visualization and returned among the outputs. Default is `FALSE`. The update consists of cutting off branches and mapping that are younger than the `focal_time`.

`keep_tip_labels`
Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label` on the updated contMap. Default is `TRUE`. Used only if `update_Map = TRUE`.

Details

The mapped phylogeny (`contMap` or `densityMaps`) is cut at a specific time in the past (i.e. the `focal_time`) and the current trait values of the overlapping edges/branches are extracted.

—— Extract `trait_data` ——

For continuous trait data:

`contMap` is the default input representing the interpolated ML estimates of trait values along the phylogeny.

If providing only the `contMap` trait values at tips and internal nodes will be extracted from the mapping of the `contMap` leading to a slight discrepancy with the actual tip data and estimated ancestral character values.

True ML trait estimates will be used if `tip_data` and/or `ace` are provided as optional inputs. In practice the discrepancy is negligible.

Alternatively to a unique `contMap`, the user can provide a full set of continuous stochastic maps as `contMaps`. Each map then represents an independent evolutionary history of ancestral trait evolution, conditioned on observed trait data and model fit (i.e., stochastic maps). The 'most likely' trait values will be extracted as the mean of values observed across the stochastic maps. This quantity closely approximates the ML estimates as typically provided with a `contMap`.

For categorical trait and biogeographic data:

densityMaps are the default input. Most likely states/ranges are directly extracted from the posterior probabilities displayed in the densityMaps. The state/range with the highest probability is assigned to each tip and cut branches at focal_time.

True ML states/ranges will be used if tip_data and/or ace are provided as optional inputs. In practice the discrepancy is negligible.

Alternatively to densityMaps, the user can provide a full set of stochastic maps as simmaps. Each map then represents an independent evolutionary history of ancestral trait evolution, conditioned on observed trait data and model fit (i.e., stochastic maps). The 'most likely' states / ranges will be extracted as the most frequent state/range observed across the stochastic maps. This quantity is fully equivalent to what is derived from the densityMaps.

—— Update the contMap(s)/densityMaps/simmaps ——

To obtain updated contMap(s)/densityMaps/simmaps alongside the trait data, set update_Map = TRUE. The update consists of cutting off branches and mapping that are younger than the focal_time.

- When a branch with a single descendant tip is cut and keep_tip_labels = TRUE, the leaf left is labeled with the tip.label of the unique descendant tip.
- When a branch with a single descendant tip is cut and keep_tip_labels = FALSE, the leaf left is labeled with the node ID of the unique descendant tip.
- In all cases, when a branch with multiple descendant tips (i.e., a clade) is cut, the leaf left is labeled with the node ID of the MRCA of the cut-off clade.

The mapping in contMap(s)/densityMaps ($tree$maps and $tree$mapped.edge) or simmaps ($maps and $mapped.edge) is updated accordingly by removing mapping associated with the cut off branches.

To extract all trait data across multiple stochastic maps, and not just the most likely trait value/state/range, see this associated function: extract_all_trait_values_for_focal_time()

Value

By default, the function returns a list with three elements.

- $trait_data A named numeric vector with ML trait values found along branches overlapping the focal_time. Names are the tip.label/tipward node ID.
- $focal_time Integer. The time, in terms of time distance from the present, at which the trait data were extracted.
- $trait_data_type Character string. Define the type of trait data as "continuous", "categorical", or "biogeographic". Used in downstream analyses to select appropriate statistical processing.

If update_Map = TRUE, the output also contains updated mapped phylogenies as: $contMap and/or $contMaps or $densityMaps and/or $simmaps.

For continuous trait data:

- $contMap An object of class "contMap" that contains the updated contMap with branches and mapping that are younger than the focal_time cut off. The function also adds multiple useful sub-elements to the $contMap$tree element.
  - $root_age Integer. Stores the age of the root of the tree.

  - $nodes_ID_df Data.frame with two columns. Provides the conversion from the new_node_ID to the initial_node_ID. Each row is a node.
  - $initial_nodes_ID Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with ape::nodelabels().
  - $edges_ID_df Data.frame with two columns. Provides the conversion from the new_edge_ID to the initial_edge_ID. Each row is an edge/branch.
  - $initial_edges_ID Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with ape::edgelabels().
- $contMaps List of "contMap" objects as described above.

For categorical trait and biogeographic data:

- $densityMaps A list of objects of class "densityMap" that contains the updated densityMap of each state/range, with branches and mapping that are younger than the focal_time cut off. The function also adds multiple useful sub-elements to the $densityMaps$tree elements.
  - $root_age Integer. Stores the age of the root of the tree.
  - $nodes_ID_df Data.frame with two columns. Provides the conversion from the new_node_ID to the initial_node_ID. Each row is a node.
  - $initial_nodes_ID Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with ape::nodelabels().
  - $edges_ID_df Data.frame with two columns. Provides the conversion from the new_edge_ID to the initial_edge_ID. Each row is an edge/branch.
  - $initial_edges_ID Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with ape::edgelabels().
- $simmaps A list of objects of classes "phylo" and "simmap", that contains the updated stochastic maps simmap for each simulation of trait/range evolution. The same useful sub-elements as described above are also added to each simmap object.

Author(s)

Maël Doré

See Also

cut_phylo_for_focal_time() cut_contMap_for_focal_time() cut_densityMaps_for_focal_time()

Equivalent function to extract all data for multiple stochastic maps extract_all_trait_values_for_focal_time()

Examples

```
# ----- Example 1: Continuous trait ----- #

## Prepare data

# Load eel data from the R package phytools
# Source: Collar et al., 2014; DOI: 10.1038/ncomms6505

library(phytools)
data(eel.tree)
data(eel.data)
```

```
# Extract body size
eel_data <- setNames(eel.data$Max_TL_cm,
                     rownames(eel.data))

 # (May take several minutes to run)
## Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
eel_ACE <- phytools::fastAnc(tree = eel.tree, x = eel_data)

## Interpolate ML estimates of trait values based on a Brownian Motion model
# to interpolate values along branches and obtain a "contMap" object
eel_contMap <- phytools::contMap(eel.tree, x = eel_data,
                                 res = 100, # Number of time steps
                                 plot = FALSE)

# Set focal time to 50 Mya
focal_time <- 50

## Extract trait data and update contMap for the given focal_time

# Extract from the contMap (values are not exact ML estimates)
eel_cont_50 <- extract_most_likely_trait_values_for_focal_time(
   contMap = eel_contMap,
   trait_data_type = "continuous",
   focal_time = focal_time,
   update_Map = TRUE)
# Extract from tip data and ML estimates of ancestral characters (values are true ML estimates)
eel_cont_50 <- extract_most_likely_trait_values_for_focal_time(
   contMap = eel_contMap,
   ace = eel_ACE, tip_data = eel_data,
   trait_data_type = "continuous",
   focal_time = focal_time,
   update_Map = TRUE)

## Visualize outputs

# Print trait data
eel_cont_50$trait_data

# Plot node labels on initial stochastic map with cut-off
plot(eel_contMap, fsize = c(0.5, 1))
ape::nodelabels()
abline(v = max(phytools::nodeHeights(eel_contMap$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated contMap with initial node labels
plot(eel_cont_50$contMap)
ape::nodelabels(text = eel_cont_50$contMap$tree$initial_nodes_ID)


# ----- Example 2: Categorical trait ----- #
```

```
 # (May take several minutes to run)
## Load categorical trait data mapped on a phylogeny
data(eel_cat_3lvl_data, package = "deepSTRAPP")

# Explore data
str(eel_cat_3lvl_data, 1)
eel_cat_3lvl_data$densityMaps # Three density maps: one per state

# Set focal time to 10 Mya
focal_time <- 10

## Extract trait data and update densityMaps for the given focal_time

# Extract from the densityMaps
eel_cat_3lvl_data_10My <- extract_most_likely_trait_values_for_focal_time(
   densityMaps = eel_cat_3lvl_data$densityMaps,
   trait_data_type = "categorical",
   focal_time = focal_time,
   update_Map = TRUE)

## Print trait data
str(eel_cat_3lvl_data_10My, 1)
eel_cat_3lvl_data_10My$trait_data

## Plot density maps as overlay of all state posterior probabilities

# Plot initial density maps with ACE pies
plot_densityMaps_overlay(densityMaps = eel_cat_3lvl_data$densityMaps)
abline(v = max(phytools::nodeHeights(eel_cat_3lvl_data$densityMaps[[1]]$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated densityMaps with ACE pies
plot_densityMaps_overlay(eel_cat_3lvl_data_10My$densityMaps)


# ----- Example 3: Biogeographic ranges ----- #

 # (May take several minutes to run)
## Load biogeographic range data mapped on a phylogeny
data(eel_biogeo_data, package = "deepSTRAPP")

# Explore data
str(eel_biogeo_data, 1)
eel_biogeo_data$densityMaps # Two density maps: one per unique area: A, B.
eel_biogeo_data$densityMaps_all_ranges # Three density maps: one per range: A, B, and AB.

# Set focal time to 10 Mya
focal_time <- 10

## Extract trait data and update densityMaps for the given focal_time

# Extract from the densityMaps
eel_biogeo_data_10My <- extract_most_likely_trait_values_for_focal_time(
```

```
   densityMaps = eel_biogeo_data$densityMaps,
   # ace = eel_biogeo_data$ace,
   trait_data_type = "biogeographic",
   focal_time = focal_time,
   update_Map = TRUE)

## Print trait data
str(eel_biogeo_data_10My, 1)
eel_biogeo_data_10My$trait_data

## Plot density maps as overlay of all range posterior probabilities

# Plot initial density maps with ACE pies
plot_densityMaps_overlay(densityMaps = eel_biogeo_data$densityMaps)
abline(v = max(phytools::nodeHeights(eel_biogeo_data$densityMaps[[1]]$tree)[,2]) - focal_time,
       col = "red", lty = 2, lwd = 2)

# Plot updated densityMaps with ACE pies
plot_densityMaps_overlay(eel_biogeo_data_10My$densityMaps)
```

`extract_trait_data_melted_df_for_focal_time`

Extract trait data from a trait_data object

Description

Extract trait data from a trait_data_list object as produced within the deepSTRAPP workflow for a specific time in the past (i.e. the `focal_time`) and produce a melted dataset recording trait values across (stochastic maps x) edges for the associated `focal_time`.

Usage

```
extract_trait_data_melted_df_for_focal_time(trait_data_list)
```

Arguments

`trait_data_list`
: Object summarizing trait data extracted for a given `focal_time`, including a `$trait_data` element storing trait data recorded across (stochastic maps x) edges for the associated `$focal_time`. Use `extract_all_trait_values_for_focal_time()` to obtain trait data across multiple stochastic maps. Use `extract_most_likely_trait_values_for_foc` to obtain only the most likely trait data values.

Value

Returns a data.frame with five columns.

- `$focal_time` Integer. The time, in terms of time distance from the present, at which the trait data were extracted.

- $Map_ID Character string. ID of the stochastic map from which the trait data are extracted. If using 'rate_only' strategy to account for uncertainty in ancestral estimate, this is fixed to "Map_ML". If using 'paired' or 'full' strategies, this records either true stochastic maps "Map_X", or dummy maps "Dummy_map_X", depending on whether you provided respectively stochastic maps (contMaps/simmaps) or simply densityMaps as inputs.
- $tip_ID Character string. Tip labels of the branches cut-off at focal_time.
  - If keep_tip_labels = TRUE was used in update_rates_and_regimes_for_focal_time(), cut-off branches with a single descendant tip retain their initial tip.label.
  - If keep_tip_labels = FALSE was used in update_rates_and_regimes_for_focal_time(), all cut-off branches are labeled using their tipward node ID.
- $trait_data_type Character string. Records the type of traits: "continuous", "categorical", or "biogeographic".
- $trait_value Numerical or Character string.
  - For "continuous" traits: numerical values.
  - For "categorical" traits: states.
  - For "biogeographic" data: ranges.

Author(s)

Maël Doré

Examples

```
# ----- Example 1: Extract ML estimates data ----- #

## Load categorical trait data mapped on a phylogeny
data(eel_cat_3lvl_data, package = "deepSTRAPP")

# Explore data
str(eel_cat_3lvl_data, 1)
eel_cat_3lvl_data$densityMaps # Three density maps: one per state

# Set focal time to 10 Mya
focal_time <- 10

## Extract ML estimates of ancestral states for the given focal_time

# Extract from the densityMaps
eel_cat_3lvl_data_10My <- extract_most_likely_trait_values_for_focal_time(
   densityMaps = eel_cat_3lvl_data$densityMaps,
   trait_data_type = "categorical",
   focal_time = focal_time)

## Format ML states data as a melted df

melted_df <- extract_trait_data_melted_df_for_focal_time(trait_data_list = eel_cat_3lvl_data_10My)
head(melted_df)


# ----- Example 2: Extract data across multiple stochastic maps ----- #
```

```
## Load biogeographic range data mapped on a phylogeny
data(eel_biogeo_data, package = "deepSTRAPP")

# Explore data
str(eel_biogeo_data, 1)
length(eel_biogeo_data$simmaps) # 100 stochastic maps: one per simulated biogeographic history

# Set focal time to 10 Mya
focal_time <- 10

## Extract range data across stochastic maps for the given focal_time

# Extract from the simmaps
eel_biogeo_data_10My <- extract_all_trait_values_for_focal_time(
   simmaps = eel_biogeo_data$simmaps,
   trait_data_type = "biogeographic",
   focal_time = focal_time)

## Format range data as a melted df

melted_df <- extract_trait_data_melted_df_for_focal_time(trait_data_list = eel_biogeo_data_10My)
head(melted_df)
```

`mammals` *Phylogeny and body mass data for extant and extinct mammal families/genera from Slater, 2013*

Description

A list containing two elements:

- `$mammal.mass` A data.frame with mean and standard error of mammal body masses in ln(mass in g).
- `$mammal.phy` A time-calibrated phylogeny of extinct and extant mammal families/genera.

Source: Slater, G. J. (2013). Phylogenetic evidence for a shift in the mode of mammalian body size evolution at the Cretaceous-Palaeogene boundary. Methods in Ecology and Evolution, 4(8), 734-744. doi:10.1111/2041210X.12084

Usage

```
data(mammals)
```

Format

A list with 2 elements.

- `$mammal.mass` A data.frame of 213 observations and two columns. Values are numerical.
- `$mammal.phy` A time-calibrated phylogeny of 211 tips.

Details

Initial dataset from Slater, G. J. (2013). The R object was imported from R package `motmot` to include in deepSTRAPP.

`plot_BAMM_rates` *Plot diversification rates and regime shifts from BAMM on phylogeny*

Description

Plot on a time-calibrated phylogeny the evolution of diversification rates and the location of regime shifts estimated from a BAMM (Bayesian Analysis of Macroevolutionary Mixtures). Each branch is colored according to the estimated rates of speciation, extinction, or net diversification stored in an object of class `"bammdata"`. Rates can vary along time, thus colors evolve along individual branches.

This function is a wrapper of original functions from the R package {`BAMMtools`}:

- Step 1: Use `BAMMtools::plot.bammdata()` to map rates on the phylogeny.
- Step 2: Add the location of regime shifts with `BAMMtools::addBAMMshifts()` (if `add_regime_shifts = TRUE`).

Usage

```
plot_BAMM_rates(
  BAMM_object,
  rate_type = "net_diversification",
  method = "phylogram",
  configuration_type = "MAP",
  sample_index = 1,
  regimes_fill = "grey",
  regimes_size = 1,
  regimes_pch = 21,
  regimes_border_col = "black",
  regimes_border_width = 1,
  ...,
  add_regime_shifts = TRUE,
  adjust_size_to_prob = TRUE,
  display_plot = TRUE,
  PDF_file_path = NULL
)
```

Arguments

`BAMM_object`	Object of class `"bammdata"`, typically generated with `prepare_diversification_data()`, that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples. It works also for `BAMM_object` updated for a specific `focal_time` using `update_rates_and_regimes_for_focal_time()`, or the deepSTRAPP workflow with run_deepSTRAPP_for_focal_time and `run_deepSTRAPP_over_time`

`rate_type` A character string specifying the type of diversification rates to plot. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).

`method` A character string indicating the method for plotting the phylogenetic tree.

- `method = "phylogram"` (default) plots the phylogenetic tree using rectangular coordinates.
- `method = "polar"` plots the phylogenetic tree using polar coordinates.

`configuration_type` A character string specifying how to select the location of regime shifts across posterior samples.

- `configuration_type = "MAP"`: Use the average locations recorded in posterior samples with the Maximum A Posteriori probability (MAP) configuration. This regime shift configuration is the most frequent configuration among the posterior samples (See `BAMMtools::getBestShiftConfiguration()`). This is the default option.
- `configuration_type = "MSC"`: Use the average locations recorded in posterior samples with the Maximum Shift Credibility (MSC) configuration. This regime shift configuration has the highest product of marginal probabilities across branches (See `BAMMtools::maximumShiftCredibility()`).
- `configuration_type = "index"`: Use the configuration of a unique posterior sample whose index is provided in `sample_index`.

`sample_index` Integer. Index of the posterior samples to use to plot the location of regime shifts. Used only if `configuration_type = index`. Default = 1.

`regimes_fill` Character string. Set the color of the background of the symbols showing the location of regime shifts. Equivalent to the bg argument in `BAMMtools::addBAMMshifts()`. Default is `"grey"`.

`regimes_size` Numeric. Set the size of the symbols showing the location of regime shifts. Equivalent to the cex argument in `BAMMtools::addBAMMshifts()`. Default is 1.

`regimes_pch` Integer. Set the shape of the symbols showing the location of regime shifts. Equivalent to the pch argument in `BAMMtools::addBAMMshifts()`. Default is 21.

`regimes_border_col` Character string. Set the color of the border of the symbols showing the location of regime shifts. Equivalent to the col argument in `BAMMtools::addBAMMshifts()`. Default is `"black"`.

`regimes_border_width` Numeric. Set the width of the border of the symbols showing the location of regime shifts. Equivalent to the lwd argument in `BAMMtools::addBAMMshifts()`. Default is 1.

`...` Additional graphical arguments to pass down to `BAMMtools::plot.bammdata()`, `BAMMtools::addBAMMshifts()`, and `par()`. Among them, `par.reset` is ignored, with a message: it would make `BAMMtools::plot.bammdata()` restore a full par() snapshot, discarding the coordinate system of the phylogeny and rewinding the panel counter of a multi-panel layout. Graphical parameters are instead restored by `plot_BAMM_rates()` when it exits: every parameter the

call actually changed is put back, except those describing the plot itself, which are kept so that the phylogeny can be annotated afterwards with for instance `graphics::abline()` or `graphics::title()`, as in the examples.

`add_regime_shifts`
: Logical. Whether to add the location of regime shifts on the phylogeny (Step 2). Default is `TRUE`. Provide the full argument name to avoid ambiguity with the `add` graphical argument.

`adjust_size_to_prob`
: Logical. Whether to scale the size of the symbols showing the location of regime shifts according to the marginal shift probability of the shift happening on each location/branch. This will only work if there is an `$MSP_tree` element summarizing the marginal shift probabilities across branches in the `BAMM_object`. Default is `TRUE`. Provide the full argument name to avoid ambiguity with the `adj` graphical argument.

`display_plot`
: Logical. Whether to display the plot generated in the R console. Default is `TRUE`.

`PDF_file_path`
: Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

The main input `BAMM_object` is the typical output of `prepare_diversification_data()`. It provides information on rates and regime shifts across the posterior samples of a BAMM.
`$MAP_BAMM_object` and `$MSC_BAMM_object` elements are required to plot regime shift locations following the "MAP" or "MSC" `configuration_type` respectively. A `$MSP_tree` element is required to scale the size of the symbols showing the location of regime shifts according to marginal shift probabilities. (If `adjust_size_to_prob = TRUE`).

The default option to display regime shift is to use the average locations from the posterior samples with the Maximum A Posteriori probability (MAP) configuration. However, sometimes, multiple configurations have similarly high frequency in the posterior samples (See `BAMMtools::credibleShiftSet()`). An alternative is to use the average locations from posterior samples with the Maximum Shift Credibility (MSC) configuration instead. This regime shift configuration has the highest product of marginal probabilities across branches where a shift is estimated. It may differ from the MAP configuration. (See `BAMMtools::maximumShiftCredibility()`).

Value

The function returns (invisibly) a list with three elements similarly to `BAMMtools::plot.bammdata()`.

- `$coords`: A matrix of plot coordinates. Rows correspond to branches. Columns 1-2 are starting (x,y) coordinates of each branch and columns 3-4 are ending (x,y) coordinates of each branch. If method = "polar" a fifth column gives the angle (in radians) of each branch.
- `$colorbreaks`: A vector of percentiles used to group macroevolutionary rates into color bins.
- `$colordens`: A matrix of the kernel density estimates (column 2) of evolutionary rates (column 1) and the color (column 3) corresponding to each rate value.

Author(s)

Maël Doré

Original functions by Mike Grundler & Pascal Title in R package `{BAMMtools}`.

See Also

Initial functions in BAMMtools: BAMMtools::plot.bammdata() BAMMtools::addBAMMshifts()

Associated functions in deepSTRAPP: prepare_diversification_data() update_rates_and_regimes_for_focal_tim
run_deepSTRAPP_for_focal_time() run_deepSTRAPP_over_time()

Examples

```
# Load BAMM output
data(whale_BAMM_object, package = "deepSTRAPP")

mfrow_ini <- par()$mfrow
par(mfrow = c(1,2))

## Plot overall mean rates with MAP configuration for regime shifts
# (rates are averaged across all posterior samples)
plot_BAMM_rates(whale_BAMM_object, # Use the full object to plot mean overall rates
                add_regime_shifts = TRUE,
                configuration_type = "MAP",
                regimes_size = 3, bg = "black")
title("Overall mean - MAP shift config", col.main = "white")

## Plot overall mean rates with MSC configuration for regime shifts
# (rates are averaged across all posterior samples)
plot_BAMM_rates(whale_BAMM_object, add_regime_shifts = TRUE,
                configuration_type = "MSC",
                regimes_size = 3, bg = "black")
title("Overall mean - MSC shift config", col.main = "white")

par(mfrow_ini)


mfrow_ini <- par()$mfrow
par(mfrow = c(1,2))

## Plot mean MAP rates with regime shifts
# (rates are averaged only across MAP samples)
plot_BAMM_rates(whale_BAMM_object$MAP_BAMM_object, # Use the MAP object to plot mean MAP rates
                add_regime_shifts = TRUE,
                configuration_type = "index",
                # Set to index to use the regime shift location from MAP configuration
                regimes_size = 3)
title("MAP mean - MAP shift config")

## Plot mean MSC rates with regime shifts
# (rates averaged only across MSC samples)
plot_BAMM_rates(whale_BAMM_object$MSC_BAMM_object, # Use the MSC object to plot mean MSC rates
                add_regime_shifts = TRUE,
                configuration_type = "index",
                # Set to index to use the regime shift data from MSC configuration
                regimes_size = 3)
title("MSC mean - MSC shift config")
```

```
par(mfrow_ini)
```

`plot_contMap` *Plot continuous trait evolution on the tree*

Description

Plot on a time-calibrated phylogeny the evolution of a continuous trait as summarized in a `contMap` object typically generated with `prepare_trait_data()`.

This function is a wrapper of original functions from the R package `{phytools}`:

- Step 1: Use `phytools::setMap()` to update the color scale if requested.
- Step 2: Use `phytools::plot.contMap()` to plot the mapped phylogeny.

Usage

```
plot_contMap(
  contMap,
  ...,
  color_scale = NULL,
  display_plot = TRUE,
  PDF_file_path = NULL
)
```

Arguments

`contMap`	List of class `"contMap"`, typically generated with `prepare_trait_data()`, that contains a phylogenetic tree and associated ancestral trait estimates mapped along branches in `contMap$tree$maps`.
`...`	Additional arguments to pass down to `phytools::plot.contMap()` to control plotting.
`color_scale`	Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` showing the evolution of a continuous trait. From lowest values to highest values. Default (`color_scale = NULL`) is using the color palette recorded in the `contMap$cols` item. If none was provided, the `rainbow()` palette is used. Provide the full argument name to avoid ambiguity with the `col` graphical argument.
`display_plot`	Logical. Whether to display the plot generated in the R console. Default is `TRUE`.
`PDF_file_path`	Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

This function is a wrapper of original functions `phytools::setMap()` and `phytools::plot.contMap()`. Additions are listed below:

- The color scale can be controlled directly with the argument `color_scale`.
- The plot can be exported in PDF using `PDF_file_path` to define the output file.

Value

If `display_plot = TRUE`, the function plots the time-calibrated phylogeny displaying the evolution of a continuous trait. If `PDF_file_path` is provided, the function exports the plot into a PDF file.

An object of class `"contMap"` with an (optionally) updated color scale (`$cols`) is returned invisibly.

Author(s)

Maël Doré

Original functions by Liam Revell in R package {phytools}. Contact: <liam.revell@umb.edu>

See Also

`phytools::plot.contMap()` `plot_densityMaps_overlay()`

Examples

```
# Load phylogeny
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_tree, package = "deepSTRAPP")

## Prepare trait data

 # (May take several minutes to run)
# Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

# Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree, x = Ponerinae_cont_tip_data)

# Infer Ancestral Character Estimates based on a Brownian Motion model
# and run a Stochastic Mapping to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(Ponerinae_tree, x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)

# Plot contMap with the phytools method
plot(x = Ponerinae_contMap, fsize = c(0.5, 1))

# Plot contMap with an updated color scale
plot_contMap(contMap = Ponerinae_contMap, fsize = c(0.5, 1),
             color_scale = c("darkgreen", "limegreen", "orange", "red"))
             # PDF_file_path = "Ponerinae_contMap.pdf")
```

plot_densityMaps_overlay *Plot posterior probabilities of states/ranges on phylogeny from densityMaps*

Description

Plot on a time-calibrated phylogeny the evolution of a categorical trait/biogeographic ranges summarized from densityMaps typically generated with prepare_trait_data(). Each branch is colored according to the posterior probability of being in a given state/range. Colors for each state/range are overlaid using transparency to produce a single plot for all states/ranges.

Usage

```
plot_densityMaps_overlay(
  densityMaps,
  ace = NULL,
  ...,
  colors_per_levels = NULL,
  add_ACE_pies = TRUE,
  cex_pies = 0.5,
  display_plot = TRUE,
  PDF_file_path = NULL
)
```

Arguments

densityMaps	List of objects of class "densityMap", typically generated with prepare_trait_data(), that contains a phylogenetic tree and associated posterior probability of being in a given state/range along branches. Each object (i.e., densityMap) corresponds to a state/range. If no color is provided for multi-area ranges, they will be interpolated.
ace	Numerical matrix. To provide the posterior probabilities of ancestral states/ranges (characters) estimates (ACE) at internal nodes used to plot the ACE pies. Rows are internal nodes. Columns are states/ranges. Values are posterior probabilities of each state per node. Typically generated with prepare_trait_data() in the $ace slot. If NULL (default), the ACE are extracted from the densityMaps with a possible slight discrepancy with the actual tip states and estimated posterior probabilities of ancestral states.
...	Additional arguments to pass down to phytools::plotSimmap() to control plotting.
colors_per_levels	Named character string. To set the colors to use to map each state/range posterior probabilities. Names = states/ranges; values = colors. If NULL (default), the color scale provided in the densityMaps will be used. Provide the full argument name to avoid ambiguity with the col graphical argument.

add_ACE_pies	Logical. Whether to add pies of posterior probabilities of states/ranges at internal nodes on the mapped phylogeny. Default = TRUE. Provide the full argument name to avoid ambiguity with the add graphical argument.
cex_pies	Numeric. To adjust the size of the ACE pies. Default = 0.5. Provide the full argument name to avoid ambiguity with the cex graphical argument.
display_plot	Logical. Whether to display the plot generated in the R console. Default is TRUE.
PDF_file_path	Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Value

If display_plot = TRUE, the function plots a time-calibrated phylogeny displaying the evolution of a categorical trait/biogeographic ranges. If PDF_file_path is provided, the function exports the plot into a PDF file.

Author(s)

Maël Doré

Original functions by Liam Revell in R package {phytools}. Contact: <liam.revell@umb.edu>

See Also

phytools::plot.densityMap() phytools::plotSimmap()

Examples

```
# Load phylogeny and tip data
library(phytools)
data(eel.tree)
data(eel.data)

# Transform feeding mode data into a 3-level factor
eel_data <- stats::setNames(eel.data$feed_mode, rownames(eel.data))
eel_data <- as.character(eel_data)
eel_data[c(1, 5, 6, 7, 10, 11, 15, 16, 17, 24, 25, 28, 30, 51, 52, 53, 55, 58, 60)] <- "kiss"
eel_data <- stats::setNames(eel_data, rownames(eel.data))
table(eel_data)

# Manually define a Q_matrix for rate classes of state transition to use in the 'matrix' model
# Does not allow transitions from state 1 ("bite") to state 2 ("kiss") or state 3 ("suction")
# Does not allow transitions from state 3 ("suction") to state 1 ("bite")
# Set symmetrical rates between state 2 ("kiss") and state 3 ("suction")
Q_matrix = rbind(c(NA, 0, 0), c(1, NA, 2), c(0, 2, NA))

# Set colors per state
colors_per_levels <- c("limegreen", "orange", "dodgerblue")
names(colors_per_levels) <- c("bite", "kiss", "suction")

 # (May take several minutes to run)
# Run evolutionary models to prepare trait data
```

```
eel_cat_3lvl_data <- prepare_trait_data(tip_data = eel_data, phylo = eel.tree,
    trait_data_type = "categorical",
    colors_per_levels = colors_per_levels,
    evolutionary_models = c("ER", "SYM", "ARD", "meristic", "matrix"),
    Q_matrix = Q_matrix,
    nb_simulations = 1000,
    plot_map = TRUE,
    plot_overlay = TRUE,
    return_best_model_fit = TRUE,
    return_model_selection_df = TRUE)

# Load directly output
data(eel_cat_3lvl_data, package = "deepSTRAPP")

# Plot densityMaps one by one
plot(eel_cat_3lvl_data$densityMaps[[1]]) # densityMap for state n°1 ("bite")
plot(eel_cat_3lvl_data$densityMaps[[2]]) # densityMap for state n°1 ("kiss")
plot(eel_cat_3lvl_data$densityMaps[[3]]) # densityMap for state n°1 ("suction")

# Plot overlay of all densityMaps
plot_densityMaps_overlay(densityMaps = eel_cat_3lvl_data$densityMaps)
```

`plot_histograms_STRAPP_tests_over_time` *Plot multiple histograms of STRAPP test statistics over time-steps*

Description

Plot a histogram of the distribution of the test statistics obtained from a deepSTRAPP workflow carried out for each focal time in `$time_steps`. Main input = output of a deepSTRAPP run over time using `run_deepSTRAPP_over_time()`).

Returns one histogram for overall tests for each focal time in `$time_steps`. If `plot_posthoc_tests = TRUE`, it will return one faceted plot with a histogram per post hoc test for each focal time in `$time_steps`.

If a PDF file path is provided in `PDF_file_path`, the plots will be saved directly in a PDF file, with one page per focal time in `$time_steps`.

Usage

```
plot_histograms_STRAPP_tests_over_time(
  deepSTRAPP_outputs,
  display_plots = TRUE,
  plot_posthoc_tests = FALSE,
  PDF_file_path = NULL
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_over_time()`, that summarize the results of multiple deepSTRAPP runs across `$time_steps`. It needs to include the `$STRAPP_results_over_time` element with `$perm_data_df` obtained when setting both `return_STRAPP_results = TRUE` and `return_perm_data = TRUE`.

`display_plots`
: Logical. Whether to display the histograms generated in the R console. Default is `TRUE`.

`plot_posthoc_tests`
: Logical. For multinomial data only. Whether to plot the histograms for the overall Kruskal-Wallis test across all states (`plot_posthoc_tests = FALSE`), or plot the histograms for all the pairwise post hoc Dunn's tests across pairs of states (`plot_posthoc_tests = TRUE`). Time-steps at which the data does not yield more than two states/ranges will show a warning and generate no plot. Default is `FALSE`.

`PDF_file_path`
: Character string. If provided, the plots will be saved in a unique PDF file following the path provided here. The path must end with ".pdf". Each page of the PDF corresponds to a focal time in `$time_steps`.

Details

The main input `deepSTRAPP_outputs` is the typical output of `run_deepSTRAPP_over_time()`. It provides information on results of STRAPP tests performed over multiple time-steps.

Histograms are built based on the distribution of the test statistics. Such distributions are recorded in the outputs of a deepSTRAPP run carried out with `run_deepSTRAPP_over_time()` when `return_STRAPP_results = TRUE` AND `return_perm_data = TRUE`. The `$STRAPP_results_over_time` objects provided within the input are lists that must contain a `$perm_data_df` element that summarizes test statistics computed across posterior samples.

For multinomial data (categorical or biogeographic data with more than 2 states), it is possible to plot the histograms of post hoc pairwise tests. Set `plot_posthoc_tests = TRUE` to generate histograms for all the pairwise post hoc Dunn's tests across pairs of states. To achieve this, the `$STRAPP_results_over_time` objects must contain a `$posthoc_pairwise_tests$perm_data_array` element that summarizes test statistics computed across posterior samples for all pairwise post hoc tests. This is obtained from `run_deepSTRAPP_over_time()` when setting `return_STRAPP_results = TRUE` to return the STRAPP results, `posthoc_pairwise_tests = TRUE` to carry out post hoc tests, and `return_perm_data = TRUE` to record distributions of test statistics. Time-steps for which the data do not yield more than two states/ranges will show a warning and generate no plot.

Value

By default, the function returns a list of sub-lists of classes `gg` and `ggplot` ordered as in `$time_steps`. Each sub-list corresponds to a ggplot for a given `focal_time i` that can be displayed on the console with `print(output[[i]])`. If `display_plots = TRUE`, the histograms are being displayed on the console one by one while generated.

If using multinomial data and set `plot_posthoc_tests = TRUE`, the function will return a list of sub-lists of objects ordered as in `$time_steps`. Each sub-list is a list of the ggplots associated

with pairwise post hoc tests carried out for a given `focal_time`. For a given `focal_time` i, to plot each histogram j individually, use `print(output_list[[i]][[j]])`. To plot all histograms of a given `focal_time` `i` at once in a multifaceted plot, as displayed sequentially on the console if `display_plots` = `TRUE`, use `cowplot::plot_grid(plotlist = output_list[[i]])`.

Each plot also displays summary statistics for the STRAPP test associated with the data displayed.

- QX% = The quantile of null statistic distribution at the significance threshold used to define test significance. This is the value found on the red dashed line. The test will be considered significant (i.e., the null hypothesis is rejected) if this value is higher than zero (the black dashed line).
- P-value = The p-value of the STRAPP test which corresponds to the proportion of cases in which the statistic was lower than expected under the null hypothesis (i.e., the proportion of the histogram found below / on the left-side of the black dashed line).
- N = The number of test statistics in the null distribution.

If a `PDF_file_path` is provided, the function will also generate a PDF file of the plots with one page per `$time_steps`. For post hoc tests, this will save the multifaceted plots.

Author(s)

Maël Doré

See Also

Associated functions in deepSTRAPP: `run_deepSTRAPP_over_time()` `plot_histogram_STRAPP_test_for_focal_time(`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #

 # Load fake trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare trait data

 # Extract continuous trait data as a named vector
 Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                     nm = Ponerinae_trait_tip_data$Taxa)

 # Select a color scheme from lowest to highest values
 color_scale = c("darkgreen", "limegreen", "orange", "red")
```

```
# Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

 # (May take several minutes to run)
# Run a Stochastic Mapping based on a Brownian Motion model
# to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(tree = Ponerinae_tree_old_calib,
                                       x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
# Plot contMap = stochastic mapping of continuous trait
plot_contMap(contMap = Ponerinae_contMap,
             color_scale = color_scale)

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

## Run deepSTRAPP on net diversification rates
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- run_deepSTRAPP_over_time(
   contMap = Ponerinae_contMap,
   ace = Ponerinae_ACE,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "rates_only",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly trait data output
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Plot histograms of STRAPP overall test results
# Tests are Spearman's rank correlation tests

# Plot all histograms
histogram_ggplots <- plot_histograms_STRAPP_tests_over_time(
```

```
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
   display_plots = TRUE,
   # PDF_file_path = "./plot_STRAPP_histogram_overall_test.pdf",
   plot_posthoc_tests = FALSE)

# Print histogram for time step 1 = 0 My
print(histogram_ggplots[[1]])
# Adjust aesthetics of plot for time step 1 a posteriori
histogram_ggplot_adj <- histogram_ggplots[[1]] +
   ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(histogram_ggplot_adj)

# ----- Example 2: Categorical data ----- #

## Load data

# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
## Produce densityMaps using stochastic character mapping based on an ARD Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
   tip_data = Ponerinae_cat_3lvl_tip_data,
   phylo = Ponerinae_tree_old_calib,
   trait_data_type = "categorical",
   colors_per_levels = colors_per_states,
   evolutionary_models = "ARD",
   nb_simulations = 100,
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   plot_map = FALSE)

# Load directly trait data output
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")
```

```
## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates across time-steps.
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- run_deepSTRAPP_over_time(
   densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
   ace = Ponerinae_cat_3lvl_data_old_calib$ace,
   tip_data = Ponerinae_cat_3lvl_tip_data,
   nb_simulations = 100,
   trait_data_type = "categorical",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   rate_type = "net_diversification",
   uncertainty_strategy = "paired",
   seed = 1234, # Set for reproducibility
   alpha = 0.10, # Select a generous level of significance for the sake of the example
   posthoc_pairwise_tests = TRUE,
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly deepSTRAPP output
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Explore output
str(Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40, max.level = 1)

## Plot histograms of STRAPP overall test results #
# Tests are Kruskall-Wallis H tests when more than two states/ranges are present.
# Tests are Mann-Whitney-Wilcoxon rank-sum tests when only two states/ranges are present.

histogram_ggplots <- plot_histograms_STRAPP_tests_over_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
   display_plots = TRUE,
   # PDF_file_path = "./plot_STRAPP_histograms_overall_tests.pdf",
   plot_posthoc_tests = FALSE)

# Print histogram for time step 1 = 0 My
```

```
print(histogram_ggplots[[1]])
# Adjust aesthetics of plot for time step 1 a posteriori
histogram_ggplot_adj <- histogram_ggplots[[1]] +
    ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(histogram_ggplot_adj)

## Plot histograms of STRAPP post hoc test results ------ #
# Tests are Dunn's multiple comparison pairwise post hoc tests possible
# only when more than two states/ranges are present.

histograms_ggplots_list <- plot_histograms_STRAPP_tests_over_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
    display_plots = TRUE,
    # PDF_file_path = "./plot_STRAPP_histograms_posthoc_tests.pdf",
    plot_posthoc_tests = TRUE)

# Print all histograms for time step 1 (= 0 My) one by one
print(histograms_ggplots_list[[1]])
# Plot all histograms for time step 1 (= 0 My) on one faceted plot
cowplot::plot_grid(plotlist = histograms_ggplots_list[[1]])
}
```

`plot_histogram_STRAPP_test_for_focal_time` *Plot histogram of STRAPP test statistics to assess results*

Description

Plot a histogram of the distribution of the test statistics obtained from a STRAPP test carried out for a unique `focal_time`. (See `compute_STRAPP_test_for_focal_time()` and `run_deepSTRAPP_for_focal_time()`).

Returns a single histogram for overall tests. If `plot_posthoc_tests = TRUE`, it will return a faceted plot with a histogram per post hoc test.

If a PDF file path is provided in `PDF_file_path`, the plot will be saved directly in a PDF file.

Usage

```
plot_histogram_STRAPP_test_for_focal_time(
  deepSTRAPP_outputs,
  focal_time = NULL,
  display_plot = TRUE,
  plot_posthoc_tests = FALSE,
  PDF_file_path = NULL
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_for_focal_time()`, that summarize the results of a STRAPP test for a specific time in the past (i.e. the `focal_time`). `deepSTRAPP_outputs` can also be extracted from the output of `run_deepSTRAPP_over_time()` that run the whole deepSTRAPP workflow and store results inside `$STRAPP_results_over_time`.

`focal_time`
: Numeric. (Optional) If `deepSTRAPP_outputs` comprises results over multiple time-steps (i.e., output of `run_deepSTRAPP_over_time()`, this is the time of the STRAPP test targeted for plotting.

`display_plot`
: Logical. Whether to display the histogram(s) generated in the R console. Default is `TRUE`.

`plot_posthoc_tests`
: Logical. For multinomial data only. Whether to plot the histogram for the overall Kruskal-Wallis test across all states (`plot_posthoc_tests = FALSE`), or plot the histograms for all the pairwise post hoc Dunn's tests across pairs of states (`plot_posthoc_tests = TRUE`). Default is `FALSE`.

`PDF_file_path`
: Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

The main input `deepSTRAPP_outputs` is the typical output of `run_deepSTRAPP_for_focal_time()`. It provides information on results of a STRAPP test performed at a given `focal_time`.

Histograms are built based on the distribution of the test statistics. Such distributions are recorded in the outputs of a deepSTRAPP run carried out with `run_deepSTRAPP_for_focal_time()` when `return_perm_data = TRUE` so that the distributions of test stats computed across posterior samples are returned among the outputs under `$STRAPP_results$perm_data_df`.

For multinomial data (categorical or biogeographic data with more than 2 states), it is possible to plot the histograms of post hoc pairwise tests. Set `plot_posthoc_tests = TRUE` to generate histograms for all the pairwise post hoc Dunn's tests across pairs of states. To achieve this, the `deepSTRAPP_outputs` input object must contain a `$STRAPP_results$posthoc_pairwise_tests$perm_data_array` element that summarizes test statistics computed across posterior samples for all pairwise post hoc tests. This is obtained from `run_deepSTRAPP_for_focal_time()` when setting both `posthoc_pairwise_tests = TRUE` to carry out post hoc tests, and `return_perm_data = TRUE` to record distributions of test statistics.

Alternatively, the main input `deepSTRAPP_outputs` can be the output of `run_deepSTRAPP_over_time()`, providing results of STRAPP tests over multiple time-steps. In this case, you must provide a `focal_time` to select the unique time-step used for plotting.

- `return_perm_data` must be set to `TRUE` so that the permutation data used to compute the tests are returned among the outputs under `$STRAPP_results_over_time[[i]]$perm_data_df`.
- `posthoc_pairwise_tests` must be set to `TRUE` so that the permutation data used to perform the post hoc tests are also returned among the outputs under `$STRAPP_results_over_time[[i]]$posthoc_pairwise_`

For plotting all time-steps at once, see `plot_histograms_STRAPP_tests_over_time()`.

Value

By default, the function returns a list of classes gg and ggplot. This object is a ggplot that can be displayed on the console with `print(output)`. It corresponds to the histogram being displayed on the console when the function is run, if `display_plot = TRUE`, and can be further modify for aesthetics using the ggplot2 grammar.

If using multinomial data and setting `plot_posthoc_tests = TRUE`, the function will return a list of objects. Each object is the ggplot associated with a pairwise post hoc test. To plot each histogram `i` individually, use `print(output_list[[i]])`. To plot all histograms at once in a multifaceted plot, as displayed on the console if `display_plot = TRUE`, use `cowplot::plot_grid(plotlist = output_list)`.

Each plot also displays summary statistics for the STRAPP test associated with the data displayed.

- QX% = The quantile of null statistic distribution at the significance threshold used to define test significance. This is the value found on the red dashed line. The test will be considered significant (i.e., the null hypothesis is rejected) if this value is higher than zero (the black dashed line).
- P-value = The p-value of the STRAPP test which corresponds to the proportion of cases in which the statistic was lower than expected under the null hypothesis (i.e., the proportion of the histogram found below / on the left-side of the black dashed line).
- N = The number of test statistics in the null distribution.

If a `PDF_file_path` is provided, the function will also generate a PDF file of the plot. For post hoc tests, this will save the multifaceted plot.

Author(s)

Maël Doré

See Also

Associated functions in deepSTRAPP: `run_deepSTRAPP_for_focal_time()` `plot_histograms_STRAPP_tests_over_tim`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #

 # Load fake trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.
```

```
## Prepare trait data

# Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

# Select a color scheme from lowest to highest values
color_scale = c("darkgreen", "limegreen", "orange", "red")

# Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

 # (May take several minutes to run)
# Run a Stochastic Mapping based on a Brownian Motion model
# to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(tree = Ponerinae_tree_old_calib,
                                       x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
# Plot contMap = stochastic mapping of continuous trait
plot_contMap(contMap = Ponerinae_contMap,
             color_scale = color_scale)

## Set focal time to 10 Mya
focal_time <- 10

## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_cont_old_calib_10My <- run_deepSTRAPP_for_focal_time(
  contMap = Ponerinae_contMap,
  ace = Ponerinae_ACE,
  tip_data = Ponerinae_cont_tip_data,
  trait_data_type = "continuous",
  BAMM_object = Ponerinae_BAMM_object_old_calib,
  focal_time = focal_time,
  rate_type = "net_diversification",
  uncertainty_strategy = "rates_only",
  return_perm_data = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_cont_old_calib_10My, max.level = 1)

# ----- Plot histogram of STRAPP overall test results from run_deepSTRAPP_for_focal_time() ----- #

histogram_ggplot <- plot_histogram_STRAPP_test_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_10My,
   display_plot = TRUE,
   # PDF_file_path = "./plot_STRAPP_histogram_10My.pdf",
   plot_posthoc_tests = FALSE)
# Adjust aesthetics a posteriori
histogram_ggplot_adj <- histogram_ggplot +
   ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
```

```
print(histogram_ggplot_adj)

# ----- Plot histogram of STRAPP overall test results from run_deepSTRAPP_over_time() ----- #

## Load directly outputs from run_deepSTRAPP_over_time()
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

# Select focal_time = 10My
focal_time <- 10

## Plot histogram for overall test
histogram_ggplot <- plot_histogram_STRAPP_test_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
   focal_time = focal_time,
   display_plot = TRUE,
   # PDF_file_path = "./plot_STRAPP_histogram_10My.pdf",
   plot_posthoc_tests = FALSE)
# Adjust aesthetics a posteriori
histogram_ggplot_adj <- histogram_ggplot +
  ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(histogram_ggplot_adj)

# ----- Example 2: Categorical trait ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
```

```
## Produce densityMaps using stochastic character mapping based on an ARD Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
   tip_data = Ponerinae_cat_3lvl_tip_data,
   phylo = Ponerinae_tree_old_calib,
   trait_data_type = "categorical",
   colors_per_states = colors_per_states,
   evolutionary_models = "ARD", # Use default ARD model
   nb_simulations = 100, # Reduce number of simulations to save time
   seed = 1234, # Seet seed for reproducibility
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   plot_map = FALSE)

# Load directly output
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")

## Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My <- run_deepSTRAPP_for_focal_time(
   densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
   nb_simulations = 100,
   ace = Ponerinae_cat_3lvl_data_old_calib$ace,
   tip_data = Ponerinae_cat_3lvl_tip_data,
   trait_data_type = "categorical",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   focal_time = focal_time,
   rate_type = "net_diversification",
   uncertainty_strategy = "paired",
   posthoc_pairwise_tests = TRUE,
   return_perm_data = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_object = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My, max.level = 1)

# ------ Plot histogram of STRAPP overall test results ------ #

histogram_ggplot <- plot_histogram_STRAPP_test_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My,
   display_plot = TRUE,
   # PDF_file_path = "./plot_STRAPP_histogram_overall_test.pdf",
   plot_posthoc_tests = FALSE)
# Adjust aesthetics a posteriori
histogram_ggplot_adj <- histogram_ggplot +
   ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(histogram_ggplot_adj)

# ------ Plot histograms of STRAPP post hoc test results ------ #
```

```
 histograms_ggplot_list <- plot_histogram_STRAPP_test_for_focal_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My,
    display_plot = TRUE,
    # PDF_file_path = "./plot_STRAPP_histograms_posthoc_tests.pdf",
    plot_posthoc_tests = TRUE)
 # Plot all histograms one by one
 print(histograms_ggplot_list)
 # Plot all histograms on one faceted plot
 cowplot::plot_grid(plotlist = histograms_ggplot_list)
}
```

`plot_rates_through_time` *Plot evolution of diversification rates in relation to trait values over time*

Description

Plot the evolution of diversification rates in relation to trait values extracted for multiple `time_steps` with `run_deepSTRAPP_over_time()`.

Rates are averaged across branches at each time step (i.e., `focal_time`).

- For continuous data, branches are grouped by ranges of trait values defined by `quantile_ranges`.
- For categorical data, branches are grouped by trait states.
- For biogeographic data, branches are grouped by ranges.

Usage

```
plot_rates_through_time(
  deepSTRAPP_outputs,
  rate_type = "net_diversification",
  quantile_ranges = c(0, 0.25, 0.5, 0.75, 1),
  select_trait_levels = "all",
  time_range = NULL,
  color_scale = NULL,
  colors_per_levels = NULL,
  plot_CI = FALSE,
  CI_type = "fuzzy",
  CI_quantiles = 0.95,
  display_plot = TRUE,
  PDF_file_path = NULL,
  return_mean_data_per_samples_df = FALSE,
  return_median_data_across_samples_df = FALSE,
  verbose = TRUE
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_over_time()`, that summarize the results of multiple STRAPP tests across `$time_steps`. The list needs to include two data.frame: `$trait_data_df_over_time` and `$diversification_data_df_over_time` by setting `extract_trait_data_melted_df = TRUE` and `extract_diversification_data_melted_df = TRUE`.

`rate_type`
: A character string specifying the type of diversification rates to use. Must be one of 'speciation', 'extinction' or 'net_diversification' (default). Even if the `deepSTRAPP_outputs` object was generated with `run_deepSTRAPP_over_time()` for testing another type of rates, the `$trait_data_df_over_time` and `$diversification_data_df_ove` data frames will contain data for all types of rates.

`quantile_ranges`
: Numeric vector. Only for continuous trait data. Quantiles used as thresholds to group branches by trait values. It must start with 0 and finish with 1. Default is `c(0, 0.25, 0.5, 0.75, 1.0)` which produces four balanced quantile groups.

`select_trait_levels`
: (Vector of) character string. Only for categorical and biogeographic trait data. To provide a list of a subset of states/ranges to plot. Names must match the ones found in `deepSTRAPP_outputs$trait_data_df_over_time$trait_value`. Default is `all` which means all states/ranges will be plotted.

`time_range`
: Numeric vector of length 2. Time boundaries used for the plot. If `NULL` (the default), the range of data provided in `deepSTRAPP_outputs` will be used.

`color_scale`
: Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` to display the quantile groups used to discretize the continuous trait data. From lowest values to highest values. Only for continuous data. Default = `NULL` will use the 'Spectral' color palette in `RColorBrewer::brewer.pal()`.

`colors_per_levels`
: Named character string. To set the colors to use to plot rates of each state/range. Names = states/ranges; values = colors. If `NULL` (default), the default ggplot2 color palette (`scales::hue_pal()`) will be used. Only for categorical and biogeographic data.

`plot_CI`
: Logical. Whether to plot a confidence interval (CI) based on the distribution of rates found in posterior samples. Default is `FALSE`.

`CI_type`
: Character string. To select the type of confidence interval (CI) to plot.
  - `fuzzy` (default): to overlay the evolution of rates found in all posterior samples with high transparency levels.
  - `quantiles_rect`: to add a polygon encompassing a proportion of the rate values found in posterior samples. This proportion is defined with `CI_quantiles`.

`CI_quantiles`
: Numeric. Proportion of rate values across posterior samples encompassed by the confidence interval. Only if `CI_type = "quantiles_rect"`. Default is `0.95`.

`display_plot`
: Logical. Whether to display the plot generated in the R console. Default is `TRUE`.

`PDF_file_path`
: Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

return_mean_data_per_samples_df
: Logical. Whether to include in the output the data.frame of mean rates per trait values computed for each stochastic map X BAMM posterior sample at each time-step (aggregated across groups of branches based on trait data). This is used to draw the confidence interval. Default is FALSE.

return_median_data_across_samples_df
: Logical. Whether to include in the output the data.frame of median rates per trait values across stochastic maps X BAMM posterior samples computed at each time-step (aggregated across groups of branches based on trait data AND stochastic maps X BAMM posterior samples). This is used to draw the median lines on the plot. Default is FALSE.

verbose
: Logical. Should progression be displayed? Default is TRUE.

Value

The function returns a list with at least one element.

- rates_TT_ggplot An object of classes gg and ggplot. This is a ggplot that can be displayed on the console with print(output$rates_TT_ggplot). It corresponds to the plot being displayed on the console when the function is run, if display_plot = TRUE, and can be further modified for aesthetics using the ggplot2 grammar.

Optional summary data frames:

- mean_data_per_samples_df A data.frame with five columns providing the $mean_rates observed along branches with a similar $trait_value (if categorical or biogeographic) or falling into the same $quantile_ranges. Data are extracted for each stochastic map ($Map_ID) X BAMM posterior sample ($BAMM_sample_ID) used for the STRAPP tests at each time-step (i.e., $focal_time). This is used to draw the confidence interval. Included if return_mean_data_per_samples_df = TRUE.
- $median_data_across_samples_df A data.frame with three columns providing the $median_rates observed across all stochastic maps X BAMM posterior samples used for the STRAPP tests in $mean_data_per_samples_df. This is used to draw the lines on the plot. Included if return_median_data_across_samples_df = TRUE.

If a PDF_file_path is provided, the function will also generate a PDF file of the plot.

Author(s)

Maël Doré

See Also

run_deepSTRAPP_over_time()

For a guided tutorial, see this vignette: vignette("plot_rates_through_time", package = "deepSTRAPP")

Examples

```
# ------ Example 1: Plot rates through time for continuous data ------ #

if (deepSTRAPP::is_dev_version())
{
  ## Load results of run_deepSTRAPP_over_time()
  Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
     "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
  ## This dataset is only available in development versions installed from GitHub.
  # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

  # Visualize trait data
  hist(Ponerinae_deepSTRAPP_cont_old_calib_0_40$trait_data_df_over_time$trait_value,
     xlab = "Trait values", main = NULL)

  # Generate plot
  plotTT_continuous <- plot_rates_through_time(
     deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
     quantile_ranges = c(0, 0.25, 0.5, 0.75, 1.0),
     time_range = c(0, 50), # Control range of the X-axis
     # color_scale = c("limegreen", "red"),
     plot_CI = TRUE,
     CI_type = "quantiles_rect",
     CI_quantiles = 0.9,
     display_plot = FALSE,
     # PDF_file_path = "./plotTT_continuous.pdf",
     return_mean_data_per_samples_df = TRUE,
     return_median_data_across_samples_df = TRUE)

  # Explore output
  # str(plotTT_continuous, max.level = 1)

  # Plot
  print(plotTT_continuous$rates_TT_ggplot)
  # Adjust aesthetics of plot a posteriori
  plotTT_continuous_adj <- plotTT_continuous$rates_TT_ggplot +
      ggplot2::theme(
         plot.title = ggplot2::element_text(color = "red", size = 15),
         axis.title = ggplot2::element_text(size = 14),
         axis.text = ggplot2::element_text(size = 12))
  # Plot again
  print(plotTT_continuous_adj)
}


# ------ Example 2: Plot rates through time for categorical data ------ #

if (deepSTRAPP::is_dev_version())
{
  ## Load results of run_deepSTRAPP_over_time()
  Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- readRDS(system.file("extdata",
```

```
    "Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40.rds", package = "deepSTRAPP"))
  ## This dataset is only available in development versions installed from GitHub.
  # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

  # Explore trait data
 table(Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40$trait_data_df_over_time$trait_value)

  # Set colors to use
  colors_per_states <- c("forestgreen", "sienna", "goldenrod")
  names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

  # Generate plot only for "arboreal" and "terricolous"
  plotTT_categorical <- plot_rates_through_time(
      deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
      select_trait_levels = c("arboreal", "terricolous"),
      time_range = c(0, 50),
      colors_per_levels = colors_per_states,
      plot_CI = TRUE,
      CI_type = "quantiles_rect",
      CI_quantiles = 0.9,
      display_plot = FALSE,
      # PDF_file_path = "./plotTT_categorical.pdf",
      return_mean_data_per_samples_df = TRUE,
      return_median_data_across_samples_df = TRUE)

  # Explore output
  # str(plotTT_categorical, max.level = 1)

  # Adjust aesthetics of plot a posteriori
  plotTT_categorical_adj <- plotTT_categorical$rates_TT_ggplot +
      ggplot2::theme(
         plot.title = ggplot2::element_text(size = 15),
         axis.title = ggplot2::element_text(size = 14),
         axis.text = ggplot2::element_text(size = 12))
  print(plotTT_categorical_adj)
}

# ------ Example 3: Plot rates through time for biogeographic data ------ #

if (deepSTRAPP::is_dev_version())
{
  ## Load results of run_deepSTRAPP_over_time()
  Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- readRDS(system.file("extdata",
     "Ponerinae_deepSTRAPP_biogeo_old_calib_0_40.rds", package = "deepSTRAPP"))
  ## This dataset is only available in development versions installed from GitHub.
  # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

  # Explore range data
  table(Ponerinae_deepSTRAPP_biogeo_old_calib_0_40$trait_data_df_over_time$trait_value)

  # Set colors to use
```

```
  colors_per_ranges <- c("mediumpurple2", "peachpuff2")
  names(colors_per_ranges) <- c("N", "O")

  plotTT_biogeographic <- plot_rates_through_time(
      deepSTRAPP_outputs = Ponerinae_deepSTRAPP_biogeo_old_calib_0_40,
      select_trait_levels = "all",
      time_range = c(0, 50),
      colors_per_levels = colors_per_ranges,
      plot_CI = TRUE,
      CI_type = "quantiles_rect",
      CI_quantiles = 0.9,
      display_plot = FALSE,
      # PDF_file_path = "./plotTT_biogeographic.pdf",
      return_mean_data_per_samples_df = TRUE,
      return_median_data_across_samples_df = TRUE)

  # Explore output
  # str(plotTT_biogeographic, max.level = 1)

  # Adjust aesthetics of plot a posteriori
  plotTT_biogeographic_adj <- plotTT_biogeographic$rates_TT_ggplot +
      ggplot2::theme(
         plot.title = ggplot2::element_text(size = 15),
         axis.title = ggplot2::element_text(size = 14),
         axis.text = ggplot2::element_text(size = 12))
  print(plotTT_biogeographic_adj)
}
```

`plot_rates_vs_trait_data_for_focal_time` *Plot rates vs. trait data for a given focal time*

Description

Plot rates vs. trait data of branches as extracted for a given focal time. Data are extracted from the output of a deepSTRAPP run carried out with `run_deepSTRAPP_for_focal_time()` or `run_deepSTRAPP_over_time()`).

Returns a single plot showing mean rates vs. trait data extracted across all branches for a given focal time. If the trait data are 'continuous', the plot is a scatter plot. If the trait data are 'categorical' or 'biogeographic', the plot is a boxplot.

If a PDF file path is provided in `PDF_file_path`, the plot will be saved directly in a PDF file.

Usage

```
plot_rates_vs_trait_data_for_focal_time(
  deepSTRAPP_outputs,
  focal_time = NULL,
  rate_type = "net_diversification",
  select_trait_levels = "all",
```

```
  color_scale = NULL,
  colors_per_levels = NULL,
  display_plot = TRUE,
  PDF_file_path = NULL,
  return_mean_data_per_branches_df = FALSE
)
```

Arguments

deepSTRAPP_outputs	List of elements generated with run_deepSTRAPP_for_focal_time(), that summarize the results of a STRAPP test for a specific time in the past (i.e. the focal_time). The list needs to include two data.frames: $trait_data_df_over_time and $diversification_data_df_over_time by setting extract_trait_data_melted_df = TRUE and extract_diversification_data_melted_df = TRUE. deepSTRAPP_outputs can also be extracted from the output of run_deepSTRAPP_over_time() that runs the whole deepSTRAPP workflow over multiple time-steps.
focal_time	Numeric. (Optional) If deepSTRAPP_outputs comprises results over multiple time-steps (i.e., output of run_deepSTRAPP_over_time(), this is the time of the STRAPP test targeted for plotting.
rate_type	A character string specifying the type of diversification rates to plot. Must be one of 'speciation', 'extinction' or 'net_diversification' (default). Even if the deepSTRAPP_outputs object was generated with run_deepSTRAPP_over_time() for testing another type of rates, the object will contain data for all types of rates.
select_trait_levels	(Vector of) character string. Only for categorical and biogeographic trait data. To provide a list of a subset of states/ranges to plot. Names must match the ones found in the deepSTRAPP_outputs. Default is all which means all states/ranges will be plotted.
color_scale	Character vector. List of colors to use to build the color scale with grDevices::colorRampPalette() to display the points. Color scale from lowest values to highest rate values. Only for continuous data. Default = NULL will use the 'Spectral' color palette in RColorBrewer::brewer.pal().
colors_per_levels	Named character string. To set the colors to use to plot data points and box for each state/range. Names = states/ranges; values = colors. If NULL (default), the default ggplot2 color palette (scales::hue_pal()) will be used. Only for categorical and biogeographic data.
display_plot	Logical. Whether to display the plot generated in the R console. Default is TRUE.
PDF_file_path	Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".
return_mean_data_per_branches_df	Logical. Whether to include in the output the data.frame of mean rates per trait values/states/ranges of all branches computed at the focal time and used for the plot. Default is FALSE.

Details

The main input deepSTRAPP_outputs is the typical output of run_deepSTRAPP_for_focal_time(). It provides information on results of a STRAPP test performed at a given focal_time.

Plots are built based on both trait data and diversification data as extracted for the given focal_time. Such data are recorded in the outputs of a deepSTRAPP run carried out with run_deepSTRAPP_for_focal_time() when extract_trait_data_melted_df = TRUE for trait data, and extract_diversification_data_melted_df = TRUE for diversification data. Please ensure to select those arguments when running deepSTRAPP.

Alternatively, the main input deepSTRAPP_outputs can be the output of run_deepSTRAPP_over_time(), providing results of STRAPP tests over multiple time-steps. In this case, you must provide a focal_time to select the unique time-step used for plotting.

- extract_trait_data_melted_df must be set to TRUE so that trait data are returned in a melted data.frame among the outputs under $trait_data_df_over_time.
- extract_diversification_data_melted_df must be set to TRUE so that the diversification rates are returned in a melted data.frame among the outputs under $diversification_data_df_over_time.
- return_STRAPP_results must be set to TRUE so that the STRAPP results objects recording the summary statistics for the STRAPP tests are returned among the outputs under $STRAPP_results_over_time.

For plotting all time-steps at once, see plot_rates_vs_trait_data_over_time().

Value

The function returns a list with at least one element.

- rates_vs_trait_ggplot An object of classes gg and ggplot. This is a ggplot that can be displayed on the console with print(output$rates_vs_trait_ggplot). It corresponds to the plot being displayed on the console when the function is run, if display_plot = TRUE, and can be further modified for aesthetics using the ggplot2 grammar.

If the trait data are 'continuous', the plot is a scatter plot showing how mean diversification rates vary with mean trait values across branches. If the trait data are 'categorical' or 'biogeographic', the plot is a boxplot showing mean diversification rates by states/ranges per branch.

Each plot also displays summary statistics for the STRAPP test associated with the data displayed:

- An observed statistic computed across the mean traits/ranges and rates values shown on the plot. This is not the statistic of the STRAPP test itself, which is conducted across stochastic maps X BAMM posterior samples (i.e., it is a distribution, not a unique value).
- The quantile of null statistic distribution at the significance threshold used to define test significance. The test will be considered significant (i.e., the null hypothesis is rejected) if this value is higher than zero.
- The p-value of the associated STRAPP test.

Optional summary data.frame:

- mean_data_per_branches_df A data.frame with four columns providing the $mean_trait_values (continuous) / $states (categorical) / $ranges (biogeographic) and $mean_rates computed along branches ($tip_ID) at focal_time. Rates/Traits are averaged for each stochastic maps X BAMM posterior samples used for testing, in accordance with the uncertainty_strategy selected when running deepSTRAPP. This is the raw data used to draw the plot. Included if return_mean_data_per_branches_df = TRUE.

If a PDF_file_path is provided, the function will also generate a PDF file of the plot.

Author(s)

Maël Doré

See Also

Associated functions in deepSTRAPP: run_deepSTRAPP_for_focal_time() plot_rates_vs_trait_data_over_time()

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #

 # Load fake trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare trait data

 # Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

 # Select a color scheme from lowest to highest values
 color_scale = c("darkgreen", "limegreen", "orange", "red")

 # Get Ancestral Character Estimates based on a Brownian Motion model
 # To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

  # (May take several minutes to run)
 # Run a Stochastic Mapping based on a Brownian Motion model
 # to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
 # Plot contMap = stochastic mapping of continuous trait
 plot_contMap(contMap = Ponerinae_contMap,
              color_scale = color_scale)

 ## Set focal time to 10 Mya
 focal_time <- 10
```

```
## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_cont_old_calib_10My <- run_deepSTRAPP_for_focal_time(
   contMap = Ponerinae_contMap,
   ace = Ponerinae_ACE,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   focal_time = focal_time,
   rate_type = "net_diversification",
   uncertainty_strategy = "rates_only",
   return_perm_data = TRUE,
   # Need to be set to TRUE to save diversification data
   extract_diversification_data_melted_df = TRUE,
   # Need to be set to TRUE to save trait data
   extract_trait_data_melted_df = TRUE,
   return_updated_BAMM_object = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_cont_old_calib_10My, max.level = 1)

# ----- Plot scatterplot of rates vs. trait values from run_deepSTRAPP_for_focal_time() ----- #

# Get plot
rates_vs_trait_output <- plot_rates_vs_trait_data_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_10My,
   color_scale = c("grey80", "orange"),
   display_plot = TRUE,
   # PDF_file_path = "./plot_rates_vs_trait_10My.pdf"
   return_mean_data_per_branches_df = TRUE)
# Adjust aesthetics a posteriori
rates_vs_trait_ggplot_adj <- rates_vs_trait_output$rates_vs_trait_ggplot +
   ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(rates_vs_trait_ggplot_adj)

# Explore melted data.frame of mean rates and trait values extracted for the given focal time.
head(rates_vs_trait_output$mean_data_per_branches_df)

# ----- Plot scatterplot of rates vs. trait values from run_deepSTRAPP_over_time() ----- #

## Load directly outputs from run_deepSTRAPP_over_time()
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

# Select focal_time = 10My
focal_time <- 10

# Get plot
rates_vs_trait_output <- plot_rates_vs_trait_data_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
```

```
   focal_time = focal_time,
   color_scale = c("grey80", "purple"),
   display_plot = TRUE)
   # PDF_file_path = "./plot_rates_vs_trait_10My.pdf"

# Adjust aesthetics a posteriori
rates_vs_trait_ggplot_adj <- rates_vs_trait_output$rates_vs_trait_ggplot +
    ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(rates_vs_trait_ggplot_adj)


# ----- Example 2: Categorical trait ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
## Produce densityMaps using stochastic character mapping based on an ARD Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
   tip_data = Ponerinae_cat_3lvl_tip_data,
   phylo = Ponerinae_tree_old_calib,
   trait_data_type = "categorical",
   colors_per_states = colors_per_states,
   evolutionary_models = "ARD", # Use default ARD model
   nb_simulations = 100, # Reduce number of simulations to save time
   seed = 1234, # Seet seed for reproducibility
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   plot_map = FALSE)

# Load directly output
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")
```

```
## Set focal time to 10 Mya
focal_time <- 10

## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My <- run_deepSTRAPP_for_focal_time(
   densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
   ace = Ponerinae_cat_3lvl_data_old_calib$ace,
   tip_data = Ponerinae_cat_3lvl_tip_data,
   nb_simulations = 100,
   trait_data_type = "categorical",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   focal_time = focal_time,
   rate_type = "net_diversification",
   uncertainty_strategy = "paired",
   posthoc_pairwise_tests = TRUE,
   return_perm_data = TRUE,
   # Need to be set to TRUE to save diversification data
   extract_diversification_data_melted_df = TRUE,
   # Need to be set to TRUE to save trait data
   extract_trait_data_melted_df = TRUE,
   return_updated_BAMM_object = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My, max.level = 1)

## Plot rates vs. states
rates_vs_trait_output <- plot_rates_vs_trait_data_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_10My,
   focal_time = 10,
   select_trait_levels = c("arboreal", "terricolous"), # Select only two levels
   colors_per_levels = colors_per_states[c("arboreal", "terricolous")], # Adjust colors
   display_plot = TRUE,
   # PDF_file_path = "./plot_rates_vs_trait_10My.pdf",
   return_mean_data_per_branches_df = TRUE)

# Adjust aesthetics a posteriori
rates_vs_trait_ggplot_adj <- rates_vs_trait_output$rates_vs_trait_ggplot +
   ggplot2::theme(plot.title = ggplot2::element_text(color = "red", size = 15))
print(rates_vs_trait_ggplot_adj)

# Explore melted data.frame of mean rates and states extracted for the given focal time.
head(rates_vs_trait_output$mean_data_per_branches_df)
}
```

---

plot_rates_vs_trait_data_over_time

Plot mean rates vs. trait data over time-steps

Description

Plot mean rates vs. trait data of branches as extracted for all time-steps. Data are extracted from the output of a deepSTRAPP run carried out with `run_deepSTRAPP_over_time()`) over multiple time-steps.

For each time-step, returns a plot showing rates vs. trait data. If the trait data are 'continuous', the plot is a scatter plot. If the trait data are 'categorical' or 'biogeographic', the plot is a boxplot.

If a PDF file path is provided in `PDF_file_path`, all plots will be saved directly in a unique PDF file, with one page per plot/time-step.

Usage

```
plot_rates_vs_trait_data_over_time(
  deepSTRAPP_outputs,
  rate_type = "net_diversification",
  select_trait_levels = "all",
  color_scale = NULL,
  colors_per_levels = NULL,
  display_plot = TRUE,
  PDF_file_path = NULL,
  return_mean_data_per_branches_df = FALSE
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_over_time()` that runs the whole deepSTRAPP workflow over multiple time-steps. The list needs to include two data.frames: `$trait_data_df_over_time` and `$diversification_data_df_over_time` by setting `extract_trait_data_melted_df = TRUE` and `extract_diversification_data_melted_df = TRUE`.

`rate_type`
: A character string specifying the type of diversification rates to plot. Must be one of 'speciation', 'extinction' or 'net_diversification' (default). Even if the `deepSTRAPP_outputs` object was generated with `run_deepSTRAPP_over_time()` for testing another type of rates, the object will contain data for all types of rates.

`select_trait_levels`
: (Vector of) character string. Only for categorical and biogeographic trait data. To provide a list of a subset of states/ranges to plot. Names must match the ones found in the `deepSTRAPP_outputs`. Default is `all` which means all states/ranges will be plotted.

`color_scale`
: Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` to display the points. Color scale from lowest values to highest rate values. Only for continuous data. Default = `NULL` will use the 'Spectral' color palette in `RColorBrewer::brewer.pal()`.

`colors_per_levels`
: Named character string. To set the colors to use to plot data points and box for each state/range. Names = states/ranges; values = colors. If `NULL` (default),

the default ggplot2 color palette (scales::hue_pal()) will be used. Only for categorical and biogeographic data.

display_plot Logical. Whether to display the plot generated in the R console. Default is TRUE.

PDF_file_path Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

return_mean_data_per_branches_df
Logical. Whether to include in the output the data.frame of mean rates per trait values/states/ranges of all branches computed over all time-steps and used for the plots. Default is FALSE.

Details

The main input deepSTRAPP_outputs is the typical output of run_deepSTRAPP_over_time(). It provides information on results of a STRAPP test performed over multiple time-steps.

Plots are built based on both trait data and diversification data as extracted for the time-steps. Such data are recorded in the outputs of a deepSTRAPP run carried out with run_deepSTRAPP_over_time().

- extract_trait_data_melted_df must be set to TRUE so that trait data are returned in a melted data.frame among the outputs under $trait_data_df_over_time.
- extract_diversification_data_melted_df must be set to TRUE so that the diversification rates are returned among the outputs under $diversification_data_df_over_time.
- return_STRAPP_results must be set to TRUE so that the STRAPP results objects recording the summary statistics for the STRAPP tests are returned among the outputs under $STRAPP_results_over_time.

For plotting a single focal_time, see plot_rates_vs_trait_data_for_focal_time().

Value

The function returns a list with at least one element.

- rates_vs_trait_ggplots A list of objects of classes gg and ggplot ordered as in $time_steps. Each element corresponds to a ggplot for a given focal_time. They can be displayed on the console with print(output$rates_vs_trait_ggplots[[i]]). They correspond to the plots being displayed on the console one by one when the function is run, if display_plot = TRUE, and can be further modified for aesthetics using the ggplot2 grammar.

If the trait data are 'continuous', the plots are scatter plots showing how mean diversification rates vary with mean trait values across branches. If the trait data are 'categorical' or 'biogeographic', the plots are boxplots showing mean diversification rates per state/ranges per branch.

Each plot also displays summary statistics for the STRAPP test associated with the data displayed:

- An observed statistic computed across the mean traits/ranges and rates values shown on the plot. This is not the statistic of the STRAPP test itself, which is conducted across stochastic maps X BAMM posterior samples (i.e., it is a distribution, not a unique value).
- The quantile of null statistic distribution at the significance threshold used to define test significance. The test will be considered significant (i.e., the null hypothesis is rejected) if this value is higher than zero.
- The p-value of the associated STRAPP test.

Optional summary data.frame:

- `mean_data_per_branches_df` A data.frame with four columns providing the `$mean_trait_value` (continuous) / `$states` (categorical) / `$ranges` (biogeographic) and `$mean_rates` computed along branches (`$tip_ID`) at the different `focal_time`. Rates/Traits are averaged for each stochastic maps X BAMM posterior samples used for testing, in accordance with the `uncertainty_strategy` selected when running deepSTRAPP. This is the raw data used to draw each plot for each `focal_time`. Included if `return_mean_data_per_branches_df = TRUE`.

If a `PDF_file_path` is provided, the function will also generate a unique PDF file with one plot/page per `$time_steps`.

Author(s)

Maël Doré

See Also

Associated functions in deepSTRAPP: `run_deepSTRAPP_over_time()` `plot_rates_vs_trait_data_for_focal_time()`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #

 # Load fake trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

  ## Prepare trait data

 # Extract continuous trait data as a named vector
 Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                     nm = Ponerinae_trait_tip_data$Taxa)

 # Select a color scheme from lowest to highest values
 color_scale = c("darkgreen", "limegreen", "orange", "red")

 # Get Ancestral Character Estimates based on a Brownian Motion model
 # To obtain values at internal nodes
 Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

  # (May take several minutes to run)
 # Run a Stochastic Mapping based on a Brownian Motion model
```

```
# to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
# Plot contMap = stochastic mapping of continuous trait
plot_contMap(contMap = Ponerinae_contMap,
             color_scale = color_scale)

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

## Run deepSTRAPP on net diversification rates
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- run_deepSTRAPP_over_time(
   contMap = Ponerinae_contMap,
   ace = Ponerinae_ACE,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "rates_only",
   return_perm_data = TRUE,
   # Need to be set to TRUE to save trait data
   extract_trait_data_melted_df = TRUE,
   # Need to be set to TRUE to save diversification data
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_BAMM_object = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly trait data output
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

# Explore output
str(Ponerinae_deepSTRAPP_cont_old_calib_0_40, max.level = 1)

## Plot for all time-steps
rates_vs_trait_outputs <- plot_rates_vs_trait_data_over_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
   color_scale = c("grey80", "purple"), # Adjust color scale
   display_plot = TRUE,
   # PDF_file_path = "./plot_rates_vs_trait_0_40My.pdf",
   return_mean_data_per_branches_df = TRUE)

## Print plot for time step 3 = 10 My
```

```
print(rates_vs_trait_outputs$rates_vs_trait_ggplots[[3]])

## Explore melted data.frame of rates and trait data
head(rates_vs_trait_outputs$mean_data_per_branches_df)

# ----- Example 2: Categorical data ----- #

## Load data

# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
## Produce densityMaps using stochastic character mapping based on an ARD Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
   tip_data = Ponerinae_cat_3lvl_tip_data,
   phylo = Ponerinae_tree_old_calib,
   trait_data_type = "categorical",
   colors_per_levels = colors_per_states,
   evolutionary_models = "ARD",
   nb_simulations = 100,
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   plot_map = FALSE)

# Load directly trait data output
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

 # (May take several minutes to run)
```

```
## Run deepSTRAPP on net diversification rates across time-steps.
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- run_deepSTRAPP_over_time(
   densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
   ace = Ponerinae_cat_3lvl_data_old_calib$ace,
   tip_data = Ponerinae_cat_3lvl_tip_data,
   nb_simulations = 100,
   trait_data_type = "categorical",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   rate_type = "net_diversification",
   uncertainty_strategy = "paired",
   seed = 1234, # Set for reproducibility
   alpha = 0.10, # Select a generous level of significance for the sake of the example
   posthoc_pairwise_tests = TRUE,
   return_perm_data = TRUE,
   # Need to be set to TRUE to save trait data
   extract_trait_data_melted_df = TRUE,
   # Need to be set to TRUE to save diversification data
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_BAMM_object = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly deepSTRAPP output
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

# Explore output
str(Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40, max.level = 1)

# Adjust color scheme
colors_per_states <- c("orange", "dodgerblue", "red")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

## Plot for all time-steps
rates_vs_trait_outputs <- plot_rates_vs_trait_data_over_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
   colors_per_levels = colors_per_states, # Adjust color scheme
   display_plot = TRUE,
   # PDF_file_path = "./plot_rates_vs_trait_0_40My.pdf",
   return_mean_data_per_branches_df = TRUE)

## Print plot for time step 3 = 10 My
print(rates_vs_trait_outputs$rates_vs_trait_ggplots[[3]])

## Explore melted data.frame of rates and states
head(rates_vs_trait_outputs$mean_data_per_branches_df)
```

```
}
```

`plot_STRAPP_pvalues_over_time` *Plot evolution of p-values of STRAPP tests over time*

Description

Plot the evolution of the p-values of STRAPP tests carried out across multiple `time_steps`, obtained from `run_deepSTRAPP_over_time()`.

By default, return a plot with a single line for p-values of overall tests. If `plot_posthoc_tests = TRUE`, it will return a plot with multiple lines, one per pair in post hoc tests (only for multinomial data, with more than two states).

If a PDF file path is provided in `PDF_file_path`, the plot will be saved directly in a PDF file.

Usage

```
plot_STRAPP_pvalues_over_time(
  deepSTRAPP_outputs,
  time_range = NULL,
  pvalues_max = NULL,
  alpha = 0.05,
  display_plot = TRUE,
  plot_significant_time_frame = TRUE,
  plot_posthoc_tests = FALSE,
  select_posthoc_pairs = "all",
  plot_adjusted_pvalues = FALSE,
  PDF_file_path = NULL
)
```

Arguments

`deepSTRAPP_outputs`	List of elements generated with `run_deepSTRAPP_over_time()`, that summarize the results of multiple deepSTRAPP runs across `$time_steps`.
`time_range`	Numeric vector of length 2. Time boundaries used for the X-axis of the plot. If `NULL` (the default), the range of data provided in `deepSTRAPP_outputs` will be used.
`pvalues_max`	Numeric. Set the max boundary used for the Y-axis of the plot. If `NULL` (the default), the maximum p-value provided in `deepSTRAPP_outputs` will be used.
`alpha`	Numeric. Significance level to display as a red dashed line on the plot. If set to `NULL`, no line will be added. Default is `0.05`.
`display_plot`	Logical. Whether to display the plot generated in the R console. Default is `TRUE`.

`plot_significant_time_frame`
: Logical. Whether to display a green band over the time frame that yields significant results according to the chosen alpha level. Default is `TRUE`.

`plot_posthoc_tests`
: Logical. For multinomial data only. Whether to plot the p-values for the overall Kruskal-Wallis test across all states (`plot_posthoc_tests = FALSE`), or plot the p-values for the pairwise post hoc Dunn's test across pairs of states (`plot_posthoc_tests = TRUE`). Default is `FALSE`. This is only possible if `deepSTRAPP_outputs` contains the `$pvalues_summary_df_for_posthoc_pairwise_tests` element returned by `run_deepSTRAPP_over_time()` when `posthoc_pairwise_tests = TRUE`.

`select_posthoc_pairs`
: Character vector used to specify the pairs to include in the plot. Names of pairs must match the pairs found in `deepSTRAPP_outputs$pvalues_summary_df_for_posthoc_pairwise_t` Default is "all" to include all pairs.

`plot_adjusted_pvalues`
: Logical. Whether to display the p-values adjusted for multiple testing rather than the raw p-values. See argument 'p.adjust_method' in `run_deepSTRAPP_for_focal_time()` or `run_deepSTRAPP_over_time()`. Default is `FALSE`.

`PDF_file_path`
: Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

Plots are built based on the p-values recorded in summary_df provided by `run_deepSTRAPP_over_time()`.

For overall tests, those p-values are found in `$pvalues_summary_df`.

For multinomial data (categorical or biogeographic data with more than 2 states), it is possible to plot p-values of post hoc pairwise tests. Set `plot_posthoc_tests = TRUE` to generate plots for the pairwise post hoc Dunn's test across pairs of states. To achieve this, the `deepSTRAPP_outputs` input object must contain a `$pvalues_summary_df_for_posthoc_pairwise_tests` element that summarizes p-values computed across pairs of states for all post hoc tests. This is obtained from `run_deepSTRAPP_over_time()` when setting `posthoc_pairwise_tests = TRUE` to carry out post hoc tests.

Value

The function returns a list of classes `gg` and `ggplot`. This object is a ggplot that can be displayed on the console with `print(output)`. It corresponds to the plot being displayed on the console when the function is run, if `display_plot = TRUE`, and can be further modified for aesthetics using the ggplot2 grammar.

If a `PDF_file_path` is provided, the function will also generate a PDF file of the plot.

Author(s)

Maël Doré

See Also

`run_deepSTRAPP_over_time()`

Examples

```
if (deepSTRAPP::is_dev_version())
{
  ## Load results of run_deepSTRAPP_over_time() for categorical data with 3-levels
  Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- readRDS(system.file("extdata",
     "Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40.rds", package = "deepSTRAPP"))
  ## This dataset is only available in development versions installed from GitHub.
  # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Plot results of overall Kruskal-Wallis / Mann-Whitney-Wilcoxon tests across all time-steps
  plot_overall <- plot_STRAPP_pvalues_over_time(
     deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
     alpha = 0.10,
     time_range = c(0, 30), # Adjust time range if needed
     display_plot = FALSE)

  # Adjust aesthetics a posteriori
  plot_overall <- plot_overall +
     ggplot2::theme(
        plot.title = ggplot2::element_text(size = 16))

  print(plot_overall)

  ## Plot results of post hoc pairwise Dunn's tests between selected pairs of states
  plot_posthoc <- plot_STRAPP_pvalues_over_time(
     deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40,
     alpha = 0.10,
     plot_posthoc_tests = TRUE,
     # PDF_file_path = "./pvalues_over_time.pdf",
     select_posthoc_pairs = c("arboreal != subterranean",
                              "arboreal != terricolous"),
     display_plot = FALSE)

  # Adjust aesthetics a posteriori
  plot_posthoc <- plot_posthoc +
     ggplot2::theme(
        plot.title = ggplot2::element_text(size = 16),
        legend.title = ggplot2::element_text(size  = 14),
        legend.position.inside = c(0.25, 0.25))

  print(plot_posthoc)
}
```

`plot_traits_vs_rates_on_phylogeny_for_focal_time` *Plot trait/range evolution vs. diversification rates and regime shifts on phylogeny*

Description

Plot two mapped phylogenies with evolutionary data with branches cut off at `focal_time`.

- Left facet: plot the evolution of trait data/geographic ranges on the left time-calibrated phylogeny.
  - For continuous data: Each branch is colored according to the estimates value of the traits.
  - For categorical and biogeographic data: Each branch is colored according to the posterior probability of being in a given state/range. Colors for each state/range are overlaid using transparency to produce a single plot for all states/ranges.
- Right facet: plot the evolution of diversification rates and location of regime shifts estimated from a BAMM (Bayesian Analysis of Macroevolutionary Mixtures). Each branch is colored according to the estimated rates of speciation, extinction, or net diversification stored in an object of class `"bammdata"`. Rates can vary along time, thus colors evolve along individual branches.

This function is a wrapper of multiple plotting functions:

- For continuous traits: `plot_contMap()`
- For categorical and biogeographic data: `plot_densityMaps_overlay()`
- For BAMM rates and regime shifts: `plot_BAMM_rates()`

Usage

```
plot_traits_vs_rates_on_phylogeny_for_focal_time(
  deepSTRAPP_outputs,
  focal_time = NULL,
  color_scale = NULL,
  colors_per_levels = NULL,
  add_ACE_pies = TRUE,
  rate_type = "net_diversification",
  keep_initial_colorbreaks = FALSE,
  add_regime_shifts = TRUE,
  configuration_type = "MAP",
  sample_index = 1,
  regimes_fill = "grey",
  regimes_size = 1,
  regimes_pch = 21,
  regimes_border_col = "black",
  regimes_border_width = 1,
  ...,
  cex_pies = 0.5,
  adjust_size_to_prob = TRUE,
  display_plot = TRUE,
  PDF_file_path = NULL
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_for_focal_time()`, that summarize the results of a STRAPP test for a specific time in the past (i.e. the `focal_time`). `deepSTRAPP_outputs` can also be extracted from the output of `run_deepSTRAPP_over_time()` that run the whole deepSTRAPP workflow over multiple time-steps. This must contain an updated `contMap`/`densityMaps` and updated `BAMM_object` (see Details section below).

`focal_time`
: Numeric. (Optional) If `deepSTRAPP_outputs` comprises results over multiple time-steps (i.e., output of `run_deepSTRAPP_over_time()`, this is the time of the STRAPP test targeted for plotting.

`color_scale`
: Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` showing the evolution of a continuous trait. From lowest values to highest values. (For continuous trait data only)

`colors_per_levels`
: Named character string. To set the colors to use to map each state/range posterior probabilities. Names = states/ranges; values = colors. If `NULL` (default), the color scale provided in `densityMaps` will be used. (For categorical and biogeographic data only)

`add_ACE_pies`
: Logical. Whether to add pies of posterior probabilities of states/ranges at internal nodes on the mapped phylogeny. Default = `TRUE`.

`rate_type`
: A character string specifying the type of diversification rates to plot. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).

`keep_initial_colorbreaks`
: Logical. Whether to keep the same color breaks as used for the most recent focal time. Typically, the current time (t = 0). This will only work if you provide the output of `run_deepSTRAPP_over_time()` as `deepSTRAPP_outputs`. Default = `FALSE`.

`add_regime_shifts`
: Logical. Whether to add the location of regime shifts on the phylogeny (Step 2). Default is `TRUE`.

`configuration_type`
: A character string specifying how to select the location of regime shifts across posterior samples.

  - `configuration_type = "MAP"`: Use the average locations recorded in posterior samples with the Maximum A Posteriori probability (MAP) configuration. This regime shift configuration is the most frequent configuration among the posterior samples (See `BAMMtools::getBestShiftConfiguration()`). This is the default option.
  - `configuration_type = "MSC"`: Use the average locations recorded in posterior samples with the Maximum Shift Credibility (MSC) configuration. This regime shift configuration has the highest product of marginal probabilities across branches (See `BAMMtools::maximumShiftCredibility()`).
  - `configuration_type = "index"`: Use the configuration of a unique posterior sample whose index is provided in `sample_index`.

`sample_index` Integer. Index of the posterior samples to use to plot the location of regime shifts. Used only if `configuration_type = index`. Default = 1.

`regimes_fill` Character string. Set the color of the background of the symbols showing the location of regime shifts. Equivalent to the bg argument in `BAMMtools::addBAMMshifts()`. Default is `"grey"`.

`regimes_size` Numeric. Set the size of the symbols showing the location of regime shifts. Equivalent to the `cex` argument in `BAMMtools::addBAMMshifts()`. Default is 1.

`regimes_pch` Integer. Set the shape of the symbols showing the location of regime shifts. Equivalent to the `pch` argument in `BAMMtools::addBAMMshifts()`. Default is 21.

`regimes_border_col` Character string. Set the color of the border of the symbols showing the location of regime shifts. Equivalent to the `col` argument in `BAMMtools::addBAMMshifts()`. Default is `"black"`.

`regimes_border_width` Numeric. Set the width of the border of the symbols showing the location of regime shifts. Equivalent to the `lwd` argument in `BAMMtools::addBAMMshifts()`. Default is 1.

`...` Additional graphical arguments to pass down to `phytools::plot.contMap()`, `phytools::plotSimmap()`, `plot_densityMaps_overlay()`, `BAMMtools::plot.bammdata()`, `BAMMtools::addBAMMshifts()`, and `par()`. If provided, `par.reset` is ignored with a message. See `plot_BAMM_rates()` for details.

`cex_pies` Numeric. To adjust the size of the ACE pies. Default = 0.5. Provide the full argument name to avoid ambiguity with the `cex` graphical argument.

`adjust_size_to_prob` Logical. Whether to scale the size of the symbols showing the location of regime shifts according to the marginal shift probability of the shift happening on each location/branch. This will only work if there is an `$MSP_tree` element summarizing the marginal shift probabilities across branches in the `BAMM_object`. Default is `TRUE`. Provide the full argument name to avoid ambiguity with the `adj` graphical argument.

`display_plot` Logical. Whether to display the plot generated in the R console. Default is `TRUE`.

`PDF_file_path` Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

The main input `deepSTRAPP_outputs` is the typical output of `run_deepSTRAPP_for_focal_time()`. It provides information on results of a STRAPP test performed at a given `focal_time`, and can also encompass updated phylogenies with mapped trait evolution and diversification rates and regime shifts if appropriate arguments are set.

- `return_updated_Maps` must be set to `TRUE` so that the updated version of mapped phylogeny (`contMap`/`densityMaps`) for the given `focal_time` is returned among the outputs under `$updated_Maps`. The update of the `contMap`/`densityMaps` consists of cutting off branches and mappings that are younger than the `focal_time`. `contMap`/`densityMaps` must have been provided among the inputs to `run_deepSTRAPP_for_focal_time()` in the first place.

- `return_updated_BAMM_object` must be set to `TRUE` so that the `updated_BAMM_object` with phylogeny and mapped diversification rates cut-off at the `focal_time` are returned among the outputs under `$updated_BAMM_object`.

`$MAP_BAMM_object` and `$MSC_BAMM_object` elements are required in `$updated_BAMM_object` to plot regime shift locations following the "MAP" or "MSC" `configuration_type` respectively. A `$MSP_tree` element is required to scale the size of the symbols showing the location of regime shifts according to marginal shift probabilities. (If `adjust_size_to_prob = TRUE`).

Alternatively, the main input `deepSTRAPP_outputs` can be the output of `run_deepSTRAPP_over_time()`, providing results of STRAPP tests over multiple time-steps. In this case, you must provide a `focal_time` to select the unique time-step used for plotting.

- `return_updated_Maps` must be set to `TRUE` so that the updated version of mapped phylogenies (`contMap`/`densityMaps`) are returned among the outputs under `$updated_Maps_over_time`.
- `return_updated_BAMM_object` must be set to `TRUE` so that the `BAMM_objects` with phylogeny and mapped diversification rates cut-off at the specified time-steps are returned among the outputs under `$updated_BAMM_objects_over_time`.

For plotting all time-steps at once, see `plot_traits_vs_rates_on_phylogeny_over_time()`.

Value

If `display_plot = TRUE`, the function displays a plot with two facets in the R console:

- (Left) A time-calibrated phylogeny displaying the evolution of trait/biogeographic data.
- (Right) A time-calibrated phylogeny displaying diversification rates and regime shifts.

If `PDF_file_path` is provided, the plot will be exported in a PDF file.

Author(s)

Maël Doré

See Also

`phytools::plot.densityMap()` `plot_densityMaps_overlay()` `plot_BAMM_rates()`

Functions in deepSTRAPP needed to produce the `deepSTRAPP_outputs` as input: `run_deepSTRAPP_for_focal_time()` `run_deepSTRAPP_over_time()` Function in deepSTRAPP to plot all time-steps at once: `plot_traits_vs_rates_on_phylo`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #
 ## Load data

 # Load trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
```

```
# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

# Select a color scheme from lowest to highest values
color_scale = c("darkgreen", "limegreen", "orange", "red")

# Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

 # (May take several minutes to run)
# Run a Stochastic Mapping based on a Brownian Motion model
# to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
# Plot contMap = stochastic mapping of continuous trait
plot_contMap(contMap = Ponerinae_contMap,
             color_scale = color_scale)

## Set focal time to 10 Mya
focal_time <- 10

## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_cont_old_calib_10My <- run_deepSTRAPP_for_focal_time(
  contMap = Ponerinae_contMap,
  ace = Ponerinae_ACE,
  tip_data = Ponerinae_cont_tip_data,
  trait_data_type = "continuous",
  BAMM_object = Ponerinae_BAMM_object_old_calib,
  focal_time = focal_time,
  uncertainty_strategy = "rates_only",
  return_updated_Maps = TRUE,
  return_updated_BAMM_object = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_cont_old_calib_10My, max.level = 1)

## Plot updated contMap vs. updated diversification rates
plot_traits_vs_rates_on_phylogeny_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_10My,
   keep_initial_colorbreaks = TRUE, # To use the same color breaks as for t = 0 My
   color_scale = c("limegreen", "orange", "red"), # Adjust color scale on contMap
```

```
   legend = TRUE, labels = TRUE, # Show legend and label on BAMM plot
   cex = 0.7) # Adjust label size on contMap
   # PDF_file_path = "Updated_maps_cont_10My.pdf")

# ----- Example 2: Biogeographic data ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_binary_range_table, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare range data for Old World vs. New World

# No overlap in ranges
table(Ponerinae_binary_range_table$Old_World, Ponerinae_binary_range_table$New_World)

Ponerinae_NO_tip_data <- stats::setNames(object = Ponerinae_binary_range_table$Old_World,
                                         nm = Ponerinae_binary_range_table$Taxa)
Ponerinae_NO_tip_data <- as.character(Ponerinae_NO_tip_data)
Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "TRUE"] <- "O" # O = Old World
Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "FALSE"] <- "N" # N = New World
names(Ponerinae_NO_tip_data) <- Ponerinae_binary_range_table$Taxa
table(Ponerinae_NO_tip_data)

colors_per_levels <- c("mediumpurple2", "peachpuff2")
names(colors_per_levels) <- c("N", "O")

 # (May take several minutes to run)

# Load ape and BioGeoBEARS to use models internally
library(ape)
library(BioGeoBEARS)

## Run evolutionary models
Ponerinae_biogeo_data <- prepare_trait_data(
    tip_data = Ponerinae_NO_tip_data,
    trait_data_type = "biogeographic",
    phylo = Ponerinae_tree_old_calib,
    evolutionary_models = "DEC+J", # Default = "DEC" for biogeographic
    BioGeoBEARS_directory_path = tempdir(), # Ex: "./BioGeoBEARS_directory/"
    keep_BioGeoBEARS_files = FALSE,
    prefix_for_files = "Ponerinae_old_calib",
    max_range_size = 2,
    split_multi_area_ranges = TRUE, # Set to TRUE to display the two outputs
    nb_simulations = 100, # Reduce to save time (Default = '1000')
```

```
    colors_per_levels = colors_per_levels,
    return_model_selection_df = TRUE,
    verbose = TRUE)

# Load directly output
data(Ponerinae_biogeo_data_old_calib, package = "deepSTRAPP")

## Explore output
str(Ponerinae_biogeo_data_old_calib, 1)

## Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

Ponerinae_deepSTRAPP_biogeo_old_calib_10My <- run_deepSTRAPP_for_focal_time(
   densityMaps = Ponerinae_biogeo_data_old_calib$densityMaps,
   ace = Ponerinae_biogeo_data_old_calib$ace,
   tip_data = Ponerinae_NO_tip_data,
   nb_simulations = 100,
   trait_data_type = "biogeographic",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   focal_time = focal_time,
   uncertainty_strategy = "paired",
   return_updated_Maps = TRUE,
   return_updated_BAMM_object = TRUE)

## Explore output
str(Ponerinae_deepSTRAPP_biogeo_old_calib_10My, max.level = 1)

## Plot updated contMap vs. updated diversification rates
plot_traits_vs_rates_on_phylogeny_for_focal_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_biogeo_old_calib_10My,
   # Adjust colors on densityMaps
   colors_per_levels = c("N" = "dodgerblue2", "O" = "orange"),
   # Show legend and label on BAMM plot
   legend = TRUE, labels = TRUE,
   cex_pies = 0.2, # Adjust size of ACE pies on densityMaps
   cex = 0.7) # Adjust size of tip labels on BAMM plot
   # PDF_file_path = "Updated_maps_biogeo_old_calib_10My.pdf")

# ----- Example 3: With outputs over_time ----- #

## Load directly outputs from run_deepSTRAPP_over_time()
Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_biogeo_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Explore output
str(Ponerinae_deepSTRAPP_biogeo_old_calib_0_40, max.level =  1)
```

```
 ## Plot updated contMap vs. updated diversification rates for focal_time = 40My
 plot_traits_vs_rates_on_phylogeny_for_focal_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_biogeo_old_calib_0_40,
    focal_time = 40, # Select focal_time = 40My
    # Adjust colors on densityMaps
    colors_per_levels = c("N" = "dodgerblue2", "O" = "orange"),
    # Show legend and label on BAMM plot
    legend = TRUE, labels = TRUE,
    cex_pies = 0.2, # Adjust size of ACE pies on densityMaps
    cex = 0.7) # Adjust size of tip labels on BAMM plot
    # PDF_file_path = "Updated_maps_biogeo_old_calib_40My.pdf")
}
```

`plot_traits_vs_rates_on_phylogeny_over_time`
Plot multiple mapped phylogenies of trait/range evolution vs. diversification rates and regime shifts over time-steps

---

Description

Plot two mapped phylogenies with evolutionary data with branches cut off for each time-step. Main input = output of a deepSTRAPP run over time using `run_deepSTRAPP_over_time()`).

- Left facet: plot the evolution of trait data/geographic ranges on the left time-calibrated phylogeny.
  - For continuous data: Each branch is colored according to the estimated value of the traits.
  - For categorical and biogeographic data: Each branch is colored according to the posterior probability of being in a given state/range. Colors for each state/range are overlaid using transparency to produce a single plot for all states/ranges.
- Right facet: plot the evolution of diversification rates and location of regime shifts estimated from a BAMM (Bayesian Analysis of Macroevolutionary Mixtures). Each branch is colored according to the estimated rates of speciation, extinction, or net diversification stored in an object of class `"bammdata"`. Rates can vary along time, thus colors evolve along individual branches.

This function is a wrapper of `run_deepSTRAPP_for_focal_time()` used to plot mapped phylogenies for a unique given `focal_time`.

Returns one plot per focal time in `$time_steps`. If a PDF file path is provided in `PDF_file_path`, the plots will be saved directly in a PDF file, with one page per focal time in `$time_steps`.

Usage

```
plot_traits_vs_rates_on_phylogeny_over_time(
  deepSTRAPP_outputs,
  color_scale = NULL,
  colors_per_levels = NULL,
```

```
  add_ACE_pies = TRUE,
  rate_type = "net_diversification",
  keep_initial_colorbreaks = FALSE,
  add_regime_shifts = TRUE,
  configuration_type = "MAP",
  sample_index = 1,
  regimes_fill = "grey",
  regimes_size = 1,
  regimes_pch = 21,
  regimes_border_col = "black",
  regimes_border_width = 1,
  ...,
  cex_pies = 0.5,
  adjust_size_to_prob = TRUE,
  display_plot = TRUE,
  PDF_file_path = NULL
)
```

Arguments

`deepSTRAPP_outputs`
: List of elements generated with `run_deepSTRAPP_over_time()`, that summarize the results of a STRAPP run over multiple time-steps in the past.

`color_scale`
: Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` showing the evolution of a continuous trait. From lowest values to highest values. (For continuous trait data only)

`colors_per_levels`
: Named character string. To set the colors to use to map each state/range posterior probabilities. Names = states/ranges; values = colors. If `NULL` (default), the color scale provided in `densityMaps` will be used. (For categorical and biogeographic data only)

`add_ACE_pies`
: Logical. Whether to add pies of posterior probabilities of states/ranges at internal nodes on the mapped phylogeny. Default = `TRUE`.

`rate_type`
: A character string specifying the type of diversification rates to plot. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).

`keep_initial_colorbreaks`
: Logical. Whether to keep the same color breaks as used for the most recent focal time. Typically, the current time (t = 0). Default = `FALSE`.

`add_regime_shifts`
: Logical. Whether to add the location of regime shifts on the phylogeny (Step 2). Default is `TRUE`.

`configuration_type`
: A character string specifying how to select the location of regime shifts across posterior samples.

  - `configuration_type = "MAP"`: Use the average locations recorded in posterior samples with the Maximum A Posteriori probability (MAP) configuration. This regime shift configuration is the most frequent configuration

among the posterior samples (See BAMMtools::getBestShiftConfiguration()). This is the default option.

- configuration_type = "MSC": Use the average locations recorded in posterior samples with the Maximum Shift Credibility (MSC) configuration. This regime shift configuration has the highest product of marginal probabilities across branches (See BAMMtools::maximumShiftCredibility()).
- configuration_type = "index": Use the configuration of a unique posterior sample whose index is provided in sample_index.

sample_index
: Integer. Index of the posterior samples to use to plot the location of regime shifts. Used only if configuration_type = index. Default = 1.

regimes_fill
: Character string. Set the color of the background of the symbols showing the location of regime shifts. Equivalent to the bg argument in BAMMtools::addBAMMshifts(). Default is "grey".

regimes_size
: Numeric. Set the size of the symbols showing the location of regime shifts. Equivalent to the cex argument in BAMMtools::addBAMMshifts(). Default is 1.

regimes_pch
: Integer. Set the shape of the symbols showing the location of regime shifts. Equivalent to the pch argument in BAMMtools::addBAMMshifts(). Default is 21.

regimes_border_col
: Character string. Set the color of the border of the symbols showing the location of regime shifts. Equivalent to the col argument in BAMMtools::addBAMMshifts(). Default is "black".

regimes_border_width
: Numeric. Set the width of the border of the symbols showing the location of regime shifts. Equivalent to the lwd argument in BAMMtools::addBAMMshifts(). Default is 1.

...
: Additional graphical arguments to pass down to phytools::plot.contMap(), phytools::plot.simmap(), plot_densityMaps_overlay(), BAMMtools::plot.bammdata(), BAMMtools::addBAMMshifts(), and par(). If provided, par.reset is ignored with a message. See plot_BAMM_rates() for details.

cex_pies
: Numeric. To adjust the size of the ACE pies. Default = 0.5. Provide the full argument name to avoid ambiguity with the cex graphical argument.

adjust_size_to_prob
: Logical. Whether to scale the size of the symbols showing the location of regime shifts according to the marginal shift probability of the shift happening on each location/branch. This will only work if there is an $MSP_tree element summarizing the marginal shift probabilities across branches in the BAMM_object. Default is TRUE. Provide the full argument name to avoid ambiguity with the adj graphical argument.

display_plot
: Logical. Whether to display the plot generated in the R console. Default is TRUE.

PDF_file_path
: Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

Details

The main input deepSTRAPP_outputs is the typical output of run_deepSTRAPP_over_time(). It provides information on results of a STRAPP run performed over multiple time-steps, and can also encompass updated phylogenies with mapped trait evolution and diversification rates and regime shifts if appropriate arguments are set.

- return_updated_Maps must be set to TRUE so that the updated version of mapped phylogeny (contMap/densityMaps) for the given $time_steps are returned among the outputs under $updated_Maps_over_time. The update of the contMap/densityMaps consists of cutting off branches and mappings that are younger than the successive $time_steps. contMap/densityMaps must have been provided among the inputs to run_deepSTRAPP_over_time() in the first place.
- return_updated_BAMM_objects must be set to TRUE so that the list of updated_BAMM_object with phylogenies and mapped diversification rates cut-off at the successive $time_steps are returned among the outputs under $updated_BAMM_objects_over_time.

$MAP_BAMM_object and $MSC_BAMM_object elements are required in the elements of $updated_BAMM_objects_over_time to plot regime shift locations following the "MAP" or "MSC" configuration_type respectively. A $MSP_tree element is required to scale the size of the symbols showing the location of regime shifts according to marginal shift probabilities. (If adjust_size_to_prob = TRUE).

For plotting a single time-step (i.e., $focal_time) at once, see plot_traits_vs_rates_on_phylogeny_for_focal_time()

Value

If display_plot = TRUE, the function displays the successive plots with two facets in the R console:

- (Left) A time-calibrated phylogeny displaying the evolution of trait/biogeographic data.
- (Right) A time-calibrated phylogeny displaying diversification rates and regime shifts. A plot per time-steps is generated and displayed successively in the console.

If PDF_file_path is provided, the plots will be exported in a unique PDF file with one page per time-step.

Author(s)

Maël Doré

See Also

plot_contMap() phytools::plot.densityMap() plot_densityMaps_overlay() plot_BAMM_rates()

Function in deepSTRAPP needed to produce the deepSTRAPP_outputs as input: run_deepSTRAPP_over_time()
Function in deepSTRAPP to plot a single time-step at once: plot_traits_vs_rates_on_phylogeny_for_focal_time()

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #

 ## Load data
```

```
# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
# Load phylogeny with old calibration
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

# Select a color scheme from lowest to highest values
color_scale = c("darkgreen", "limegreen", "orange", "red")

 # Get Ancestral Character Estimates based on a Brownian Motion model
# To obtain values at internal nodes
Ponerinae_ACE <- phytools::fastAnc(tree = Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data)

 # (May take several minutes to run)
# Run a Stochastic Mapping based on a Brownian Motion model
# to interpolate values along branches and obtain a "contMap" object
Ponerinae_contMap <- phytools::contMap(Ponerinae_tree_old_calib, x = Ponerinae_cont_tip_data,
                                       res = 100, # Number of time steps
                                       plot = FALSE)
# Plot contMap = stochastic mapping of continuous trait
plot_contMap(contMap = Ponerinae_contMap,
             color_scale = color_scale)

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

## Run deepSTRAPP on net diversification rates
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- run_deepSTRAPP_over_time(
   contMap = Ponerinae_contMap,
   ace = Ponerinae_ACE,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "rates_only",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
```

```
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly trait data output
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
    "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Plot updated contMap vs. updated diversification rates
plot_traits_vs_rates_on_phylogeny_over_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
   # To use the same color breaks as for t = 0 My in all BAMM plots
   keep_initial_colorbreaks = TRUE,
   color_scale = c("limegreen", "orange", "red"), # Adjust color scale on contMap
   legend = TRUE, labels = TRUE, # Show legend and label on BAMM plot
   cex = 0.7) # Adjust label size on contMap
   # PDF_file_path = "Updated_maps_cont_old_calib_0_40My.pdf")

# ----- Example 2: Biogeographic data ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_binary_range_table, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare range data for Old World vs. New World

# No overlap in ranges
table(Ponerinae_binary_range_table$Old_World, Ponerinae_binary_range_table$New_World)

Ponerinae_NO_tip_data <- stats::setNames(object = Ponerinae_binary_range_table$Old_World,
                                         nm = Ponerinae_binary_range_table$Taxa)
Ponerinae_NO_tip_data <- as.character(Ponerinae_NO_tip_data)
Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "TRUE"] <- "O" # O = Old World
Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "FALSE"] <- "N" # N = New World
names(Ponerinae_NO_tip_data) <- Ponerinae_binary_range_table$Taxa
table(Ponerinae_NO_tip_data)

colors_per_levels <- c("mediumpurple2", "peachpuff2")
```

```
names(colors_per_levels) <- c("N", "O")

 # (May take several minutes to run)

# Load ape and BioGeoBEARS to use models internally
library(ape)
library(BioGeoBEARS)

## Run evolutionary models
Ponerinae_biogeo_data <- prepare_trait_data(
   tip_data = Ponerinae_NO_tip_data,
   trait_data_type = "biogeographic",
   phylo = Ponerinae_tree_old_calib,
   evolutionary_models = "DEC+J", # Default = "DEC" for biogeographic
   BioGeoBEARS_directory_path = tempdir(), # Ex: "./BioGeoBEARS_directory/"
   keep_BioGeoBEARS_files = FALSE,
   prefix_for_files = "Ponerinae_old_calib",
   max_range_size = 2,
   split_multi_area_ranges = TRUE, # Set to TRUE to use only unique areas "A" and "B"
   nb_simulations = 100, # Reduce to save time (Default = '1000')
   colors_per_levels = colors_per_levels,
   return_model_selection_df = TRUE,
   verbose = TRUE)

# Load directly output
data(Ponerinae_biogeo_data_old_calib, package = "deepSTRAPP")

## Set for time steps of 5 My. Will generate deepSTRAPP workflows from 0 to 40 Mya.
time_range <- c(0, 40)
time_step_duration <- 5

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for time-steps = 0 to 40 Mya.
Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- run_deepSTRAPP_over_time(
   densityMaps = Ponerinae_biogeo_data_old_calib$densityMaps,
   ace = Ponerinae_biogeo_data_old_calib$ace,
   tip_data = Ponerinae_NO_tip_data,
   nb_simulations = 100,
   trait_data_type = "biogeographic",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "paired",
   seed = 1234, # Set seed for reproducibility
   alpha = 0.10, # Select a generous level of significance for the sake of the example
   rate_type = "net_diversification",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
```

```
   verbose_extended = TRUE)

## Load directly output
Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- readRDS(system.file("extdata",
    "Ponerinae_deepSTRAPP_biogeo_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Explore output
str(Ponerinae_deepSTRAPP_biogeo_old_calib_0_40, max.level =  1)

## Plot updated contMap vs. updated diversification rates
plot_traits_vs_rates_on_phylogeny_over_time(
   deepSTRAPP_outputs = Ponerinae_deepSTRAPP_biogeo_old_calib_0_40,
   # Adjust colors on densityMaps
   colors_per_levels = c("N" = "dodgerblue2", "O" = "orange"),
   legend = TRUE, labels = TRUE, # Show legend and label on BAMM plot
   cex_pies = 0.2, # Adjust size of ACE pies on densityMaps
   cex = 0.7) # Adjust label size on contMap
   # PDF_file_path = "Updated_maps_biogeo_old_calib_0_40My.pdf")
}
```

`Ponerinae_binary_range_table` *Dataset providing biogeographic range data for extant ponerine ants*

Description

A data.frame of range location for the 1534 extant ponerine ant taxa (Ponerinae subfamily).

Source: Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

Usage

```
data(Ponerinae_binary_range_table)
```

Format

A data.frame with 1534 rows and 10 columns.

Details

A data.frame of range locations covering the 1534 extant ponerine ant taxa (Ponerinae subfamily).

- `$Taxa` Character string. Names of Ponerinae ant taxa.

- `$Afrotropics` Logical. Whether the range of the taxa extends to Afrotropics.
- `$Australasia` Logical. Whether the range of the taxa extends to Australasia.
- `$Indomalaya` Logical. Whether the range of the taxa extends to Indomalaya.
- `$Nearctic` Logical. Whether the range of the taxa extends to Nearctic.
- `$Neotropics` Logical. Whether the range of the taxa extends to Neotropics.
- `$Eastern_Palearctic` Logical. Whether the range of the taxa extends to Eastern Palearctic.
- `$Western_Palearctic` Logical. Whether the range of the taxa extends to Western Palearctic.
- `$Old_World` Logical. Whether the range of the taxa extends to the Old World: encompassing any bioregion among Afrotropics, Australasia, Indomalaya, Eastern Palearctic, or Western Palearctic.
- `$New_World` Logical. Whether the range of the taxa extends to the New World: encompassing any bioregion among Nearctic, or Neotropics.

References

Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

`Ponerinae_biogeo_data_old_calib`

Data summarizing the evolution of geographic ranges in Ponerinae ants using an old ill-calibrated phylogeny for illustrative purposes

---

Description

A list containing geographic ranges data of Ponerinae mapped on the old ill-calibrated phylogeny, modeled with R package `BioGeoBEARS`. Ranges are labeled between "Old World" (O) and New World (N). This object was obtained with `prepare_trait_data()`. The phylogeny used is also NOT a properly time-calibrated phylogeny. It uses an ill-designed old calibration for illustrative purposes.

Source: Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

Usage

```
data(Ponerinae_biogeo_data_old_calib)
```

Format

A list with 6 elements.

Details

A list of six elements containing information on the biogeographic history of Ponerinae ants. This object was obtained with `prepare_trait_data()`.

- `$densityMaps` List of objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given range along branches. The list contains only a `"densityMap"` per unique area because `split_multi_area_ranges` was set to TRUE. In this case, unique areas are "N" (= "New World") and "O" (= "Old World")
- `$densityMaps_all_ranges` List of objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given range along branches. The list contains one `"densityMap"` per range found along branches during the simulated biogeographic histories. Here those ranges are "N" (= "New World"), "O" (= "Old World"), and "NO" for multi-area ranges encompassing both regions.
- `$trait_data_type` Character string. Record the type of trait data. Here: "biogeographic".
- `$ace` Numeric matrix. Record the posterior probabilities of ancestral ranges estimated at internal nodes. Only unique areas (i.e., "N" and "O") are considered among the ranges. Multi-area ranges (i.e., "NO") have been split among unique ranges. Rows are internal nodes. Columns are ranges. Values are posterior probabilities of each range per node.
- `$ace_all_ranges` Numeric matrix. Record the posterior probabilities of ancestral ranges estimated at internal nodes. All ranges observed along branches during the simulated biogeographic histories are present (i.e., "N", "O", and "NO"). Rows are internal nodes. Columns are ranges. Values are posterior probabilities of each range per node.
- `$model_selection_df` Data.frame that summarizes model comparisons used to select the best fitting model.

References

Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

See Also

`prepare_trait_data()`

Ponerinae_cat_2lvl_data_old_calib

Data summarizing the evolution of fake size data in Ponerinae ants using a 2-level factor as categorical trait

---

Description

A list containing fake size data of Ponerinae ants mapped on the phylogeny, modeled with geiger::fitDiscrete. This object was obtained with `prepare_trait_data()`. This is NOT real biological/ecological data. They were designed for illustrative purposes only. The phylogeny used is also NOT a properly time-calibrated phylogeny. It uses an ill-designed old calibration for illustrative purposes.

Usage

```
data(Ponerinae_cat_2lvl_data_old_calib)
```

Format

A list with 5 elements.

Details

A list of five objects containing information on the evolution of fake size data in Ponerinae ants. This object was obtained with `prepare_trait_data()`.

- `$densityMaps` List of two objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given state along branches. The list contains one `"densityMap"` per state found in the `tip_data` (i.e., "large" and "small").
- `$trait_data_type` Character string. Record the type of trait data. Here: "categorical".
- `$ace` Numeric matrix. Record the posterior probabilities of ancestral states (characters) estimates (ACE) at internal nodes. Rows are internal nodes. Columns are states (i.e., "large" and "small"). Values are posterior probabilities of each state per node.
- `$best_model_fit` List that provides the output of the best fitting model (Here: ARD model).
- `$model_selection_df` Data.frame that summarizes model comparisons used to select the best fitting model.

See Also

`prepare_trait_data()`

`Ponerinae_cat_3lvl_data_old_calib`

Data summarizing the evolution of fake habitat data in Ponerinae ants using a 3-level factor as categorical trait

---

Description

A list containing fake habitat data of Ponerinae ants mapped on the phylogeny, modeled with geiger::fitDiscrete. This object was obtained with `prepare_trait_data()`. This is NOT real biological/ecological data. They were designed for illustrative purposes only. The phylogeny used is also NOT a properly time-calibrated phylogeny. It uses an ill-designed old calibration for illustrative purposes.

Usage

```
data(Ponerinae_cat_3lvl_data_old_calib)
```

Format

A list with 5 elements.

Details

A list of five objects containing information on the evolution of fake habitat data in Ponerinae ants. This object was obtained with `prepare_trait_data()`.

- `$densityMaps` List of three objects of class `"densityMap"` that contains a phylogenetic tree and associated mapping of probability to harbor a given state along branches. The list contains one `"densityMap"` per state found in the `tip_data` (i.e., "arboreal", "subterranean", and "terricolous").
- `$trait_data_type` Character string. Record the type of trait data. Here: "categorical".
- `$ace` Numeric matrix. Record the posterior probabilities of ancestral states (characters) estimates (ACE) at internal nodes. Rows are internal nodes. Columns are states (i.e., "arboreal", "subterranean", and "terricolous"). Values are posterior probabilities of each state per node.
- `$best_model_fit` List that provides the output of the best fitting model (Here: ARD model).
- `$model_selection_df` Data.frame that summarizes model comparisons used to select the best fitting model.

See Also

`prepare_trait_data()`

`Ponerinae_trait_cont_tip_data_10My`

Data summarizing the evolution of a fake continuous trait in Ponerinae ants extracted for 10 Mya

---

Description

A list containing estimated values for a fake continuous trait mapped on the Ponerinae ant phylogeny, modeled with geiger::fitContinuous. This object was obtained with `extract_most_likely_trait_values_for_foc` applied on trait evolution data obtained with `prepare_trait_data()`. The phylogeny used is NOT a properly time-calibrated phylogeny. It uses an ill-designed old calibration for illustrative purposes.

Usage

```
data(Ponerinae_trait_cont_tip_data_10My)
```

Format

A list with 4 elements.

Details

A list of four elements containing information on the evolution of a continuous trait in Ponerinae ants extracted for 10 Mya.

- `$trait_data` Named numeric vector. Names are the taxa or internal tipward node ID associated with the values. Values are the continuous trait data estimated along branches for 10 Mya.

- $focal_time Numeric. Time in the past at which the trait data were extracted.
- $trait_data_type Character string. Record the type of trait data. Here: "continuous".
- $contMap A phylogenetic tree and associated mapping of estimated trait values. It was updated such that the tips correspond to lineages found 10 Mya (i.e., at the focal time in the past).

`Ponerinae_trait_tip_data` *Dataset providing fake trait data for extant ponerine ants for illustrative purposes*

Description

A data.frame of fake trait data covering the 1534 extant ponerine ant taxa (Ponerinae subfamily). This is NOT real biological/ecological data. They were designed for illustrative purposes only.

Usage

```
data(Ponerinae_trait_tip_data)
```

Format

A data.frame with 1534 rows and 4 columns.

Details

A data.frame of fake trait data covering the 1534 extant ponerine ant taxa (Ponerinae subfamily).

- $Taxa Character string. Names of Ponerinae ant taxa.
- $fake_cont_tip_data Numeric. Fake continuous trait data.
- $fake_cat_2lvl_tip_data Character vector. Fake categorical size data with two levels: "large" and "small".
- $fake_cat_3lvl_tip_data Character vector. Fake categorical habitat data with three levels: "arboreal", "subterranean", and "terricolous".

`Ponerinae_tree` *Dataset providing the extensive time-calibrated phylogeny of extant ponerine ants*

Description

A phylo object describing the time-calibrated phylogeny of the 1534 extant ponerine ants (Ponerinae subfamily).

Source: Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

Usage

```
data(Ponerinae_tree)
```

Format

A phylo object with 4 elements.

Details

A time-calibrated phylogeny as a phylo object with 4 elements.

- $edge Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- $edge.length Numeric vector. Length of edges/branches.
- $Nnode Integer. Number of internal nodes.
- $tip.label Character vector. Labels of all tips.

References

Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

Ponerinae_tree_old_calib

Dataset providing the extensive time-calibrated phylogeny of extant ponerine ants using an old calibration for illustrative purposes

---

Description

A phylo object describing the time-calibrated phylogeny of the 1534 extant ponerine ants (Ponerinae subfamily). This is NOT a properly time-calibrated phylogeny. It uses an ill-designed old calibration for illustrative purposes. For a proper time-calibrated phylogeny of ponerine ants, see the Ponerinae_tree object in deepSTRAPP.

Source: Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

Usage

```
data(Ponerinae_tree_old_calib)
```

Format

A phylo object with 4 elements.

Details

A time-calibrated phylogeny as a `phylo` object with 4 elements.

- `$edge` Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- `$edge.length` Numeric vector. Length of edges/branches.
- `$Nnode` Integer. Number of internal nodes.
- `$tip.label` Character vector. Labels of all tips.

References

Doré, M., Borowiec, M. L., Branstetter, M. G., Camacho, G. P., Fisher, B. L., Longino, J. T., Ward, P. S., Blaimer, B. B. (2025). Evolutionary history of ponerine ants highlights how the timing of dispersal events shapes modern biodiversity. Nature Communications, 16, 8297. doi:10.1038/s41467025637093

`prepare_diversification_data`

Run a full BAMM (Bayesian Analysis of Macroevolutionary Mixtures) workflow

---

Description

Run a full BAMM (Bayesian Analysis of Macroevolutionary Mixtures) workflow to produce a `BAMM_object` that contains a phylogenetic tree and associated diversification rates mapped along branches, across selected posterior samples:

- Step 1: Set BAMM - Record BAMM settings and generate all input files needed for BAMM.
- Step 2: Run BAMM - Run BAMM and move output files into a dedicated directory.
- Step 3: Evaluate BAMM - Produce evaluation plots and ESS data.
- Step 4: Import BAMM outputs - Load `BAMM_object` in R and subset posterior samples.
- Step 5: Clean BAMM files - Remove files generated during the BAMM run.

The `BAMM_object` output is typically used as input to run deepSTRAPP with `run_deepSTRAPP_for_focal_time()` or `run_deepSTRAPP_over_time()`. Diversification rates and regime shift can be visualized with `plot_BAMM_rates()`.

BAMM is a model of diversification for time-calibrated phylogenies that explores complex diversification dynamics by allowing multiple regime shifts across clades without a priori hypotheses on the location of such shifts. It uses reversible jump Markov chain Monte Carlo (rjMCMC) to automatically explore a vast range of models with different speciation and extinction rates, and different number and location of regime shifts.

This function will work only if you have the BAMM C++ program installed on your machine. See the BAMM website: `http://bamm-project.org/` and the companion R package {`BAMMtools`}.

Usage

```
prepare_diversification_data(
  BAMM_install_directory_path,
  phylo,
  prefix_for_files = NULL,
  seed = NULL,
  numberOfGenerations = 10^7,
  globalSamplingFraction = 1,
  sampleProbsFilename = NULL,
  expectedNumberOfShifts = NULL,
  eventDataWriteFreq = NULL,
  burn_in = 0.25,
  nb_posterior_samples = 1000,
  additional_BAMM_settings = list(),
  BAMM_output_directory_path = NULL,
  keep_BAMM_outputs = TRUE,
  MAP_odds_ratio_threshold = 5,
  skip_evaluations = FALSE,
  plot_evaluations = TRUE,
  save_evaluations = TRUE
)
```

Arguments

`BAMM_install_directory_path`
: Character string. The path to the directory where BAMM is. It can be an absolute path, or a path relative to the current working directory (Check with `getwd()` if needed).

`phylo`
: Time-calibrated phylogeny. Object of class `"phylo"` as defined in R package `{ape}`. The phylogeny must be rooted and fully resolved. BAMM does not currently work with fossils, so the tree must also be ultrametric.

`prefix_for_files`
: Character string. Prefix to add to all BAMM files stored in the `BAMM_output_directory_path` if `keep_BAMM_outputs = TRUE`. Files will be exported as 'prefix_*' with an underscore separating the prefix and the file name. Default is `NULL` (no prefix is added).

`seed`
: Integer. Set the seed to ensure reproducibility. Default is `NULL` (a random seed is used).

`numberOfGenerations`
: Integer. Number of steps in the MCMC run. It should be set high enough to reach the equilibrium distribution and allow posterior samples to be uncorrelated. Check the Effective Sample Size of parameters with coda::effectiveSize() in the Evaluation step. Default value is `10^7`.

`globalSamplingFraction`
: Numeric. Global sampling fraction representing the overall proportion of terminals in the phylogeny compared to the estimated overall richness in the clade. It acts as a multiplier on the rates needed to achieve such extant diversity. Default is `1.0` (assuming all taxa are in the phylogeny).

sampleProbsFilename
: Character string. The path to the .txt file used to provide clade-specific sampling fractions. See BAMMtools::samplingProbs() to generate such a file. If provided, globalSamplingFraction is ignored. Default = NULL.

expectedNumberOfShifts
: Integer. Set the expected number of regime shifts. It acts as a hyperparameter controlling the exponential prior distribution used to modulate reversible jumps across model configurations in the rjMCMC run. If set to NULL (default), an empirical rule will be used to define this value: 1 regime shift expected for every 100 tips in the phylogeny, with a minimum of 1. The best practice is to try several values and inspect the similarity of the prior and posterior distribution of the regime shift parameter. See BAMMtools::plotPrior() and the Evaluation step to produce such evaluation plot.

eventDataWriteFreq
: Integer. Set the frequency at which to write the event data to the output file = the sampling frequency of posterior samples. If set to NULL (default), will set the frequency such that 2000 posterior samples are recorded, i.e. eventDataWriteFreq = numberOfGenerations / 2000.

burn_in
: Numeric. Proportion of posterior samples removed from the BAMM output to ensure that the remaining samples were drawn once the equilibrium distribution was reached. This can be evaluated looking at the MCMC trace (see Evaluation step). Default is 0.25.

nb_posterior_samples
: Numeric. Number of posterior samples to extract, after removing the burn-in, in the final BAMM_object to use for downstream analyses. Default = 1000.

additional_BAMM_settings
: List of named elements. Additional settings options for BAMM provided as a list of named arguments. Ex: list(lambdaInit0 = 0.5, muInit0 = 0). See available settings in the template file provided within the deepSTRAPP package files as 'BAMM_template_diversification.txt'. The template can also be loaded directly in R with utils::data(BAMM_template_diversification) and displayed with print(BAMM_template_diversification).

BAMM_output_directory_path
: Character string. The path to the directory used to store input/output files generated. It can be an absolute path, or a path relative to the current working directory (Check with getwd() if needed).

keep_BAMM_outputs
: Logical. Whether the BAMM_output_directory should be kept after the run. Default = TRUE.

MAP_odds_ratio_threshold
: Numeric. Controls the definition of 'core-shifts' used to distinguish across configurations when fetching the MAP samples. Shifts that have an odds ratio of marginal posterior probability / prior lower than MAP_odds_ratio_threshold are ignored. See BAMMtools::getBestShiftConfiguration(). Default = 5.

skip_evaluations
: Logical. Whether to skip the Evaluation step including MCMC trace, ESS, and prior/posterior comparisons for expected number of shifts. Default = FALSE.

`plot_evaluations`
: Logical. Whether to display the plots generated during the Evaluation step: MCMC trace, and prior/posterior comparisons for expected number of shifts. Default = `TRUE`.

`save_evaluations`
: Logical. Whether to save the outputs of evaluations in a table (ESS), and PDFs (MCMC trace, and prior/posterior comparisons for expected number of shifts) in the `BAMM_output_directory`. Default = `TRUE`.

Details

This function runs a full BAMM (Bayesian Analysis of Macroevolutionary Mixtures) workflow to produce a `BAMM_object` that contains a phylogenetic tree and associated diversification rates mapped along branches, across selected posterior samples.

Step 1: Set BAMM

- Produces a tree file for the phylogeny. Default file: 'phylogeny.tree'.
- Save configuration settings used for the BAMM run. Default file: 'config_file.txt'.
- Save default priors generated by BAMMtools::setBAMMpriors based on the phylogeny. Default file: 'priors.txt'.

Step 2: Run BAMM

- Run BAMM using the system console
- Move output files in dedicated `BAMM_output_directory`. Default directory is `./BAMM_outputs/`.
  - 'run_info.txt' containing a summary of your parameters/settings.
  - 'mcmc_log.txt' containing raw MCMC information useful in diagnosing convergence.
  - 'event_data.txt' containing all evolutionary rate parameters and their topological mappings.
  - 'chain_swap.txt' containing data about each chain swap proposal (when a proposal occurred, which chains might be swapped, and whether the swap was accepted).
  - 'acceptance_info.txt' containing the history of acceptance/proposal of MCMC steps (If additional setting `outputAcceptanceInfo` is set to 1).

Step 3: Evaluate BAMM

- Plot the MCMC trace = evolution of logLik across MCMC generations. Output file = 'MCMC_trace_logLik.pdf'.
- Compute the Effective Sample Size (ESS) across posterior samples (after removing burn-in) using `coda::effectiveSize()`. This is a way to evaluate if your MCMC runs have enough generations to produce robust estimates. Ideally, ESS should be higher than 200. Output file = 'ESS_df.csv'.
- Plot the comparison of prior and posterior distributions of the number of regime shifts with BAMMtools::plotPrior. Output file = 'PP_nb_shifts_plot.pdf'. A good value for `expectedNumberOfShifts` is one with high similarities between the distributions hinting that the information in the data coincides with your expectations for the number of regime shifts. The best practice is to try several values to check whether it affects the final output.

Step 4: Import BAMM outputs

- Load BAMM outputs with BAMMtools::getEventData.
- Subset posterior samples to the requested nb_posterior_samples with BAMMtools::subsetEventData.
- Record the $expectedNumberOfShifts used to set the prior. This is useful for downstream analyses involving comparison of prior vs. posterior probabilities (See BAMMtools::distinctShiftConfigurations(
- Record the marginal posterior probability of regime shift along branches based on the proportion of samples harboring a regime shift along each branch. (See BAMMtools::marginalShiftProbsTree()). Result is stored in $MSP_tree as phylogenetic tree with $edge.length scaled to the marginal posterior probability.
- Extract the Maximum A Posteriori probability (MAP) configuration = the configuration of regime shift location found the most frequently among the posterior samples. (See BAMMtools::getBestShiftConfigu This ignores shifts that have an odds ratio of marginal posterior probability / prior lower than MAP_odds_ratio_threshold to avoid noise from non-core shifts. MAP sample indices are stored in $MAP_indices. Diversification rates and shift locations on branches are then averaged across all MAP samples and recorded as an object of class "bammdata" in $MAP_BAMM_object with a single $eventData table used to plot regime shifts on the phylogeny with plot_BAMM_rates().
- Extract the Maximum Shift Credibility (MSC) configuration = the configuration of regime shift location with the highest product of marginal probabilities across branches. (See BAMMtools::maximumShiftCred MSC sample indices are stored in $MSC_indices. Diversification rates and shift locations on branches are then averaged across all MSC samples and recorded as an object of class "bammdata" in $MSC_BAMM_object with a single $eventData table used to plot regime shifts on the phylogeny with plot_BAMM_rates().

Step 5: Clean BAMM files

- Remove files generated in Steps 1 & 2 if keep_BAMM_outputs = FALSE.
- Delete the BAMM_output_directory if empty after cleaning files.

The BAMM_object output:

- is typically used as input to run deepSTRAPP with run_deepSTRAPP_for_focal_time() or run_deepSTRAPP_over_time().
- can be used to extract rates and regimes for any focal_time in the past with update_rates_and_regimes_for_focal_
- can be used to map diversification rates and regime shifts on the phylogeny with plot_BAMM_rates().

Value

The function returns a BAMM_object of class "bammdata" which is a list with at least 22 elements.

Phylogeny-related elements used to plot a phylogeny with ape::plot.phylo():

- $edge Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- $Nnode Integer. Number of internal nodes.
- $tip.label Character vector. Labels of all tips.
- $edge.length Numeric vector. Length of edges/branches.
- $node.label Character vector. Labels of all internal nodes. (Present only if present in the initial BAMM_object)

BAMM internal elements used for tree exploration:

- `$begin` Numeric vector. Absolute time since root of edge/branch start (rootward).
- `$end` Numeric vector. Absolute time since root of edge/branch end (tipward).
- `$downseq` Integer vector. Order of node visits when using a pre-order tree traversal.
- `$lastvisit` ID of the last node visited when starting from the node in the corresponding position in `$downseq`.

BAMM elements summarizing diversification data:

- `$numberEvents` Integer vector. Number of events/macroevolutionary regimes (k+1) recorded in each posterior configuration. k = number of shifts.
- `$eventData` List of data.frames. One per posterior sample. Records shift events and macroevolutionary regimes parameters. 1st line = Background root regime.
- `$eventVectors` List of integer vectors. One per posterior sample. Record regime ID per branch.
- `$tipStates` List of named integer vectors. One per posterior sample. Record regime ID per tip.
- `$tipLambda` List of named numeric vectors. One per posterior sample. Record speciation rates per tip.
- `$tipMu` List of named numeric vectors. One per posterior sample. Record extinction rates per tip.
- `$eventBranchSegs` List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- `$meanTipLambda` Named numeric vector. Mean tip speciation rates across all posterior configurations of tips.
- `$meanTipMu` Named numeric vector. Mean tip extinction rates across all posterior configurations of tips.
- `$type` Character string. Set the type of data modeled with BAMM. Should be "diversification".

Additional elements providing key information for downstream analyses:

- `$expectedNumberOfShifts` Integer. The expected number of regime shifts used to set the prior in BAMM.
- `$MSP_tree` Object of class `phylo`. List of 4 elements duplicating information from the Phylogeny-related elements above, except `$MSP_tree$edge.length` is recording the Marginal Shift Probability of each branch (i.e., the probability of a regime shift to occur along each branch)
- `$MAP_indices` Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- `$MAP_BAMM_object`. List of 18 elements of class `"bammdata"` recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.
- `$MSC_indices` Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.
- `$MSC_BAMM_object` List of 18 elements of class `"bammdata"` recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations. All

BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.

The function also produces files listed in the Details section and stored in the `BAMM_output_directory`.

Note on diversification models for time-calibrated phylogenies

This function relies on BAMM to provide a reliable solution to map diversification rates and regime shifts on a time-calibrated phylogeny and obtain the `BAMM_object` object needed to run the deepSTRAPP workflow (run_deepSTRAPP_for_focal_time, run_deepSTRAPP_over_time). BAMM is one option among others for modeling diversification on phylogenies. You may wish to explore alternative models such as the LSBDS model in RevBayes (Höhna et al., 2016), the MTBD model (Barido-Sottani et al., 2020), or the ClaDS2 model (Maliet et al., 2019) for your own data. However, you will need Bayesian models that infer regime shifts to be able to perform STRAPP tests (Rabosky & Huang, 2016). Additionally, you need to format the model output such as in `BAMM_object`, so it can be used in a deepSTRAPP workflow.

This function performs a single BAMM run to infer diversification rates and regime shifts. Due to the stochastic nature of the exploration of the parameter space with MCMC process, best practice recommends running multiple runs and check for convergence of the MCMC traces, ensuring that the region of high probability has been reached by your MCMC runs.

Author(s)

Maël Doré

References

For BAMM: Rabosky, D. L. (2014). Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. PloS one, 9(2), e89543. doi:10.1371/journal.pone.0089543. Website: http://bamm-project.org/.

For {BAMMtools}: Rabosky, D. L., Grundler, M., Anderson, C., Title, P., Shi, J. J., Brown, J. W., ... & Larson, J. G. (2014). BAMM tools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. Methods in Ecology and Evolution, 5(7), 701-707. doi:10.1111/2041210X.12199

See Also

run_deepSTRAPP_for_focal_time() run_deepSTRAPP_over_time() update_rates_and_regimes_for_focal_time() prepare_trait_data() plot_BAMM_rates()

For a guided tutorial, see this vignette: `vignette("model_diversification_dynamics", package = "deepSTRAPP")`

Examples

```
# ----- Example 1: Whale phylogeny ----- #

library(phytools)
data(whale.tree)
```

```
## Not run:
## You need to install the BAMM C++ software locally prior to run this function
# Visit the official BAMM website (\url{http://bamm-project.org/}) for information.

# Run BAMM workflow with deepSTRAPP
whale_BAMM_object <- prepare_diversification_data(
   BAMM_install_directory_path = "./software/bamm-2.5.0/",
   phylo = whale.tree,
   prefix_for_files = "whale",
   numberOfGenerations = 100000, # Set low for the example
   BAMM_output_directory_path = tempdir(), # Can be adjusted such as "./BAMM_outputs/"
   keep_BAMM_outputs = FALSE, # Adjust if needed
)
## End(Not run)

# Load directly the result
data(whale_BAMM_object)

# Explore output
str(whale_BAMM_object, 1)

# Plot mean net diversification rates and regime shifts on the phylogeny
plot_BAMM_rates(whale_BAMM_object, cex = 0.5,
                labels = TRUE, legend = TRUE)

# ----- Example 2: Ponerinae phylogeny ----- #

#   Load phylogeny
data("Ponerinae_tree", package = "deepSTRAPP")
plot(Ponerinae_tree, show.tip.label = FALSE)

## Not run:
## You need to install the BAMM C++ software locally prior to run this function
# Visit the official BAMM website (http://bamm-project.org/) for information.

# Run BAMM workflow with deepSTRAPP
Ponerinae_BAMM_object <- prepare_diversification_data(
   BAMM_install_directory_path = "./software/bamm-2.5.0/",
   phylo = Ponerinae_tree,
   prefix_for_files = "Ponerinae",
   numberOfGenerations = 10^7, # Set high for optimal run, but will take ages
   BAMM_output_directory_path = tempdir(), # Can be adjusted such as "./BAMM_outputs/"
   keep_BAMM_outputs = FALSE, # Adjust if needed
)
## End(Not run)

if (deepSTRAPP::is_dev_version())
{
 # Load directly the result
 data(Ponerinae_BAMM_object)
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.
```

```
 # Explore output
 str(Ponerinae_BAMM_object, 1)

 # Plot mean net diversification rates and regime shifts on the phylogeny
 plot_BAMM_rates(Ponerinae_BAMM_object,
                 labels = FALSE, legend = TRUE)
}
```

`prepare_trait_data` *Map trait evolution on a time-calibrated phylogeny*

Description

Map trait evolution on a time-calibrated phylogeny in several steps:

- Step 1: Fit evolutionary models to trait data using Maximum Likelihood.
- Step 2: Select the best fitting model comparing AICc.
- Step 3: Infer ancestral character estimates (ACE) at nodes.
- Step 4: Run stochastic mapping simulations to generate evolutionary histories compatible with the best model and inferred ACE. (Only for categorical and biogeographic data)
- Step 5: Infer ancestral states along branches.
  - For continuous traits: use interpolation to produce a `contMap`.
  - For categorical and biogeographic data: compute posterior frequencies of each state/range to produce a `densityMap` for each state/range.

Usage

```
prepare_trait_data(
  tip_data,
  trait_data_type,
  phylo,
  seed = NULL,
  evolutionary_models = NULL,
  Q_matrix = NULL,
  BioGeoBEARS_directory_path = NULL,
  keep_BioGeoBEARS_files = TRUE,
  prefix_for_files = NULL,
  nb_cores = 1,
  max_range_size = 2,
  split_multi_area_ranges = FALSE,
  ...,
  res = 100,
  run_stochastic_maps = FALSE,
  nb_simulations = 100,
```

```
  color_scale = NULL,
  colors_per_levels = NULL,
  plot_map = TRUE,
  plot_overlay = TRUE,
  add_ACE_pies = TRUE,
  PDF_file_path = NULL,
  return_ace = TRUE,
  return_BSM = FALSE,
  return_simmaps = FALSE,
  return_best_model_fit = FALSE,
  return_model_selection_df = FALSE,
  verbose = TRUE
)
```

Arguments

`tip_data`	Named numeric or character vector of trait values/states/ranges at tips. Names should be ordered as the tip labels in the phylogeny found in `phylo$tip.label`. For biogeographic data, ranges should follow the coding scheme of BioGeoBEARS with a unique CAPITAL letter per unique area (ex: A, B), combined to form multi-area ranges (Ex: AB). Alternatively, you can provide tip_data as a matrix or data.frame of binary presence/absence in each area (coded as unique CAPITAL letter). In this case, columns are unique areas, rows are taxa, and values are integer (0/1) signaling absence or presence of the taxa in the area.
`trait_data_type`	Character string. Type of trait data. Either: "continuous", "categorical" or "biogeographic".
`phylo`	Time-calibrated phylogeny. Object of class `"phylo"` as defined in `{ape}`. Tip labels (`phylo$tip.label`) should match names in `tip_data`.
`seed`	Integer. Set the seed to ensure reproducibility. Default is `NULL` (a random seed is used).
`evolutionary_models`	(Vector of) character string(s). To provide the set of evolutionary models to fit on the data. • Models available for continuous data are detailed in `geiger::fitContinuous()`. Default is `"BM"`. • Models available for categorical data are detailed in `geiger::fitDiscrete()`. Default is `"ARD"`. • Models for biogeographic data are fit with R package `BioGeoBEARS` using `BioGeoBEARS::bears_optim_run()`. Default is `"DEC"`. • See list in "Details" section.
`Q_matrix`	Custom Q-matrix for categorical data representing transition classes between states. Transitions with similar integers are estimated with a shared rate parameter. Transitions with `0` represent rates that are fixed to zero (i.e., impossible transitions). Diagonal must be populated with `NA`. `row.names(Q_matrix)` and `col.names(Q_matrix)` are the states. Provide `"matrix"` among the models

listed in 'evolutionary_models' to use the custom Q-matrix for modeling. Only for categorical data.

`BioGeoBEARS_directory_path`
: Character string. The path to the directory used to store input/output files generated for/by BioGeoBEARS during biogeographic historical inferences. Only for biogeographic data.

`keep_BioGeoBEARS_files`
: Logical. Whether the `BioGeoBEARS_directory` and its content should be kept after the run. Default = `TRUE`. Only for biogeographic data.

`prefix_for_files`
: Character string. Prefix to add to all BioGeoBEARS files stored in the `BioGeoBEARS_directory_path` if `keep_BioGeoBEARS_files = TRUE`. Files will be exported such as 'prefix_*' with an underscore separating the prefix and the file name. Default is `NULL` (no prefix is added). Only for biogeographic data.

`nb_cores`
: Integer. Number of cores to use for parallel computation during BioGeoBEARS runs. Default = `1`. Only for biogeographic data.

`max_range_size`
: Integer. Maximum number of unique areas encompassed by multi-area ranges. Default = `2`. Only for biogeographic data.

`split_multi_area_ranges`
: Logical. Whether to split multi-area ranges across unique areas when mapping ranges. Ex: For range EW, posterior probabilities will be split equally between Eastern Palearctic (E) and Western Palearctic (W). Default = `FALSE`. Only for biogeographic data.

`...`
: Additional arguments to be passed down to the functions used to fit models (See `evolutionary_models`) and produce simmaps with `phytools::make.simmap()` or `BioGeoBEARS::runBSM()`.

`res`
: Integer. Define the number of time steps used to interpolate/estimate trait value/state/range in `contMap`/`densityMaps`. Default = `100`.

`run_stochastic_maps`
: Logical. Whether to perform continuous stochastic mapping to account for uncertainty in ancestral trait estimates. Only for continuous data (This is always 'TRUE' for categorical and biogeographic data). This functionality is currently not available for total-evidence phylogenies (i.e., including fossils as tips).

`nb_simulations`
: Integer. Define the number of simulations generated for stochastic mapping. Default = `100`.

`color_scale`
: Character vector. List of colors to use to build the color scale with `grDevices::colorRampPalette()` showing the evolution of a continuous trait on the `contMap`. From lowest values to highest values. Only for continuous data. Default = `NULL` will use the rainbow() color palette.

`colors_per_levels`
: Named character string. To set the colors to use to map each state/range posterior probabilities. Names = states/ranges; values = colors. If `NULL` (default), the `rainbow()` color scale will be used for categorical trait; the `BioGeoBEARS::get_colors_for_states_li` will be used for biogeographic ranges. If no color is provided for multi-area ranges, they will be interpolated based on the colors provided for unique areas. Only for categorical and biogeographic data.

`plot_map` Logical. Whether to plot or not the phylogeny with mapped trait evolution. Default = `TRUE`.

`plot_overlay` Logical. If `TRUE` (default), plot a unique `densityMap` with overlapping states/ranges using transparency. If `FALSE`, plot a `densityMap` per state/range. Only for "categorical" and "biogeographic" data.

`add_ACE_pies` Logical. Whether to add pies of posterior probabilities of states/ranges at internal nodes on the mapped phylogeny. Default = `TRUE`. Only for categorical and biogeographic data.

`PDF_file_path` Character string. If provided, the plot will be saved in a PDF file following the path provided here. The path must end with ".pdf".

`return_ace` Logical. Whether the named vector of ancestral character estimates (ACE) at internal nodes should be returned in the output. Default = `TRUE`.

`return_BSM` Logical. (Only for Biogeographic data) Whether the summary tables of anagenetic and cladogenetic events generated during the Biogeographic Stochastic Mapping (BSM) process should be returned in the output. Default = `FALSE`.

`return_simmaps` Logical. Whether the evolutionary histories simulated during stochastic mapping (i.e., `simmaps`) should be returned in the output. Default = `TRUE`. Only for "categorical" and "biogeographic" data. This is needed to be able to track which simulated histories provided which trait data in downstream analyses, although it may create voluminous objects.

`return_best_model_fit` Logical. Whether to include the output of the best fitting model in the function output. Default = `FALSE`.

`return_model_selection_df` Logical. Whether to include the data.frame summarizing model comparisons used to select the best fitting model should be returned in the output. Default = `FALSE`.

`verbose` Logical. Should progression be displayed? A message will be printed for every step in the process. Default is `TRUE`.

Details

Map trait evolution on a time-calibrated phylogeny in several steps:

Step 1: Models are fit using a Maximum Likelihood approach:

- For "continuous" data models are fit with `geiger::fitContinuous()`: "BM", "OU", "EB", "rate_trend", "lambda", "kappa", "delta". Default is `"BM"`.
- For "categorical" data models are fit with `geiger::fitDiscrete()`: "ER", "SYM", "ARD", "meristic", "matrix". Default is `"ARD"`.
- For "biogeographic" data models are fit with R package `BioGeoBEARS`: "BAYAREALIKE", "DIVALIKE", "DEC", "BAYAREALIKE+J", "DIVALIKE+J", "DEC+J". Default is `"DEC"`.

Step 2: Best model is identified among the list of `evolutionary_models` by comparing the corrected AIC (AICc) and selecting the model with lowest AICc.

Step 3: For continuous traits: Ancestral character estimates (ACE) are inferred with phytools::fastAnc on a tree with modified branch lengths scaled to reflect the evolutionary rates estimated from the best model using `phytools::rescale()`.

Step 4: Stochastic Mapping.

For categorical and biogeographic data, stochastic mapping simulations are performed by default to generate evolutionary histories compatible with the best model and inferred ACE. Node states/ranges are drawn from the scaled marginal likelihoods of ACE, and state/range shifts along branches are simulated according to the transition matrix Q estimated from the best fitting model.

For continuous trait data, stochastic mapping simulations are performed only if `run_stochastic_maps = TRUE`. Evolutionary histories of continuous trait evolution, conditioned on the observed trait data and model fit, including estimates of ancestral trait values and variance at nodes, are produced with `contsimmap::make.contsimmap()`.

Step 5: Infer ancestral states along branches.

- For continuous traits:
  - Ancestral values along branches are interpolated using `phytools::contMap()`. This provides quick estimates of trait value at any point in time, but does not account for uncertainty in trait estimates. Moreover, it does not provide fully accurate ML estimates in case of models that are time or trait-value dependent (such as "EB" or "OU") as the interpolation used to build the contMap is assuming a constant rate along each branch. However, ancestral trait values at nodes remain accurate. It produces a single `$contMap` representing the ML estimates of ancestral trait evolution across the phylogeny.
  - If `run_stochastic_maps = TRUE`, ancestral values along branches are recorded across all simulated evolutionary histories (i.e., continuous stochastic maps), thus accounting for uncertainty in trait estimates. This is needed to apply the "paired" or "full" strategies to account for trait estimate uncertainty in deepSTRAPP tests (See the uncertainty_strategy argument in run_deepSTRAPP_* functions). It produces a list of `$contMaps`, each map representing a simulated ancestral trait evolution across the phylogeny. This functionality is currently not available for total-evidence phylogenies (i.e., including fossils as tips)
- For categorical and biogeographic data: posterior frequencies of each state/range among the simulated evolutionary histories (`simmaps`) are computed to produce a list of `$densityMaps`, one for each state/range, reflecting the changes along branches in probability of harboring a given state/range.

Value

The function returns a list with at least three elements.

- `$contMap` (For "continuous" data) Object of class `"contMap"`, typically generated with `phytools::contMap()`, that contains a phylogenetic tree and a unique continuous trait mapping representing the interpolated ML estimates of ancestral trait evolution across the phylogeny.
- `$contMaps` (For "continuous" data, with `run_stochastic_maps = TRUE`) List of objects of class `"contMap"`, each map representing a simulated ancestral trait evolution conditioned on the observed trait data and model fit.
- `$densityMaps` (For "categorical" and "biogeographic" data) List of objects of class `"densityMap"`, typically generated with `phytools::densityMap()`, that contains a phylogenetic tree and associated mapping of probability to harbor a given state/range along branches. The list contains one `"densityMap"` per state/range found in the `tip_data`.

- $densityMaps_all_ranges (For "biogeographic" data only, if split_multi_area_ranges = TRUE) Same as $densityMaps, but for all ranges including the multi-area ranges (e.g., AB) while $densityMaps will display posterior probabilities for unique areas only (e.g., A and B), with multi-area ranges split across the unique areas they encompass.
- $trait_data_type Character string. Record the type of trait data. Either: "continuous", "categorical" or "biogeographic".
- $nb_simulations Integer. The number of simulations / stochastic maps produced.

If return_ace = TRUE,

- $ace For continuous traits: Named vector that records the ancestral character estimates (ACE) at internal nodes. For categorical and biogeographic data: Matrix that records the posterior probabilities of ancestral states/ranges (characters) estimates (ACE) at internal nodes. Rows are internal nodes. Columns are states/ranges. Values are posterior probabilities of each state per node.
- $ace_all_ranges For biogeographic data, if split_multi_area_ranges = TRUE: Named vector that records the ancestral character estimates (ACE) at internal nodes, but including all ranges observed in the simmaps, including the multi-area ranges (e.g., AB), while $ace will display posterior probabilities for unique areas only (e.g., A and B), with multi-area ranges split across the unique areas they encompass.

If return_BSM = TRUE, (Only for biogeographic data)

- $BSM_output List of two lists that contains summary information of cladogenetic ($RES_caldo_events_tables) and anagenetic ($RES_ana_events_tables) events recording across the N simulations of biogeographic histories performed during Biogeographic Stochastic Mapping (BSM). Each element of the list is a data.frame recording events occurring during one simulation.

If return_simmaps = TRUE, (Only for categorical and biogeographic data)

- $simmaps List that contains as many objects of class "simmap" that nb_simulations were requested. Each simmap object is a phylogeny with one simulated discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches. This is needed to be able to track which simulated history provided which trait data in downstream analyses, although it may create voluminous objects.

If return_best_model_fit = TRUE,

- $best_model_fit List that provides the output of the best fitting model.

If model_selection_df = TRUE,

- $model_selection_df Data.frame that summarizes model comparisons used to select the best fitting model.

For biogeographic data, the function also produces input and output files associated with BioGeoBEARS and stored in the directory specified in BioGeoBEARS_directory_path. The directory and its content are kept if keep_BioGeoBEARS_files = TRUE

Note on macroevolutionary models of trait evolution

This function provides an easy solution to map trait evolution on a time-calibrated phylogeny and obtain the `contMaps`/`densityMaps` objects needed to run the deepSTRAPP workflow (run_deepSTRAPP_for_focal_time, run_deepSTRAPP_over_time). However, it does not explore the most complex options for trait evolution. You may need to explore more complex models to capture the dynamics of trait evolution. such as trait-dependent multi-rate models (`phytools::brownie.lite()`, `OUwie::OUwie()`), Bayesian MCMC implementations allowing a thorough exploration of location and number of regime shifts (Ex: BayesTraits, RevBayes), or RRphylo for a penalized phylogenetic ridge regression approach that allows regime shifts across all branches.

Note on macroevolutionary models of biogeographic history

This function provides an easy solution to infer ancestral geographic ranges using the BioGeoBEARS framework. It allows you to directly compare model fits across 6 models: DEC, DEC+J, DIVALIKE, DIVALIKE+J, BAYAREALIKE, BAYAREALIKE+J. It uses a step-wise approach by using MLE estimates of previous runs as starting parameter values when increasing complexity (adding +J parameters). However, it does not explore the most complex options for historical biogeography. You may need to explore more complex models to capture the dynamics of range evolution such as time-stratification with adjacency matrices, dispersal multipliers (+W), distance-based dispersal probabilities (+X), or other features. See for instance, http://phylo.wikidot.com/biogeobears.

The R package `BioGeoBEARS` is needed for this function to work with biogeographic data. Please install it manually from: https://github.com/nmatzke/BioGeoBEARS.

Author(s)

Maël Doré

References

For macroevolutionary models in geiger: Pennell, M. W., Eastman, J. M., Slater, G. J., Brown, J. W., Uyeda, J. C., FitzJohn, R. G., ... & Harmon, L. J. (2014). geiger v2. 0: an expanded suite of methods for fitting macroevolutionary models to phylogenetic trees. Bioinformatics, 30(15), 2216-2218. doi:10.1093/bioinformatics/btu181.

For BioGeoBEARS: Matzke, Nicholas J. (2018). BioGeoBEARS: BioGeography with Bayesian (and likelihood) Evolutionary Analysis with R Scripts. version 1.1.1, published on GitHub on November 6, 2018. doi:10.5281/zenodo.1478250. Website: `http://phylo.wikidot.com/biogeobears`.

For continuous stochastic mapping in contsimmap: Martin, B. S., & Weber, M. G. (2026). Stochastic character mapping of continuous traits on phylogenies. Systematic Biology, syag031. doi:10.1093/sysbio/syag031.

See Also

`geiger::fitContinuous()` `geiger::fitDiscrete()` `phytools::contMap()` `phytools::densityMap()`

For a guided tutorial, see the associated vignettes:

- For continuous trait data: `vignette("model_continuous_trait_evolution", package = "deepSTRAPP")`

- For categorical trait data: `vignette("model_categorical_trait_evolution", package = "deepSTRAPP")`
- For biogeographic range data: `vignette("model_biogeographic_range_evolution", package = "deepSTRAPP")`

Examples

```
# ----- Example 1: Continuous data ----- #

## Load phylogeny and tip data
# Load eel phylogeny and tip data from the R package phytools
# Source: Collar et al., 2014; DOI: 10.1038/ncomms6505
data("eel.tree", package = "phytools")
data("eel.data", package = "phytools")

# Extract body size
eel_data <- stats::setNames(eel.data$Max_TL_cm,
                            rownames(eel.data))

 # (May take several minutes to run)
## Map trait evolution on the phylogeny
mapped_cont_traits <- prepare_trait_data(
   tip_data = eel_data,
   trait_data_type = "continuous",
   phylo = eel.tree,
   evolutionary_models = c("BM", "OU", "lambda", "kappa"),
  # Example of an additional argument ('control') that can be provided to geiger::fitContinuous()
   control = list(niter = 200),
   color_scale = c("darkgreen", "limegreen", "orange", "red"),
   plot_map = FALSE,
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   verbose = TRUE)

## Explore output
plot_contMap(mapped_cont_traits$contMap) # contMap with interpolated trait values
mapped_cont_traits$model_selection_df # Summary of model selection
# Parameter estimates and optimization summary of the best model
# (Here, the best model is Pagel's lambda)
mapped_cont_traits$best_model_fit$opt
mapped_cont_traits$ace # Ancestral character estimates at internal nodes

# ----- Example 2: Categorical data ----- #

## Load phylogeny and tip data
# Load eel phylogeny and tip data from the R package phytools
# Source: Collar et al., 2014; DOI: 10.1038/ncomms6505
data("eel.tree", package = "phytools")
data("eel.data", package = "phytools")

# Transform feeding mode data into a 3-level factor
eel_data <- stats::setNames(eel.data$feed_mode, rownames(eel.data))
```

```
eel_data <- as.character(eel_data)
eel_data[c(1, 5, 6, 7, 10, 11, 15, 16, 17, 24, 25, 28, 30, 51, 52, 53, 55, 58, 60)] <- "kiss"
eel_data <- stats::setNames(eel_data, rownames(eel.data))
table(eel_data)

# Manually define a Q_matrix for rate classes of state transition to use in the 'matrix' model
# Does not allow transitions from state 1 ("bite") to state 2 ("kiss") or state 3 ("suction")
# Does not allow transitions from state 3 ("suction") to state 1 ("bite")
# Set symmetrical rates between state 2 ("kiss") and state 3 ("suction")
Q_matrix = rbind(c(NA, 0, 0), c(1, NA, 2), c(0, 2, NA))

# Set colors per state
colors_per_states <- c("limegreen", "orange", "dodgerblue")
names(colors_per_states) <- c("bite", "kiss", "suction")

 # (May take several minutes to run)
## Run evolutionary models
eel_cat_3lvl_data <- prepare_trait_data(tip_data = eel_data, phylo = eel.tree,
    trait_data_type = "categorical",
    colors_per_levels = colors_per_states,
    evolutionary_models = c("ER", "SYM", "ARD", "meristic", "matrix"),
    Q_matrix = Q_matrix,
    nb_simulations = 1000,
    plot_map = TRUE,
    plot_overlay = TRUE,
    return_best_model_fit = TRUE,
    return_model_selection_df = TRUE)

# Load directly output
data(eel_cat_3lvl_data, package = "deepSTRAPP")

## Explore output
plot(eel_cat_3lvl_data$densityMaps[[1]]) # densityMap for state n°1 ("bite")
eel_cat_3lvl_data$model_selection_df # Summary of model selection
# Parameter estimates and optimization summary of the best model
# (Here, the best model is ER)
print(eel_cat_3lvl_data$best_model_fit)$ # Summary of the best evolutionary model
eel_cat_3lvl_data$ace # Posterior probabilities of each state (= ACE) at internal nodes


# ----- Example 3: Biogeographic data ----- #

if (deepSTRAPP::is_dev_version())
{
 ## The R package 'BioGeoBEARS' is needed for this function to work with biogeographic data.
 # Please install it manually from: https://github.com/nmatzke/BioGeoBEARS.

 ## Load phylogeny and tip data
 # Load eel phylogeny and tip data from the R package phytools
 # Source: Collar et al., 2014; DOI: 10.1038/ncomms6505
 data("eel.tree", package = "phytools")
 data("eel.data", package = "phytools")
```

```
# Transform feeding mode data into biogeographic data with ranges A, B, and AB.
eel_data <- stats::setNames(eel.data$feed_mode, rownames(eel.data))
eel_data <- as.character(eel_data)
eel_data[eel_data == "bite"] <- "A"
eel_data[eel_data == "suction"] <- "B"
eel_data[c(5, 6, 7, 15, 25, 32, 33, 34, 50, 52, 57, 58, 59)] <- "AB"
eel_data <- stats::setNames(eel_data, rownames(eel.data))
table(eel_data)

colors_per_ranges <- c("dodgerblue3", "gold")
names(colors_per_ranges) <- c("A", "B")

 # (May take several minutes to run)

# Load ape and BioGeoBEARS to use models internally
library(ape)
library(BioGeoBEARS)

## Run evolutionary models
eel_biogeo_data <- prepare_trait_data(
   tip_data = eel_data,
   trait_data_type = "biogeographic",
   phylo = eel.tree,
   # Default = "DEC" for biogeographic
   evolutionary_models = c("BAYAREALIKE", "DIVALIKE", "DEC",
                           "BAYAREALIKE+J", "DIVALIKE+J", "DEC+J"),
   BioGeoBEARS_directory_path = tempdir(), # Ex: "./BioGeoBEARS_directory/"
   keep_BioGeoBEARS_files = FALSE,
   prefix_for_files = "eel",
   max_range_size = 2,
   split_multi_area_ranges = TRUE, # Set to TRUE to display the two outputs
   # Reduce the number of Stochastic Mapping simulations to save time (Default = '1000')
   nb_simulations = 100,
   colors_per_levels = colors_per_ranges,
   return_simmaps = TRUE,
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   verbose = TRUE)

# Load directly output
data(eel_biogeo_data, package = "deepSTRAPP")

## Explore output
str(eel_biogeo_data, 1)
eel_biogeo_data$model_selection_df # Summary of model selection
# Parameter estimates and optimization summary of the best model
# (Here, the best model is DEC+J)
eel_biogeo_data$best_model_fit$optim_result

# Posterior probabilities of each state (= ACE) at internal nodes
eel_biogeo_data$ace # Only with unique areas
eel_biogeo_data$ace_all_ranges # Including multi-area ranges (Here, AB)
```

```
 ## Plot densityMaps
 # densityMap for range n°1 ("A")
 plot(eel_biogeo_data$densityMaps[[1]])
 # densityMaps with all unique areas overlaid
 plot_densityMaps_overlay(eel_biogeo_data$densityMaps)
 # densityMaps with all ranges (including multi-area ranges) overlaid
 plot_densityMaps_overlay(eel_biogeo_data$densityMaps_all_ranges)
}
```

`prune_BAMM_object` *Prune a BAMM object to a subset of tips*

Description

Prune a `BAMM_object` of class `bammdata` down to a subset of tips, and update all internal elements so that the pruned object still describes the BAMM diversification dynamics, including all deepSTRAPP elements, but only for the retained tips and branches.

The `BAMM_object` is typically generated directly with `prepare_diversification_data()` or from external BAMM output files with `build_BAMM_object()`.

This function is an extension of the original function `BAMMtools::subtreeBAMM()`, which is designed to extract a subclade. However, this new function also accepts any arbitrary set of tips, whether or not it forms a monophyletic group. When tips are removed, internal nodes that are left with no descendant are removed, and internal nodes left with a single descendant are suppressed, their parent and child branches being merged into a single branch.

All BAMM elements are updated accordingly:

- regime shifts located on removed branches are dropped,
- regime shifts located on merged internal branches are re-attached to the new merged branch that contains them,
- macroevolutionary regimes are re-indexed, and branch segments are rebuilt so that each merged branch carries all the regimes it went through.

This function also preserves and updates the additional deepSTRAPP elements:

- the Marginal Shift Probability (MSP) = the probability of a regime shift to occur along each branch.
- the Maximum A Posteriori probability (MAP) configurations among the posterior samples = the configurations of regime shifts that were sampled most frequently (See `BAMMtools::getBestShiftConfiguration`
- the Maximum Shift Credibility (MSC) configurations among the posterior samples = the configurations of regime shift location with the highest product of marginal probabilities across branches (See `BAMMtools::maximumShiftCredibility()`).

Pruning is intended to restrict a downstream deepSTRAPP analysis (e.g., to the taxa for which trait or range data are available) while keeping the diversification dynamics inferred on the full phylogeny.

If you need diversification rates estimated for a specific clade or set of taxa, you should run a dedicated BAMM analysis on that subset with appropriate sampling fractions, see `prepare_diversification_data()`.

Usage

```
prune_BAMM_object(
  BAMM_object,
  tips_to_keep = NULL,
  tips_to_prune = NULL,
  MRCA_node = NULL,
  recompute_shift_configurations = FALSE,
  MAP_odds_ratio_threshold = 5,
  verbose = FALSE
)
```

Arguments

BAMM_object	Object of class "bammdata", typically generated with prepare_diversification_data() or build_BAMM_object(), that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples.
tips_to_keep	Character vector. Tips to retain in the pruned BAMM_object, given as tip labels as found in BAMM_object$tip.label. Default = NULL.
tips_to_prune	Character vector. Tips to remove from the BAMM_object, given as tip labels as found in BAMM_object$tip.label. Default = NULL.
MRCA_node	Integer. ID of a single internal node of BAMM_object, as found in BAMM_object$edge, whose descendant branches/tips must be retained. Use it to focus on the diversification dynamics of one subclade, as in BAMMtools::subtreeBAMM(). Default = NULL. Exactly one of tips_to_keep, tips_to_prune, and MRCA_node must be provided.
recompute_shift_configurations	Logical. Whether the MAP and MSC configurations must be detected again from the pruned posterior samples. • If FALSE (default), $MAP_indices and $MSC_indices are left unchanged (pruning removes branches, not posterior samples, so those indices remain valid), and $MAP_BAMM_object and $MSC_BAMM_object are simply pruned along with the main object. Use this to keep the pruned object comparable with the analysis run on the full phylogeny. • If TRUE, the MAP and MSC configurations are detected again from the pruned posterior samples, as in build_BAMM_object(). Shifts located on removed branches no longer contribute, so the configurations retained as MAP/MSC may differ from those of the full phylogeny.
MAP_odds_ratio_threshold	Numeric. Controls the definition of 'core-shifts' used to distinguish across configurations when fetching the MAP samples. Shifts that have an odds ratio of marginal posterior probability / prior lower than MAP_odds_ratio_threshold are ignored. See BAMMtools::getBestShiftConfiguration(). Only used when recompute_shift_configurations = TRUE. Default = 5.
verbose	Logical. Whether to display progress in the console. Default = FALSE.

Details

Prune a BAMM_object down to a subset of tips, and update all internal elements so that the pruned object still describes the BAMM diversification dynamics, including all deepSTRAPP elements, but only for the retained tips and branches.

When the retained tips do not span the original root, the root of the pruned phylogeny becomes the Most Recent Common Ancestor (MRCA) of the retained tips. The macroevolutionary regime that was governing the branch subtending that MRCA becomes the new background/root regime, and its rate parameters are re-anchored on the new root age (the rates occurring at any given time along the retained branches are unchanged, only the time of reference at which the initial rates are recorded is shifted). The time shift from the initial root age to the new MRCA age is recorded in $pruning_root_shift.

This function also preserves and updates the additional deepSTRAPP elements:

- the Marginal Shift Probability (MSP) = the probability of a regime shift to occur along each branch.
- the Maximum A Posteriori probability (MAP) configurations among the posterior samples = the configurations of regime shifts that were sampled most frequently (See BAMMtools::getBestShiftConfiguration
- the Maximum Shift Credibility (MSC) configurations among the posterior samples = the configurations of regime shift location with the highest product of marginal probabilities across branches (See BAMMtools::maximumShiftCredibility()).

Those additional elements are used by plot_BAMM_rates() to display regime shift probabilities and locations.

Value

The function returns a BAMM_object of class "bammdata" which is a list with at least 26 elements.

Phylogeny-related elements used to plot a phylogeny with ape::plot.phylo():

- $edge Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- $Nnode Integer. Number of internal nodes.
- $tip.label Character vector. Labels of all retained tips.
- $edge.length Numeric vector. Length of edges/branches. Branches resulting from the merging of several initial branches have a length equal to the sum of the lengths of the initial branches.
- $node.label Character vector. Labels of all internal nodes. (Present only if present in the initial BAMM_object)

BAMM internal elements used for tree exploration updated for the new pruned tree:

- $begin Numeric vector. Absolute time since root of edge/branch start (rootward).
- $end Numeric vector. Absolute time since root of edge/branch end (tipward).
- $downseq Integer vector. Order of node visits when using a pre-order tree traversal.
- $lastvisit ID of the last node visited when starting from the node in the corresponding position in $downseq.

BAMM elements summarizing diversification data:

- $numberEvents Integer vector. Number of events/macroevolutionary regimes (k+1) retained in each posterior configuration. k = number of shifts.
- $eventData List of data.frames. One per posterior sample. Records shift events and macroevolutionary regime parameters. 1st line = Background root regime.
- $eventVectors List of integer vectors. One per posterior sample. Record regime ID per branch.
- $tipStates List of integer vectors. One per posterior sample. Record regime ID per tip. Tip vectors are named after the tips only when they were named in the initial BAMM_object: the naming convention of the input is preserved.
- $tipLambda List of numeric vectors. One per posterior sample. Record speciation rates per tip.
- $tipMu List of numeric vectors. One per posterior sample. Record extinction rates per tip.
- $eventBranchSegs List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- $meanTipLambda Numeric vector. Mean tip speciation rates across all posterior configurations of tips.
- $meanTipMu Numeric vector. Mean tip extinction rates across all posterior configurations of tips.
- $type Character string. Set the type of data modeled with BAMM. Should be "diversification".

Additional elements providing key information for downstream analyses:

- $expectedNumberOfShifts Integer. The expected number of regime shifts used to set the prior in BAMM.
- $MSP_tree Object of class phylo. List of 4 elements duplicating information from the Phylogeny-related elements above, except MSP_treeedge.length is recording the Marginal Shift Probability of each branch, recomputed on the pruned phylogeny.
- $MAP_indices Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- $MAP_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configurations, pruned to the retained tips.
- $MSC_indices Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.
- $MSC_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations, pruned to the retained tips.

Elements tracking the pruning, that can be used to relate the pruned object to the initial one:

- $pruned_tip_labels Character vector. Labels of the tips that were removed.
- $pruning_root_shift Numeric. Time elapsed between the root of the initial phylogeny and the root of the pruned phylogeny. Equals 0 when the retained tips span the initial root. Since pruning does not change the distance of any retained node to the present, this is also the difference between the initial and the pruned root ages.
- $pruning_nodes_ID_df Data.frame with four columns providing the conversion table for node IDs: $new_node_ID, $initial_node_ID, $node_type ("tip", "root", or "internal"), and $node_label.

- $pruning_edges_ID_df Data.frame with four columns providing the conversion table for edge IDs: $new_edge_ID, $initial_edge_ID, $position_in_path (1 = the most rootward initial edge merged into the new edge), and $nb_merged_edges. A new edge resulting from the merging of several initial edges holds several rows.

Note on what is *not* recomputed

This function does **not** re-estimate diversification rates. Removing tips changes the incomplete taxon sampling of the phylogeny, but the sampling fractions used during the original BAMM run are not updated, and rates are not re-inferred. If you need diversification rates estimated for a specific clade or set of taxa, you should run a dedicated BAMM analysis on that subset with appropriate sampling fractions, see prepare_diversification_data().

Note on the Marginal Shift Probability tree

The $MSP_tree is always recomputed from the pruned posterior samples, independently of recompute_shift_configurati Marginal shift probabilities are a deterministic function of the posterior samples and of the topology, and are not a choice of configuration. When several branches are merged into a single one, the marginal shift probability of the merged branch is the proportion of posterior samples in which at least one shift occurred anywhere along that merged branch.

Author(s)

Maël Doré

References

For BAMM: Rabosky, D. L. (2014). Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. PloS one, 9(2), e89543. doi:10.1371/journal.pone.0089543. Website: http://bamm-project.org/.

For {BAMMtools}: Rabosky, D. L., Grundler, M., Anderson, C., Title, P., Shi, J. J., Brown, J. W., ... & Larson, J. G. (2014). BAMM tools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. Methods in Ecology and Evolution, 5(7), 701-707. doi:10.1111/2041210X.12199

See Also

Related function in BAMMtools: BAMMtools::subtreeBAMM()

Associated functions in deepSTRAPP: prepare_diversification_data() build_BAMM_object() subset_BAMM_object() plot_BAMM_rates()

For a guided tutorial, see this vignette: vignette("import_external_analyses", package = "deepSTRAPP")

Examples

```
# Load BAMM output
data(whale_BAMM_object, package = "deepSTRAPP")

# Check structure of the initial BAMM_object
```

```
str(whale_BAMM_object, 1)
# Check the initial number of tips
length(whale_BAMM_object$tip.label)
# We have initially 87 tips in the phylogeny

# ----- Example 1: Prune based on set of tips to remove ----- #

## Prune an arbitrary set of tips
# Typically, the tips for which no trait or range data is available
set.seed(seed = 1234)
tips_without_data <- sample(whale_BAMM_object$tip.label, size = 30)

whale_BAMM_object_pruned <- prune_BAMM_object(
   BAMM_object = whale_BAMM_object,
   tips_to_prune = tips_without_data,
   verbose = TRUE)

# Check structure of the pruned BAMM_object
str(whale_BAMM_object_pruned, 1)
# Check the updated number of tips
length(whale_BAMM_object_pruned$tip.label)
# We have now 87 - 30 = 57 tips in the pruned phylogeny

# The tips that were removed are recorded in the pruned BAMM_object
head(whale_BAMM_object_pruned$pruned_tip_labels)

# Branches leading to removed tips are dropped, and the remaining branches are merged
# The conversion table records which initial branches were merged into each new branch
head(whale_BAMM_object_pruned$pruning_edges_ID_df)
table(whale_BAMM_object_pruned$pruning_edges_ID_df$nb_merged_edges)

# Diversification rates estimated at the retained tips are left untouched by the pruning
retained_tips <- match(whale_BAMM_object_pruned$tip.label, whale_BAMM_object$tip.label)
all.equal(whale_BAMM_object_pruned$meanTipLambda,
          whale_BAMM_object$meanTipLambda[retained_tips])

## Plot mean rates and MAP regime shifts on the initial vs. pruned phylogeny
old_par <- par()$mfrow
par(mfrow = c(1, 2))

plot_BAMM_rates(whale_BAMM_object, regimes_size = 3)
plot_BAMM_rates(whale_BAMM_object_pruned, regimes_size = 3)

par(mfrow = old_par)

# ----- Example 2: Prune based on set of tips to keep ----- #

tips_with_data <- setdiff(whale_BAMM_object$tip.label, tips_without_data)

whale_BAMM_object_kept <- prune_BAMM_object(
   BAMM_object = whale_BAMM_object,
   tips_to_keep = tips_with_data)
```

```
# Both ways of selecting the tips give the same pruned BAMM_object
all.equal(whale_BAMM_object_pruned, whale_BAMM_object_kept)

# ----- Example 3: Prune to retain a single subclade ----- #

# Plot the initial phylogeny to pick the MRCA node of the focal subclade
plot(ape::as.phylo(whale_BAMM_object), cex = 0.5)
ape::nodelabels()

# Subset BAMM object to focus on node 103 = Odontoceti Infra-order ("toothed whales")
whale_BAMM_object_subclade <- prune_BAMM_object(
   BAMM_object = whale_BAMM_object,
  MRCA_node = 103)

# Check the number of tips retained in the subclade
length(whale_BAMM_object_subclade$tip.label)
# Only 72 odontocete species remain

# The root of the pruned phylogeny is now the MRCA of the subclade,
# so the phylogeny is shallower than the initial one
whale_BAMM_object_subclade$pruning_root_shift # 2 My shift in root_age
max(whale_BAMM_object$end) ; max(whale_BAMM_object_subclade$end)
# Cetacae phylogeny is 35.4 My old; Odontoceti is phylogeny is 33.4 My old

## Plot mean rates and MAP regime shifts on the pruned phylogeny
plot_BAMM_rates(whale_BAMM_object_subclade,
                add_regime_shifts = TRUE,
                configuration_type = "MAP",
                regimes_size = 3)

# Since we subsetted diversification dynamics only for the Odontoceti subclade,
# we may wish to identify the MAP/MSC configurations based only on the events
# occurring along the remaining branches, and not across the full initial tree.

# For this, we can set 'recompute_shift_configurations = TRUE'.

 # This may take several seconds to run
## Detect the MAP/MSC configurations again, on the pruned phylogeny
identical(whale_BAMM_object_pruned$MAP_indices, whale_BAMM_object$MAP_indices)

# Set 'recompute_shift_configurations = TRUE' to identify the configurations
# that are the most supported once the removed branches no longer contribute
whale_BAMM_object_recomputed <- prune_BAMM_object(
   BAMM_object = whale_BAMM_object,
   MRCA_node = 103,
   recompute_shift_configurations = TRUE,
   verbose = TRUE)

# Compare the number of posterior samples supporting the MAP configuration
length(whale_BAMM_object$MAP_indices)
length(whale_BAMM_object_recomputed$MAP_indices)

## Plot mean rates and updated MAP regime shifts on the pruned phylogeny
```

```
plot_BAMM_rates(whale_BAMM_object_recomputed,
                add_regime_shifts = TRUE,
                configuration_type = "MAP",
                regimes_size = 3)
```

`run_deepSTRAPP_for_focal_time` *Run deepSTRAPP to test for a relationship between diversification rates and trait data at a given focal time*

Description

Wrapper function to run deepSTRAPP workflow for a given point in the past (i.e. the `focal_time`). It starts from traits mapped on a phylogeny (trait data) and BAMM output (diversification data) and carries out the appropriate statistical method to test for a relationship between diversification rates and trait data. Tests are based on block-permutations: rates data are randomized across tips following blocks defined by the diversification regimes identified on each tip (typically from a BAMM).

Such tests are called STructured RAte Permutations on Phylogenies (STRAPP) as described in Rabosky, D. L., & Huang, H. (2016). A robust semi-parametric test for detecting trait-dependent diversification. Systematic biology, 65(2), 181-193. doi:10.1093/sysbio/syv066.

See the original `BAMMtools::traitDependentBAMM()` function used to carry out STRAPP test on extant time-calibrated phylogenies.

Tests can be carried out on speciation, extinction and net diversification rates.

Usage

```
run_deepSTRAPP_for_focal_time(
  contMap = NULL,
  contMaps = NULL,
  densityMaps = NULL,
  simmaps = NULL,
  nb_simulations = NULL,
  ace = NULL,
  tip_data = NULL,
  trait_data_type,
  keep_tip_labels = TRUE,
  BAMM_object,
  rate_type = "net_diversification",
  focal_time,
  uncertainty_strategy = "paired",
  trait_maps_vs_BAMM_samples_list = NULL,
  seed = NULL,
  nb_permutations = NULL,
```

```
  alpha = 0.05,
  two_tailed = TRUE,
  one_tailed_hypothesis = NULL,
  posthoc_pairwise_tests = FALSE,
  p.adjust_method = "none",
  trait_data_type_for_stats = NULL,
  return_perm_data = FALSE,
  nthreads = 1,
  print_hypothesis = TRUE,
  extract_trait_data_melted_df = FALSE,
  extract_diversification_data_melted_df = FALSE,
  return_updated_Maps = FALSE,
  return_updated_BAMM_object = FALSE,
  verbose = TRUE
)
```

Arguments

contMap	For continuous trait data. Object of class "contMap", typically generated with prepare_trait_data() or phytools::contMap(), that contains a phylogenetic tree and associated continuous trait mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
contMaps	For continuous trait data. List of objects of class "contMap", typically generated with prepare_trait_data(), that contains multiple continuous stochastic maps that represent possible trait evolutionary histories conditioned on the observed trait values and model fit.
densityMaps	For categorical trait or biogeographic data. List of objects of class "densityMap", typically generated with prepare_trait_data() or phytools::densityMap(), that contains a phylogenetic tree and associated posterior probability of being in a given state/range along branches. Each object (i.e., densityMap) corresponds to a state/range. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
simmaps	For categorical trait or biogeographic data. List of objects of class "simmap", typically generated with prepare_trait_data() or phytools::make.simmap(), that represent discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches. This is needed to be able to track which simulated history provided which trait data in downstream analyses employing the "paired" or "full" strategies to account for uncertainty in ancestral trait estimates.
nb_simulations	For categorical trait or biogeographic data. Integer. The number of stochastic maps used to simulate trait evolution. This is needed for the "paired" and "full" strategies to account for trait estimate uncertainty, if only densityMaps summarizing posterior state/range density are provided, but not the simmaps representing all evolutionary histories.
ace	(Optional) Ancestral Character Estimates (ACE) at the internal nodes. Obtained with prepare_trait_data() as output in the $ace slot.

- For continuous trait data: Named numeric vector typically generated with `phytools::fastAnc()`, `phytools::anc.ML()`, or `ape::ace()`. Names are nodes_ID of the internal nodes. Values are ACE of the trait.
- For categorical trait or biogeographic data: Matrix that records the posterior probabilities of ancestral states/ranges. Rows are internal nodes_ID. Columns are states/ranges. Values are posterior probabilities of each state per node. Needed in all cases to provide accurate estimates of trait values.

`tip_data` (Optional) Named vector of tip values of the trait.

- For continuous trait data: Named numeric vector of trait values.
- For categorical trait or biogeographic data: Named character vector of states/ranges. Names must match `tip.label` in the phylogeny. Needed to provide accurate tip values.
- For biogeographic data, ranges should follow the coding scheme of BioGeoBEARS with a unique CAPITAL letter per unique area (ex: A, B), combined to form multi-area ranges (Ex: AB). Alternatively, you can provide tip_data as a matrix or data.frame of binary presence/absence in each area (coded as unique CAPITAL letter). In this case, columns are unique areas, rows are taxa, and values are integer (0/1) signaling absence or presence of the taxa in the area.

`trait_data_type`
Character string. Specify the type of trait data. Must be one of "continuous", "categorical", "biogeographic".

`keep_tip_labels`
Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label` on the updated phylogeny. Default is `TRUE`.

`BAMM_object` Object of class `"bammdata"`, typically generated with `prepare_diversification_data()`, that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples. The phylogenetic tree must be the same as the one associated with the `contMap`/`densityMaps`, `ace` and `tip_data`.

`rate_type` A character string specifying the type of diversification rates to use. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).

`focal_time` Numeric. The time, in terms of time distance from the present, at which data must be extracted and the phylogeny and mappings must be cut. It must be smaller than the root age of the phylogeny.

`uncertainty_strategy`
Character string. To select the strategy used to account for uncertainty in estimates.

- `"rates_only"`: Only accounts for diversification-rate uncertainty across BAMM posterior samples. Uses ML estimates for continuous traits and the most frequent state/range observed across stochastic maps for categorical and biogeographic data.
- `"paired"`: Default option. Accounts for both diversification-rate and ancestral trait/range reconstruction uncertainty by pairing BAMM posterior samples with stochastic maps. When the number of BAMM samples and stochastic maps differ, random pairing with replacement from the smaller set is used so that all posterior samples and stochastic maps contribute to the analysis.

- "full": Exhaustive option that accounts for trait/range- and rate- uncertainty by crossing all BAMM posterior samples with all stochastic maps. Accounts for both diversification-rate and ancestral reconstruction uncertainty by evaluating every combination of BAMM posterior samples and stochastic maps. WARNING: This exhaustive approach can substantially increase computation time and memory requirements and is therefore recommended only for moderate-sized analyses.

trait_maps_vs_BAMM_samples_list
: (Optional) List of two elements manually providing the names to associate stochastic maps ($trait_map_ID) with BAMM samples ($BAMM_posterior_sample_ID). This is typically used to ensure the same stochastic maps and BAMM samples are used to test across multiple time-steps. Values are the names of the objects such as "Map_X" and "BAMM_X". Default = NULL. * For uncertainty_strategy == 'rates_only', the $trait_map_ID must be "Map_ML" as only the ML estimates of trait values/states/ranges are used. * For uncertainty_strategy == 'paired', each pair of stochastic map and BAMM sample will be used once. * For uncertainty_strategy == 'full', all stochastic maps will be matched with all BAMM samples. Those may partly differ from the actual maps and BAMM samples used for the test as recorded in $STRAPP_results$perm_data_df because invalid maps with not enough states/ranges are discarded.

seed
: Integer. Set the seed to ensure reproducibility. Default is NULL (a random seed is used).

nb_permutations
: Integer. To select the number of random permutations to perform during the tests. If NULL (default), all BAMM posterior samples will be used once.

alpha
: Numeric. Significance level to use to compute the estimate corresponding to the values of the test statistic used to assess significance of the test. This does NOT affect p-values. Default is 0.05.

two_tailed
: Logical. To define the type of tests. If TRUE (default), tests for correlations/differences in rates will be carried out with a null hypothesis that rates are not correlated with trait values (continuous data) or equal between trait states (categorical and biogeographic data). If FALSE, one-tailed tests are carried out.

  - For continuous data, it involves defining a one_tailed_hypothesis testing for either a "positive" or "negative" correlation under the alternative hypothesis.
  - For binary data (two states), it involves defining a one_tailed_hypothesis indicating which states have higher rates under the alternative hypothesis.
  - For multinomial data (more than two states), it defines the type of post hoc pairwise tests to carry out between pairs of states. If posthoc_pairwise_tests = TRUE, all two-tailed (if two_tailed = TRUE) or one-tailed (if two_tailed = FALSE) tests are automatically carried out.

one_tailed_hypothesis
: A character string specifying the alternative hypothesis in the one-tailed test. For continuous data, it is either "negative" or "positive" correlation. For binary data, it lists the trait states with states ordered in increasing rates under the alternative hypothesis, separated by a greater-than such as c('A > B').

posthoc_pairwise_tests
: Logical. Only for multinomial data (with more than two states). If TRUE, all possible post hoc pairwise (Dunn) tests will be computed across all pairs of states. This is a way to detect which pairs of states have significant differences in rates if the overall test (Kruskal-Wallis) is significant. Default is FALSE.

p.adjust_method
: A character string. Only for multinomial data (with more than two states). It specifies the type of correction to apply to the p-values in the post hoc pairwise tests to account for multiple comparisons. See stats::p.adjust() for the available methods. Default is none.

trait_data_type_for_stats
: (Optional) Character string. One of "continuous", "binary", or "multinomial". Imposes the statistical method to use, instead of deriving it from the type of traits and the number of states/ranges observed at this focal_time. This is set automatically by run_deepSTRAPP_over_time(), which fixes the method once for the whole run from the complete trait mapping, so that the same test is used at every time step and the p-values along the trajectory remain comparable. Default is NULL, in which case the method is selected from the number of states/ranges in trait_data_list$trait_data observed at this focal_time.

return_perm_data
: Logical. Whether to return the stats data computed from the posterior samples for observed and permuted data in the output. This is needed to plot the histogram of the null distribution used to assess significance of the test with plot_histogram_STRAPP_test_for_focal_time(). Default is FALSE.

nthreads
: Integer. Number of threads to use for parallel computing of the STRAPP tests across the permutations. The R package parallel must be loaded for nthreads > 1. Default is 1.

print_hypothesis
: Logical. Whether to print information on what test is carried out, detailing the null and alternative hypotheses, what significance level is used to reject or not the null hypothesis, and how uncertainty in trait estimates is handled. Default is TRUE.

extract_trait_data_melted_df
: Logical. Specify whether trait data (values/states/ranges) must be extracted from the mapped phylogenies and returned in a melted data.frame. Default is FALSE.

extract_diversification_data_melted_df
: Logical. Specify whether diversification data (regimes ID and tip rates) must be extracted from the updated_BAMM_object and returned in a melted data.frame. Default is FALSE.

return_updated_Maps
: Logical. Specify whether the updated version of the mapped phylogenies (contMap(s)/densityMaps/sim should be returned among the outputs. The update of the contMap(s)/densityMaps/simmaps consists of cutting off branches and mapping that are younger than the focal_time. Default is FALSE.

return_updated_BAMM_object
: Logical. Specify whether the updated_BAMM_object with phylogeny and mapped diversification rates cut-off at the focal_time should be returned among the outputs.

verbose — Logical. Should progression be displayed? A message will be printed at each step of the deepSTRAPP workflow, and for every batch of 100 BAMM posterior samples whose rates and regimes are updated, and optionally extracted in a melted data.frame (if extract_diversification_data_melted_df = TRUE). Default is TRUE.

Details

The function encapsulates several functions carrying out each step of the deepSTRAPP workflow:

Extract trait data:

For uncertainty_strategy = "rates_only", extract_most_likely_trait_values_for_focal_time() is used to extract the most likely trait/range values found along branches at the focal_time.

- For continuous trait, it is the ML estimates mapped in contMap

For uncertainty_strategy = "paired" or "full", extract_all_trait_values_for_focal_time() is used to extract all trait/range states/values found along branches at the focal_time across all stochastic maps.

Extracted trait values are stored in STRAPP_results$trait_data_list in a list including the extracted $trait_data, $focal_time, $trait_data_type, and $uncertainty_strategy, and used as input to compute STRAPP tests with compute_STRAPP_test_for_focal_time().

Optionally, if return_updated_Maps = TRUE, these functions can update the mapped phylogenies (contMap(s)/densityMaps/simmaps) provided as inputs such as branches overlapping the focal_time are shortened to the focal_time, and the trait mapping for the cut off branches are removed by updating the $tree$maps and $tree$mapped.edge elements.

Extract trait data in a melted df:

If requested (extract_trait_data_melted_df = TRUE), extract_trait_data_melted_df_for_focal_time() is used to format the extracted trait data (values/states/ranges) into a melted data.frame summarizing trait data as found on the phylogeny for the focal_time.

Extract diversification data:

update_rates_and_regimes_for_focal_time() updates the BAMM_object to obtain the diversification rates/regimes found along branches at the focal_time across all BAMM posterior samples. Optionally, the function can update the BAMM_object to display a mapped phylogeny such as branches overlapping the focal_time are shortened to the focal_time

Extract diversification data in a melted df:

If requested (extract_diversification_data_melted_df = TRUE), extract_diversification_data_melted_df_for will be used to extract regimes ID and tip rates from the updated_BAMM_object and provide a melted data.frame summarizing the diversification data as found on the phylogeny for the focal_time.

Compute STRAPP test:

compute_STRAPP_test_for_focal_time() carries out the appropriate statistical method to test for a relationship between diversification rates and trait data for a given point in the past (i.e. the focal_time). It can handle three types of statistical tests depending on the type of trait data provided:

- Continuous trait data: Test for correlations with the Spearman's rank correlation test (See stats::cor.test).
- Binary trait data (two states): Test for differences in rates between states with the Mann-Whitney-Wilcoxon rank-sum test (See stats::wilcox.test).
- Multinominal trait data (More than two states): Test for differences in rates across all states with the Kruskal-Wallis H test (See stats::kruskal.test). If `posthoc_pairwise_tests = TRUE`, Dunn's post hoc pairwise rank-sum tests between pairs of states will be carried out too (See dunn.test::dunn.test).

Value

The function returns a list with at least seven elements.

- `$STRAPP_results` List with at least eight elements summarizing the results of the STRAPP tests. See `compute_STRAPP_test_for_focal_time()` for a detailed description of the output.
- `$focal_time` Integer. The time, in terms of time distance from the present, at which the data were extracted and the STRAPP test carried out.
- `$trait_data_type` Character string. Specify the type of trait data. Possible values are: "continuous", "categorical", "biogeographic".
- `$trait_data_type_for_stats` Character string. The type of trait data used to select statistical method. One of 'continuous', 'binary', or 'multinomial'.
- `$rate_type` Character string. The type of diversification rates used in the tests: 'speciation', 'extinction' or 'net_diversification'.
- `$uncertainty_strategy` Character string. The strategy used to account for uncertainty in estimates. One of 'rates_only', 'paired', or 'full'.
- `$trait_maps_vs_BAMM_samples_list` List of two elements recording the stochastic maps (`$trait_map_ID`) and BAMM samples (`$BAMM_posterior_sample_ID`) chosen for testing across time-steps. Those may partly differ from the actual maps and BAMM samples used for the tests as recorded in `$STRAPP_results$perm_data_df` because invalid maps with not enough states/ranges are discarded.

Optional formatted output:

- `$trait_data_df` A data.frame with five columns summarizing the trait data as found on the mapped phylogenies for the `focal_time`. See `extract_trait_data_melted_df_for_focal_time()` for a detailed description of the output.
- `$diversification_data_df` A data.frame with six columns summarizing the diversification data as found on the phylogeny for the `focal_time`. See `extract_diversification_data_melted_df_for_focal_t` for a detailed description of the output.

Optional data updated for the `focal_time`:

- `$updated_Maps` A list that contains the updated `contMap(s)`/`densityMaps`/`simmaps` provided as inputs, with branches and mapping that are younger than the `focal_time` cut off. The updated mapped phylogenies can be visualized with `plot_contMap()` or `plot_densityMaps_overlay()`.
- `$updated_BAMM_object` An updated `BAMM_object` of class `"bammdata"` that contains rates and regimes ID found at the `focal_time`. Can be used as input of `plot_BAMM_rates()` to display a phylogeny mapped with diversification rates with branches cut at the `focal_time`. See `update_rates_and_regimes_for_focal_time()` for a detailed description of the output.

Author(s)

Maël Doré

See Also

extract_most_likely_trait_values_for_focal_time() extract_all_trait_values_for_focal_time() extract_trait_data_melted_df_for_focal_time() update_rates_and_regimes_for_focal_time() extract_diversification_data_melted_df_for_focal_time() compute_STRAPP_test_for_focal_time()

For a guided tutorial on complete deepSTRAPP workflow, see the associated vignettes:

- For continuous trait data: `vignette("deepSTRAPP_continuous_data", package = "deepSTRAPP")`
- For categorical trait data: `vignette("deepSTRAPP_categorical_3lvl_data", package = "deepSTRAPP")`
- For biogeographic range data: `vignette("deepSTRAPP_biogeographic_data", package = "deepSTRAPP")`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 # ----- Example 1: Continuous trait ----- #
 ## Load data

 # Load trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
 Ponerinae_tree_old_calib$node.label <- NULL

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare trait data

 # Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

 # Select a color scheme from lowest to highest values
 color_scale = c("darkgreen", "limegreen", "orange", "red")

  # (May take several minutes to run)

 # Map trait evolution across multiple simulations (i.e., continuous stochastic maps)
 Ponerinae_cont_data_old_calib <- prepare_trait_data(
    tip_data = Ponerinae_cont_tip_data,
    trait_data_type = "continuous",
    phylo = Ponerinae_tree_old_calib,
```

```
   seed = 1234,
   evolutionary_models = "BM",
   plot_map = FALSE,
   run_stochastic_maps = TRUE,
   nb_simulations = 100, # Run 100 simulations of trait evolution
   verbose = TRUE)

## Load directly trait data output
Ponerinae_cont_data_old_calib <- readRDS(system.file("extdata",
   "Ponerinae_cont_data_old_calib.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

# Plot contMap = ML estimates of continuous trait evolution
plot_contMap(contMap = Ponerinae_cont_data_old_calib$contMap,
             color_scale = color_scale)

## Set focal time to 10 Mya
focal_time <- 10

## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.
deepSTRAPP_output <- run_deepSTRAPP_for_focal_time(
   # Include contMap to plot ML estimates
   contMap = Ponerinae_cont_data_old_calib$contMap,
   # Include contMaps to extract trait estimates across all simulations
   contMaps = Ponerinae_cont_data_old_calib$contMaps,
   ace = Ponerinae_cont_data_old_calib$ace,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   rate_type = "net_diversification",
   focal_time = focal_time,
   uncertainty_strategy = "paired",
   seed = 1234,
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_object = TRUE)

## Explore output
str(deepSTRAPP_output, max.level = 2)

# Access deepSTRAPP results
str(deepSTRAPP_output$STRAPP_results)

# Access trait data
head(deepSTRAPP_output$trait_data_df)
# Trait data includes 100 simulations (i.e., stochastic maps)
table(deepSTRAPP_output$trait_data_df$Map_ID)

# Access the diversification data in a melted data.frame
```

```
head(deepSTRAPP_output$diversification_data_df)
# Diversification data includes 1000 BAMM posteriors
table(deepSTRAPP_output$diversification_data_df$BAMM_sample_ID)

# Plot rates vs. trait values across branches for
# all pairs of stochastic maps and BAMM posterior pairs
plot_rates_vs_trait_data_for_focal_time(deepSTRAPP_output)

# Plot updated contMap
plot_contMap(deepSTRAPP_output$updated_Maps$contMap)
ape::nodelabels(text =
  deepSTRAPP_output$updated_Maps$contMap$tree$initial_nodes_ID)

# Plot diversification rates on updated phylogeny
plot_BAMM_rates(deepSTRAPP_output$updated_BAMM_object, labels = TRUE)

# Plot histogram of test stats
plot_histogram_STRAPP_test_for_focal_time(
  deepSTRAPP_outputs = deepSTRAPP_output)


# ----- Example 2: Categorical trait ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare trait data

# Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
table(Ponerinae_cat_3lvl_tip_data)

# Select color scheme for states
colors_per_states <- c("forestgreen", "sienna", "goldenrod")
names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

 # (May take several minutes to run)
## Produce densityMaps using stochastic character mapping based on an equal-rates (ER) Mk model
Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
   tip_data = Ponerinae_cat_3lvl_tip_data,
   phylo = Ponerinae_tree_old_calib,
   trait_data_type = "categorical",
```

```
   colors_per_levels = colors_per_states,
   evolutionary_models = "ER", # Use default ER model
   nb_simulations = 100, # Reduce number of simulations to save time
   seed = 1234, # Seet seed for reproducibility
   return_best_model_fit = TRUE,
   return_model_selection_df = TRUE,
   plot_map = FALSE)

# Load directly output
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")

## Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

deepSTRAPP_output <- run_deepSTRAPP_for_focal_time(
    densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
   # Inform the nb of simulations to reconstruct state distribution across stochastic maps
    nb_simulations = 100,
    ace = Ponerinae_cat_3lvl_data_old_calib$ace,
    tip_data = Ponerinae_cat_3lvl_tip_data,
    trait_data_type = "categorical",
    rate_type = "net_diversification",
    BAMM_object = Ponerinae_BAMM_object_old_calib,
    focal_time = focal_time,
    uncertainty_strategy = "paired",
    posthoc_pairwise_tests = TRUE,
    return_perm_data = TRUE,
    extract_trait_data_melted_df = TRUE,
    extract_diversification_data_melted_df = TRUE,
    return_updated_Maps = TRUE,
    return_updated_BAMM_object = TRUE)

## Explore output
str(deepSTRAPP_output, max.level = 1)

# Access deepSTRAPP results
str(deepSTRAPP_output$STRAPP_results, max.level = 2)
# Result for overall Kruskal-Wallis test
deepSTRAPP_output$STRAPP_results[1:3]
# Results for posthoc pairwise Dunn's tests
deepSTRAPP_output$STRAPP_results$posthoc_pairwise_tests$summary_df

# Access trait data in a melted data.frame
# Because trait data was provided as densityMaps and not simmaps,
# the stochastic maps are dummy maps generated to reproduce
# the frequency of states as recorded in the densityMaps.
head(deepSTRAPP_output$trait_data_df)
table(deepSTRAPP_output$trait_data_df$Map_ID)

# Access the diversification data in a melted data.frame
```

```
head(deepSTRAPP_output$diversification_data_df)
# Diversification data includes 1000 BAMM posteriors
table(deepSTRAPP_output$diversification_data_df$BAMM_sample_ID)

# Plot rates vs. states across branches
plot_rates_vs_trait_data_for_focal_time(
    deepSTRAPP_outputs = deepSTRAPP_output,
    colors_per_levels = colors_per_states)

# Plot updated densityMaps cut at focal time
plot_densityMaps_overlay(deepSTRAPP_output$updated_Maps$densityMaps)

# Plot diversification rates on updated phylogeny
plot_BAMM_rates(BAMM_object = deepSTRAPP_output$updated_BAMM_object, legend = TRUE, labels = FALSE,
  colorbreaks = deepSTRAPP_output$updated_BAMM_object$initial_colorbreaks$net_diversification)

# Plot histogram of Kruskal-Wallis overall test stats
plot_histogram_STRAPP_test_for_focal_time(
   deepSTRAPP_outputs = deepSTRAPP_output)

# Plot histograms of posthoc pairwise Dunn's test stats
plot_histogram_STRAPP_test_for_focal_time(
   deepSTRAPP_outputs = deepSTRAPP_output,
   plot_posthoc_tests = TRUE)

# ----- Example 3: Biogeographic ranges ----- #

## Load data

# Load phylogeny
data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
# Load trait df
data(Ponerinae_binary_range_table, package = "deepSTRAPP")

# Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Prepare range data for Old World vs. New World

# No overlap in ranges
table(Ponerinae_binary_range_table$Old_World, Ponerinae_binary_range_table$New_World)

Ponerinae_NO_data <- stats::setNames(object = Ponerinae_binary_range_table$Old_World,
                                     nm = Ponerinae_binary_range_table$Taxa)
Ponerinae_NO_data <- as.character(Ponerinae_NO_data)
Ponerinae_NO_data[Ponerinae_NO_data == "TRUE"] <- "O" # O = Old World
Ponerinae_NO_data[Ponerinae_NO_data == "FALSE"] <- "N" # N = New World
names(Ponerinae_NO_data) <- Ponerinae_binary_range_table$Taxa
table(Ponerinae_NO_data)
```

```
 colors_per_ranges <- c("mediumpurple2", "peachpuff2")
 names(colors_per_ranges) <- c("N", "O")

  # (May take several minutes to run)

 # Load ape and BioGeoBEARS to use models internally
 library(ape)
 library(BioGeoBEARS)

 ## Run evolutionary models
Ponerinae_biogeo_data <- prepare_trait_data(
    tip_data = Ponerinae_NO_data,
    trait_data_type = "biogeographic",
    phylo = Ponerinae_tree_old_calib,
    evolutionary_models = "DEC+J", # Default = "DEC" for biogeographic
    BioGeoBEARS_directory_path = tempdir(), # Ex: "./BioGeoBEARS_directory/"
    keep_BioGeoBEARS_files = FALSE,
    prefix_for_files = "Ponerinae_old_calib",
    max_range_size = 2,
    split_multi_area_ranges = TRUE, # Set to TRUE to display the two outputs
    nb_simulations = 100, # Reduce to save time (Default = '1000')
    colors_per_levels = colors_per_ranges,
    return_model_selection_df = TRUE,
    verbose = TRUE)

# Load directly output
data(Ponerinae_biogeo_data_old_calib, package = "deepSTRAPP")

## Explore output
str(Ponerinae_biogeo_data_old_calib, 1)

## Set focal time to 10 Mya
focal_time <- 10

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for focal time = 10 Mya.

deepSTRAPP_output <- run_deepSTRAPP_for_focal_time(
   densityMaps = Ponerinae_biogeo_data_old_calib$densityMaps,
   # Inform the nb of simulations to reconstruct state distribution across stochastic maps
   nb_simulations = 100,
   ace = Ponerinae_biogeo_data_old_calib$ace,
   tip_data = Ponerinae_NO_data,
   trait_data_type = "biogeographic",
   rate_type = "net_diversification",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   focal_time = focal_time,
   uncertainty_strategy = "paired",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_object = TRUE)
```

```
## Explore output
str(deepSTRAPP_output, max.level = 1)

# Access deepSTRAPP results
str(deepSTRAPP_output$STRAPP_results, max.level = 2)
# Result for Mann-Whitney-Wilcoxon test
deepSTRAPP_output$STRAPP_results[1:3]

# Access trait data
# Because trait data was provided as densityMaps and not simmaps,
# the stochastic maps are dummy maps generated to reproduce
# the frequency of ranges as recorded in the densityMaps.
 head(deepSTRAPP_output$trait_data_df)

# Access the diversification data in a melted data.frame
head(deepSTRAPP_output$diversification_data_df)

# Plot rates vs. ranges across branches
plot_rates_vs_trait_data_for_focal_time(
    deepSTRAPP_outputs = deepSTRAPP_output,
    colors_per_levels = colors_per_ranges)

# Plot updated densityMaps cut at focal time
plot_densityMaps_overlay(deepSTRAPP_output$updated_Maps$densityMaps)

# Plot diversification rates on updated phylogeny
plot_BAMM_rates(BAMM_object = deepSTRAPP_output$updated_BAMM_object, legend = TRUE, labels = FALSE,
  colorbreaks = deepSTRAPP_output$updated_BAMM_object$initial_colorbreaks$net_diversification)

# Plot histogram of Mann-Whitney-Wilcoxon test
plot_histogram_STRAPP_test_for_focal_time(
  deepSTRAPP_outputs = deepSTRAPP_output)
}
```

`run_deepSTRAPP_over_time` *Run deepSTRAPP to test for a relationship between diversification rates and trait data over multiple time steps*

Description

Wrapper function to run deepSTRAPP workflows over multiple time steps in the past. It starts from traits mapped on a phylogeny (trait data) and BAMM output (diversification data) and carries out the appropriate statistical method to test for a relationship between diversification rates and trait data. The workflow is repeated over multiple points in time (i.e. the `time_steps`) and results are summarized in a data.frame. The function can also provide summaries of trait values and diversification rates extracted along branches over the different `time_steps`.

Statistical tests are based on block-permutations: rates data are randomized across tips following blocks defined by the diversification regimes identified on each tip (typically from a BAMM). Such tests are called STructured RAte Permutations on Phylogenies (STRAPP) as described in Rabosky, D. L., & Huang, H. (2016). A robust semi-parametric test for detecting trait-dependent diversification. Systematic biology, 65(2), 181-193. doi:10.1093/sysbio/syv066.

See the original `BAMMtools::traitDependentBAMM()` function used to carry out STRAPP test on extant time-calibrated phylogenies.

Tests can be carried out on speciation, extinction and net diversification rates.

Usage

```
run_deepSTRAPP_over_time(
  contMap = NULL,
  contMaps = NULL,
  densityMaps = NULL,
  simmaps = NULL,
  nb_simulations = NULL,
  ace = NULL,
  tip_data = NULL,
  trait_data_type,
  keep_tip_labels = TRUE,
  BAMM_object,
  rate_type = "net_diversification",
  time_steps = NULL,
  time_range = NULL,
  nb_time_steps = NULL,
  time_step_duration = NULL,
  uncertainty_strategy = "paired",
  trait_maps_vs_BAMM_samples_list = NULL,
  seed = NULL,
  nb_permutations = NULL,
  alpha = 0.05,
  two_tailed = TRUE,
  one_tailed_hypothesis = NULL,
  posthoc_pairwise_tests = FALSE,
  p.adjust_method = "none",
  return_perm_data = FALSE,
  nthreads = 1,
  print_hypothesis = TRUE,
  extract_trait_data_melted_df = FALSE,
  extract_diversification_data_melted_df = FALSE,
  return_STRAPP_results = FALSE,
  return_updated_Maps = FALSE,
  return_updated_BAMM_objects = FALSE,
  verbose = TRUE,
  verbose_extended = FALSE
)
```

Arguments

contMap	For continuous trait data. Object of class "contMap", typically generated with prepare_trait_data() or phytools::contMap(), that contains a phylogenetic tree and associated continuous trait mapping. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
contMaps	For continuous trait data. List of objects of class "contMap", typically generated with prepare_trait_data(), that contains multiple continuous stochastic maps that represent possible trait evolutionary histories conditioned on the observed trait values and model fit.
densityMaps	For categorical trait or biogeographic data. List of objects of class "densityMap", typically generated with prepare_trait_data(), that contains a phylogenetic tree and associated posterior probability of being in a given state/range along branches. Each object (i.e., densityMap) corresponds to a state/range. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).
simmaps	For categorical trait or biogeographic data. List of objects of class "simmap", typically generated with prepare_trait_data() or phytools::make.simmap(), that represent discrete character/geographic evolutionary history (i.e., transitions in character states/geographic ranges) mapped along branches. This is needed to be able to track which simulated history provided which trait data in downstream analyses employing the "paired" or "full" strategies to account for uncertainty in ancestral trait estimates.
nb_simulations	For categorical trait or biogeographic data. Integer. The number of stochastic maps used to simulate trait evolution. This is needed for the "paired" and "full" strategies to account for trait estimate uncertainty, if only densityMaps summarizing posterior state/range density are provided, but not the simmaps representing all evolutionary histories.
ace	(Optional) Ancestral Character Estimates (ACE) at the internal nodes. Obtained with prepare_trait_data() as output in the $ace slot. • For continuous trait data: Named numeric vector typically generated with phytools::fastAnc(), phytools::anc.ML(), or ape::ace(). Names are nodes_ID of the internal nodes. Values are ACE of the trait. • For categorical trait or biogeographic data: Matrix that records the posterior probabilities of ancestral states/ranges. Rows are internal nodes_ID. Columns are states/ranges. Values are posterior probabilities of each state per node. Needed in all cases to provide accurate estimates of trait values.
tip_data	(Optional) Named vector of tip values of the trait. • For continuous trait data: Named numeric vector of trait values. • For categorical trait or biogeographic data: Named character vector of states/ranges. Names must match tip.label in the phylogeny. Needed to provide accurate tip values. • For biogeographic data, ranges should follow the coding scheme of BioGeoBEARS with a unique CAPITAL letter per unique area (ex: A, B), combined to form multi-area ranges (Ex: AB). Alternatively, you can provide tip_data as a matrix or data.frame of binary presence/absence in each

area (coded as unique CAPITAL letter). In this case, columns are unique areas, rows are taxa, and values are integer (0/1) signaling absence or presence of the taxa in the area.

`trait_data_type`
: Character string. Specify the type of trait data. Must be one of "continuous", "categorical", "biogeographic".

`keep_tip_labels`
: Logical. Specify whether terminal branches with a single descendant tip must retain their initial `tip.label` on the updated phylogeny. Default is `TRUE`.

`BAMM_object`
: Object of class `"bammdata"`, typically generated with `prepare_diversification_data()`, that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples. The phylogenetic tree must be the same as the one associated with the `contMap`, `ace` and `tip_data`.

`rate_type`
: A character string specifying the type of diversification rates to use. Must be one of 'speciation', 'extinction' or 'net_diversification' (default).

`time_steps`
: Numeric vector. Time steps at which the STRAPP tests should be carried out. If `NULL` (the default), `time_steps` will be generated from a combination of two arguments among `time_range`, `nb_time_steps`, and/or `time_step_duration`.

`time_range`
: Numeric vector of length 2. Time boundaries within which the `time_steps` must be defined if not provided. If `NULL` (the default), and `time_range` is needed to generate the `time_steps`, the depth of the tree is used by default: `c(0, root_age)`. However, no time step will be generated for the 'root_age'.

`nb_time_steps, time_step_duration`
: Numeric. Number of time steps and duration of each time step used to generate `time_steps` if not provided. You must provide at least one of those two arguments to be able to generate `time_steps`.

`uncertainty_strategy`
: Character string. To select the strategy used to account for uncertainty in estimates.

- `"rates_only"`: Only accounts for diversification-rate uncertainty across BAMM posterior samples. Uses ML estimates for continuous traits and the most frequent state/range observed across stochastic maps for categorical and biogeographic data.
- `"paired"`: Default option. Accounts for both diversification-rate and ancestral trait/range reconstruction uncertainty by pairing BAMM posterior samples with stochastic maps. When the number of BAMM samples and stochastic maps differ, random pairing with replacement from the smaller set is used so that all posterior samples and stochastic maps contribute to the analysis.
- `"full"`: Exhaustive option that accounts for trait/range- and rate- uncertainty by crossing all BAMM posterior samples with all stochastic maps. Accounts for both diversification-rate and ancestral reconstruction uncertainty by evaluating every combination of BAMM posterior samples and stochastic maps. WARNING: This exhaustive approach can substantially increase computation time and memory requirements and is therefore recommended only for moderate-sized analyses.

`trait_maps_vs_BAMM_samples_list`
: (Optional) List of two elements manually providing the names to associate stochastic maps (`$trait_map_ID`) with BAMM samples (`$BAMM_posterior_sample_ID`). This is typically used to ensure the same stochastic maps and BAMM samples are used to test across multiple time-steps. Values are the names of the objects such as "Map_X" and "BAMM_X". Default = `NULL`. * For uncertainty_strategy == 'rates_only', the `$trait_map_ID` must be "Map_ML" as only the ML estimates of trait values/states/ranges are used. * For uncertainty_strategy == 'paired', each pair of stochastic map and BAMM sample will be used once. * For uncertainty_strategy == 'full', all stochastic maps will be matched with all BAMM samples. Those may partly differ from the actual maps and BAMM samples used for the tests as recorded in `$STRAPP_results_over_time[[X]]$perm_data_df` because invalid maps with not enough states/ranges are discarded at each time-step.

`seed`
: Integer. Set the seed to ensure reproducibility. Default is `NULL` (a random seed is used).

`nb_permutations`
: Integer. To select the number of random permutations to perform during the tests. If NULL (default), all posterior samples will be used once.

`alpha`
: Numeric. Significance level to use to compute the `estimate` corresponding to the values of the test statistic used to assess significance of the test. This does NOT affect p-values. Default is `0.05`.

`two_tailed`
: Logical. To define the type of tests. If `TRUE` (default), tests for correlations/differences in rates will be carried out with a null hypothesis that rates are not correlated with trait values (continuous data) or equal between trait states (categorical and biogeographic data). If `FALSE`, one-tailed tests are carried out.
  - For continuous data, it involves defining a `one_tailed_hypothesis` testing for either a "positive" or "negative" correlation under the alternative hypothesis.
  - For binary data (two states), it involves defining a `one_tailed_hypothesis` indicating which states have higher rates under the alternative hypothesis.
  - For multinomial data (more than two states), it defines the type of post hoc pairwise tests to carry out between pairs of states. If `posthoc_pairwise_tests = TRUE`, all two-tailed (if `two_tailed = TRUE`) or one-tailed (if `two_tailed = FALSE`) tests are automatically carried out.

`one_tailed_hypothesis`
: A character string specifying the alternative hypothesis in the one-tailed test. For continuous data, it is either "negative" or "positive" correlation. For binary data, it lists the trait states with states ordered in increasing rates under the alternative hypothesis, separated by a greater-than such as c('A > B').

`posthoc_pairwise_tests`
: Logical. Only for multinomial data (with more than two states). If `TRUE`, all possible post hoc pairwise (Dunn) tests will be computed across all pairs of states. This is a way to detect which pairs of states have significant differences in rates if the overall test (Kruskal-Wallis) is significant. Default is `FALSE`.

`p.adjust_method`
: A character string. Only for multinomial data (with more than two states). It

specifies the type of correction to apply to the p-values in the post hoc pairwise tests to account for multiple comparisons. See `stats::p.adjust()` for the available methods. Default is `none`.

`return_perm_data`
: Logical. Whether to return the stats data computed from the posterior samples for observed and permuted data in the output. This is needed to plot the histograms of the null distribution used to assess significance of the tests with `plot_histogram_STRAPP_test_for_focal_time()`. (for a single `focal_time`) and `plot_histograms_STRAPP_tests_over_time()` (for multiple `time_steps`). Default is `FALSE`.

`nthreads`
: Integer. Number of threads to use for parallel computing of the STRAPP tests across the permutations. The R package `parallel` must be loaded for `nthreads` > 1. Default is 1.

`print_hypothesis`
: Logical. Whether to print information on what test is carried out, detailing the null and alternative hypotheses, what significance level is used to reject or not the null hypothesis, and how uncertainty in trait estimates is handled. Default is `TRUE`.

`extract_trait_data_melted_df`
: Logical. Specify whether trait data must be extracted from the mapped phylogenies at each time step and returned in a melted data.frame. Default is `FALSE`.

`extract_diversification_data_melted_df`
: Logical. Specify whether diversification data (regimes ID and tip rates) must be extracted from the `updated_BAMM_object` at each time step and returned in a melted data.frame. Default is `FALSE`.

`return_STRAPP_results`
: Logical. Specify whether the `STRAPP_results` objects summarizing the results of the STRAPP tests carried out at each time step should be returned among the outputs in addition to the `$pvalues_summary_df` already providing test stat estimates and p-values obtained across all `time_steps`.

`return_updated_Maps`
: Logical. Specify whether the updated versions of the mapped phylogenies (`contMap(s)/densityMaps/si` should be returned among the outputs. The update of the `contMap(s)/densityMaps/simmaps` consists of cutting off branches and mapping that are younger than the `time_steps` used to perform the tests. Default is `FALSE`.

`return_updated_BAMM_objects`
: Logical. Specify whether the `updated_BAMM_objects` with phylogeny and mapped diversification rates cut-off at the `time_steps` used to perform the tests should be returned among the outputs.

`verbose`
: Logical. Should progression per `time_steps` be displayed? Default is `TRUE`.

`verbose_extended`
: Should progression per `time_steps` AND within each deepSTRAPP workflow be displayed? In addition to printing progress along `time_steps`, a message will be printed at each step of the deepSTRAPP workflow, and for every batch of 100 BAMM posterior samples whose rates and regimes are updated. If `extract_diversification_data_melted_df = TRUE`, a message will also be printed when rates are extracted. Default is `FALSE`.

Details

The function is a wrapper of run_deepSTRAPP_for_focal_time() that runs the deepSTRAPP workflow over multiple time_steps.

The deepSTRAPP workflow is described step by step in the run_deepSTRAPP_for_focal_time() documentation.

Its main output is the $pvalues_summary_df: a data.frame providing test stat estimates and p-values obtained across all time_steps, that can be passed down to plot_STRAPP_pvalues_over_time() to generate a plot showing the evolution of the test results across time. If using multinomial data (with more than two states) and posthoc_pairwise_tests = TRUE, the output will also contain a data.frame providing test stat estimates and p-values for post hoc pairwise tests in $pvalues_summary_df_for_posthoc_pai

The function offers options to generate summary data.frames of the data extracted across time_steps:

- If extract_trait_data_melted_df = TRUE, a data.frame of trait values found along branches at each time step is provided in $trait_data_df_over_time.
- If extract_diversification_data_melted_df = TRUE, a data.frame of diversification data (regimes ID and tip rates) found along branches at each time step is provided in $diversification_data_df_over_tir
- Those data.frames can be passed down to plot_rates_through_time() to generate a plot showing the evolution of diversification rates across trait values over time.

The function also allows you to keep records of the intermediate objects generated during the STRAPP workflow:

- If return_STRAPP_results = TRUE, a list of STRAPP test outputs is provided in $STRAPP_results_over_time. Combined with return_perm_data = TRUE, it allows you to plot the histograms of the null distributions used to assess significance of the tests with plot_histogram_STRAPP_test_for_focal_time(). (for a single focal_time) and plot_histograms_STRAPP_tests_over_time() (for multiple time_steps).
- If return_updated_Maps = TRUE, a list of objects containing the updated contMap(s)/densityMaps/simmaps provided as inputs is provided in $updated_Maps_over_time. The updated mapped phylogenies can be visualized with plot_contMap() or plot_densityMaps_overlay(), to display a phylogeny mapped with trait values/states/ranges with branches cut at each focal_time.
- If return_updated_BAMM_objects = TRUE, a list of updated BAMM_objects of class "bammdata" that contains rates and regimes ID found at each focal_time is provided in $updated_BAMM_objects_over_time. Updated BAMM_objects can be plotted with plot_BAMM_rates() to display a phylogeny mapped with diversification rates with branches cut at each focal_time.

Value

The function returns a list with at least five elements.

- $pvalues_summary_df Data.frame with three columns providing test stat $estimate and $p_value obtained for each time step (i.e., $focal_time), that can be passed down to plot_STRAPP_pvalues_over_ti to generate a plot showing the evolution of the test results across time.
- $time_steps Numeric vector. Time steps at which the STRAPP tests were carried out in the same order as the objects returned in the output lists.
- $trait_data_type Character string. Specify the type of trait data. Possible values are: "continuous", "categorical", "biogeographic".

- $trait_data_type_for_stats Character string. The type of trait data used to select statistical method. One of 'continuous', 'binary', or 'multinomial'. The method is fixed once for the whole run, based on the number of states/ranges found in the complete trait mapping, so that the same test is used at every time step for consistency across p-values.
- $states_observed_overall (Only for categorical and biogeographic data) Character string vector of all states/ranges found in the complete trait mapping. This is what the statistical method trait_data_type_for_stats is based on, and it can be larger than the set of states/ranges found at any given time step.
- $states_observed_per_time_steps (Only for categorical and biogeographic data) List of character string vectors, one per time step and named after it. States/ranges found at each time step. There can be steps with fewer states/ranges than others, which may affect the interpretation of the p-values obtained for these steps.
- $nb_states_observed_per_time_steps (Only for categorical and biogeographic data) Named numeric vector. Number of states/ranges found at each time step. Recorded so that the interpretation of the p-values obtained from steps with fewer states/ranges can be adjusted if needed. For cases with only one state/range, no test could be computed, and they should be matching a NA in $pvalues_summary_df.
- $rate_type Character string. The type of diversification rates used in the tests: 'speciation', 'extinction' or 'net_diversification'.
- $uncertainty_strategy Character string. The strategy used to account for uncertainty in estimates. One of 'rates_only', 'paired', or 'full'.
- $trait_maps_vs_BAMM_samples_list List of two elements recording the stochastic maps ($trait_map_ID) and BAMM samples ($BAMM_posterior_sample_ID) chosen for testing across time-steps. Those may partly differ from the actual maps and BAMM samples used for the tests as recorded in $STRAPP_results_over_time[[X]]$perm_data_df because invalid maps with not enough states/ranges are discarded.

Optional summary df for multinomial data, if posthoc_pairwise_tests = TRUE:

- $pvalues_summary_df_for_posthoc_pairwise_tests Data.frame with four or five columns providing test stat $estimate, $p_value, and $p_value_adjusted (if p.adjust_method used is not "none") for each $pair of states involved in post hoc Dunn's tests obtained for each time step (i.e., $focal_time). This data.frame can be passed down to plot_STRAPP_pvalues_over_time() to generate a plot showing the evolution of the post hoc test results across time.

Optional melted data.frames:

- $trait_data_df_over_time Data.frame with five columns providing $trait_value associated with each $tip_ID found along each time step (i.e., $focal_time) across all stochastic maps ($Map_ID). Set extract_trait_data_melted_df = TRUE to include it in the output.
- $diversification_data_df_over_time Data.frame with six columns providing diversification regimes ($regime_ID) and $rates sorted by $rate_type along tips ($tip_ID) found across all posterior samples ($BAMM_sample_ID) over each time step (i.e., $focal_time). Set extract_diversification_data_melted_df = TRUE to include it in the output.
- Those data.frames can be passed down to plot_rates_through_time() to generate a plot showing the evolution of diversification rates across trait values over time.

Optional objects generated for each time step (i.e., focal_time) and ordered as in $time_steps:

- $STRAPP_results_over_time List of objects summarizing the results of the STRAPP tests See compute_STRAPP_test_for_focal_time() for a detailed description of the elements in each object. Set return_STRAPP_results = TRUE to include it in the output. Combined with return_perm_data = TRUE, it allows you to plot the histograms of the null distributions used to assess significance of the tests with plot_histogram_STRAPP_test_for_focal_time(). (for a single focal_time) and plot_histograms_STRAPP_tests_over_time() (for multiple time_steps).
- $updated_Maps_over_time List of objects containing updated contMap(s)/densityMaps/simmaps. The updated mapped phylogenies can be plotted with plot_contMap() or plot_densityMaps_overlay(), to display a phylogeny mapped with trait values with branches cut at each focal_time.
- $updated_BAMM_objects_over_time List of objects containing rates and regimes ID mapped on phylogeny. Updated BAMM_objects can be plotted with plot_BAMM_rates() to display a phylogeny mapped with diversification rates with branches cut at each focal_time.

Author(s)

Maël Doré

See Also

To run the deepSTRAPP workflow for a single focal_time: run_deepSTRAPP_for_focal_time() extract_most_likely_trait_values_for_focal_time() update_rates_and_regimes_for_focal_time() extract_diversification_data_melted_df_for_focal_time() compute_STRAPP_test_for_focal_time()

For a guided tutorial on complete deepSTRAPP workflow, see the associated vignettes:

- For continuous trait data: vignette("deepSTRAPP_continuous_data", package = "deepSTRAPP")
- For categorical trait data: vignette("deepSTRAPP_categorical_3lvl_data", package = "deepSTRAPP")
- For biogeographic range data: vignette("deepSTRAPP_biogeographic_data", package = "deepSTRAPP")

Examples

```
if (deepSTRAPP::is_dev_version())
{
# ----- Example 1: Continuous trait ----- #
 ## Load data

 # Load trait df
data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny with old calibration
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare trait data
```

```
# Extract continuous trait data as a named vector
Ponerinae_cont_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cont_tip_data,
                                    nm = Ponerinae_trait_tip_data$Taxa)

# Select a color scheme from lowest to highest values
color_scale = c("darkgreen", "limegreen", "orange", "red")

 # (May take several minutes to run)
# Map trait evolution across multiple simulations (i.e., continuous stochastic maps)
Ponerinae_cont_data_old_calib <- prepare_trait_data(
    tip_data = Ponerinae_cont_tip_data,
    trait_data_type = "continuous",
    phylo = Ponerinae_tree_old_calib,
    seed = 1234,
    evolutionary_models = "BM",
    plot_map = FALSE,
    verbose = TRUE)

# Plot contMap = ML estimates of continuous trait evolution
plot_contMap(contMap = Ponerinae_cont_data_old_calib$contMap,
             color_scale = color_scale)

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

## Run deepSTRAPP on net diversification rates
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- run_deepSTRAPP_over_time(
   contMap = Ponerinae_cont_data_old_calib$contMap, # Provide ML estimates as a contMap
   ace = Ponerinae_cont_data_old_calib$ace,
   tip_data = Ponerinae_cont_tip_data,
   trait_data_type = "continuous",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "rates_only",
   rate_type = "net_diversification",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly deepSTRAPP output
Ponerinae_deepSTRAPP_cont_old_calib_0_40 <- readRDS(system.file("extdata",
    "Ponerinae_deepSTRAPP_cont_old_calib_0_40.rds", package = "deepSTRAPP"))
## This dataset is only available in development versions installed from GitHub.
```

```
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Explore output
str(Ponerinae_deepSTRAPP_cont_old_calib_0_40, max.level = 1)
 deepSTRAPP_outputs <- Ponerinae_deepSTRAPP_cont_old_calib_0_40

 # Display test summary
 # Can be passed down to [deepSTRAPP::plot_STRAPP_pvalues_over_time()] to generate a plot
 # showing the evolution of the test results across time.
 deepSTRAPP_outputs$pvalues_summary_df

 # Access trait data in a melted data.frame
 head(deepSTRAPP_outputs$trait_data_df_over_time)
 # Trait data includes only the ML estimates
 table(deepSTRAPP_outputs$trait_data_df_over_time$Map_ID)

 # Access the diversification data in a melted data.frame
 head(deepSTRAPP_outputs$diversification_data_df_over_time)
 # Diversification data includes 1000 BAMM posteriors
 table(deepSTRAPP_outputs$diversification_data_df$BAMM_sample_ID)

 # Access deepSTRAPP results ordered by time-steps
 str(deepSTRAPP_outputs$STRAPP_results, max.level = 2)

 ## Visualize updated phylogenies

  # (May take time to plot)
 # Plot updated contMap for time step n°2
 contMap_step2 <- deepSTRAPP_outputs$updated_Maps_over_time[[2]]$contMap
 plot_contMap(contMap_step2, ftype = "off", color_scale = color_scale)

 # Plot diversification rates on updated phylogeny for time step n°2
 BAMM_object_step2 <- deepSTRAPP_outputs$updated_BAMM_objects_over_time[[2]]
 plot_BAMM_rates(BAMM_object = BAMM_object_step2,
   legend = TRUE, labels = FALSE,
   colorbreaks = BAMM_object_step2$initial_colorbreaks$net_diversification)

 ## Visualize test results

  # (May take time to plot)
 # Plot p-values of Spearman tests across all time-steps
 plot_STRAPP_pvalues_over_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
    alpha = 0.05)

 # Plot evolution of mean rates through time
 plot_rates_through_time(deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
    plot_CI = TRUE)

 # Plot rates vs. trait values across branches for time step = 10 My
 plot_rates_vs_trait_data_for_focal_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
```

```
    focal_time = 10)

# Plot histogram of Spearman test stats for time step = 10 My
 plot_histogram_STRAPP_test_for_focal_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
    focal_time = 10)

 # Plot histograms of Spearman test results (One plot per time-step)
 plot_histograms_STRAPP_tests_over_time(
    deepSTRAPP_outputs = Ponerinae_deepSTRAPP_cont_old_calib_0_40,
    display_plots = TRUE)

 # ----- Example 2: Categorical trait ----- #

## Load data

 # Load trait df
 data(Ponerinae_trait_tip_data, package = "deepSTRAPP")
 # Load phylogeny
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare trait data

 # Extract categorical data with 3-levels
Ponerinae_cat_3lvl_tip_data <- setNames(object = Ponerinae_trait_tip_data$fake_cat_3lvl_tip_data,
                                        nm = Ponerinae_trait_tip_data$Taxa)
 table(Ponerinae_cat_3lvl_tip_data)

 # Select color scheme for states
 colors_per_states <- c("forestgreen", "sienna", "goldenrod")
 names(colors_per_states) <- c("arboreal", "subterranean", "terricolous")

  # (May take several minutes to run)
 ## Produce densityMaps using stochastic character mapping based on an ARD Mk model
 Ponerinae_cat_3lvl_data_old_calib <- prepare_trait_data(
    tip_data = Ponerinae_cat_3lvl_tip_data,
    phylo = Ponerinae_tree_old_calib,
    trait_data_type = "categorical",
    colors_per_levels = colors_per_states,
    evolutionary_models = "ER", # Use default ER model
    nb_simulations = 100, # Reduce number of simulations to save time
    seed = 1234, # Set seed for reproducibility
    return_best_model_fit = TRUE,
    return_model_selection_df = TRUE,
    plot_map = FALSE)

  # Load directly trait data output
```

```
data(Ponerinae_cat_3lvl_data_old_calib, package = "deepSTRAPP")

## Set for time steps of 5 My. Will generate deepSTRAPP workflows for 0 to 40 Mya.
# nb_time_steps <- 5
time_step_duration <- 5
time_range <- c(0, 40)

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates across time-steps.
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- run_deepSTRAPP_over_time(
   densityMaps = Ponerinae_cat_3lvl_data_old_calib$densityMaps,
  # Inform the nb of simulations to reconstruct state distribution across stochastic maps
   nb_simulations = 100,
   ace = Ponerinae_cat_3lvl_data_old_calib$ace,
   tip_data = Ponerinae_cat_3lvl_tip_data,
   trait_data_type = "categorical",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   # nb_time_steps = nb_time_steps,
   time_range = time_range,
   time_step_duration = time_step_duration,
   uncertainty_strategy = "paired",
   rate_type = "net_diversification",
   seed = 1234,
   alpha = 0.10, # Select a generous level of significance for the sake of the example
   posthoc_pairwise_tests = TRUE,
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly deepSTRAPP output
Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40 <- readRDS(system.file("extdata",
    "Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40.rds", package = "deepSTRAPP"))
deepSTRAPP_outputs <- Ponerinae_deepSTRAPP_cat_3lvl_old_calib_0_40
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Explore output
str(deepSTRAPP_outputs, max.level = 1)

# Display test summaries
# Can be passed down to [deepSTRAPP::plot_STRAPP_pvalues_over_time()] to generate a plot
# showing the evolution of the test results across time.
deepSTRAPP_outputs$pvalues_summary_df
# Results for posthoc pairwise Dunn's tests over time-steps
deepSTRAPP_outputs$pvalues_summary_df_for_posthoc_pairwise_tests

# Access trait data in a melted data.frame
```

```
head(deepSTRAPP_outputs$trait_data_df_over_time)
# Trait data were extracted from densityMaps recording only frequency of states.
# Thus, trait states have been distributed accordingly across 100 'Dummy_maps'
# to reproduce the recorded frequencies.
table(deepSTRAPP_outputs$trait_data_df_over_time$Map_ID)

# Access the diversification data in a melted data.frame
head(deepSTRAPP_outputs$diversification_data_df_over_time)
# Diversification data includes 1000 BAMM posteriors
table(deepSTRAPP_outputs$diversification_data_df_over_time$BAMM_sample_ID)

# Access details of deepSTRAPP results across time-steps
str(deepSTRAPP_outputs$STRAPP_results, max.level = 2)

## Visualize updated phylogenies

 # (May take time to plot)
# Plot updated densityMaps for time step n°2 = 10My
densityMaps_10My <- deepSTRAPP_outputs$updated_Maps_over_time[[2]]
plot_densityMaps_overlay(densityMaps_10My$densityMaps, colors_per_levels = colors_per_states)

# Plot diversification rates on updated phylogeny for time step n°2
BAMM_object_10My <- deepSTRAPP_outputs$updated_BAMM_objects_over_time[[2]]
plot_BAMM_rates(BAMM_object = BAMM_object_10My,
   legend = TRUE, labels = FALSE,
  colorbreaks = BAMM_object_10My$initial_colorbreaks$net_diversification)

## Visualize test results

 # (May take time to plot)
# Plot p-values of overall Kruskal-Wallis test across all time-steps
plot_STRAPP_pvalues_over_time(
   deepSTRAPP_outputs = deepSTRAPP_outputs,
  alpha = 0.10)

# Plot p-values of post hoc pairwise Dunn's tests between pairs of tests across all time-steps
plot_STRAPP_pvalues_over_time(
   deepSTRAPP_outputs = deepSTRAPP_outputs,
   alpha = 0.10,
   plot_posthoc_tests = TRUE)

# Plot evolution of mean rates through time
plot_rates_through_time(deepSTRAPP_outputs = deepSTRAPP_outputs,
   colors_per_levels = colors_per_states, plot_CI = TRUE)

# Plot rates vs. trait values across branches for time step n°2 = 10 My
plot_rates_vs_trait_data_for_focal_time(
   deepSTRAPP_outputs = deepSTRAPP_outputs,
   focal_time = 10,
   colors_per_levels = colors_per_states)

# Plot histogram of overall Kruskal-Wallis test for time step n°2 = 10 My
plot_histogram_STRAPP_test_for_focal_time(
```

```
    deepSTRAPP_outputs = deepSTRAPP_outputs,
    focal_time = 10)

 # Plot histograms of overall Kruskal-Wallis test results across all time-steps
 # (One plot per time-step)
 plot_histograms_STRAPP_tests_over_time(
    deepSTRAPP_outputs = deepSTRAPP_outputs,
    plot_posthoc_tests = FALSE)

 # Plot histograms of post hoc pairwise Dunn's test results across all time-steps
 # (One multifaceted plot per time-step)
 plot_histograms_STRAPP_tests_over_time(
    deepSTRAPP_outputs = deepSTRAPP_outputs,
    plot_posthoc_tests = TRUE)

# ----- Example 3: Biogeographic ranges ----- #

## Load data

 # Load phylogeny
 data(Ponerinae_tree_old_calib, package = "deepSTRAPP")
 # Load trait df
 data(Ponerinae_binary_range_table, package = "deepSTRAPP")

 # Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object_old_calib, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Prepare range data for Old World vs. New World

 # No overlap in ranges
 table(Ponerinae_binary_range_table$Old_World, Ponerinae_binary_range_table$New_World)

 Ponerinae_NO_tip_data <- stats::setNames(object = Ponerinae_binary_range_table$Old_World,
                                          nm = Ponerinae_binary_range_table$Taxa)
 Ponerinae_NO_tip_data <- as.character(Ponerinae_NO_tip_data)
 Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "TRUE"] <- "O" # O = Old World
 Ponerinae_NO_tip_data[Ponerinae_NO_tip_data == "FALSE"] <- "N" # N = New World
 names(Ponerinae_NO_tip_data) <- Ponerinae_binary_range_table$Taxa
 table(Ponerinae_NO_tip_data)

 colors_per_ranges <- c("mediumpurple2", "peachpuff2")
 names(colors_per_ranges) <- c("N", "O")

  # (May take several minutes to run)

 # Load ape and BioGeoBEARS to use models internally
 library(ape)
 library(BioGeoBEARS)

 ## Run evolutionary models
```

```
Ponerinae_biogeo_data <- prepare_trait_data(
  tip_data = Ponerinae_NO_tip_data,
   trait_data_type = "biogeographic",
   phylo = Ponerinae_tree_old_calib,
   evolutionary_models = "DEC+J", # Default = "DEC" for biogeographic
   BioGeoBEARS_directory_path = tempdir(), # Ex: "./BioGeoBEARS_directory/"
   keep_BioGeoBEARS_files = FALSE,
   prefix_for_files = "Ponerinae_old_calib",
   max_range_size = 2,
   split_multi_area_ranges = TRUE, # Set to TRUE to use only unique areas "A" and "B"
   nb_simulations = 100, # Reduce to save time (Default = '1000')
   colors_per_levels = colors_per_ranges,
   return_model_selection_df = TRUE,
   verbose = TRUE)

# Load directly output
data(Ponerinae_biogeo_data_old_calib, package = "deepSTRAPP")

## Set for time steps of 5 My. Will generate deepSTRAPP workflows from 0 to 40 Mya.
time_range <- c(0, 40)
time_step_duration <- 5

 # (May take several minutes to run)
## Run deepSTRAPP on net diversification rates for time-steps = 0 to 40 Mya.
Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- run_deepSTRAPP_over_time(
  densityMaps = Ponerinae_biogeo_data_old_calib$densityMaps,
 # Inform the nb of simulations to reconstruct state distribution across stochastic maps
   nb_simulations = 100,
   ace = Ponerinae_biogeo_data_old_calib$ace,
   tip_data = Ponerinae_NO_tip_data,
   trait_data_type = "biogeographic",
   BAMM_object = Ponerinae_BAMM_object_old_calib,
   time_range = time_range,
   time_step_duration = time_step_duration,
   seed = 1234, # Set seed for reproducibility
   alpha = 0.10, # Select a generous level of significance for the sake of the example
   uncertainty_strategy = "paired",
   rate_type = "net_diversification",
   return_perm_data = TRUE,
   extract_trait_data_melted_df = TRUE,
   extract_diversification_data_melted_df = TRUE,
   return_STRAPP_results = TRUE,
   return_updated_Maps = TRUE,
   return_updated_BAMM_objects = TRUE,
   verbose = TRUE,
   verbose_extended = TRUE)

## Load directly output
Ponerinae_deepSTRAPP_biogeo_old_calib_0_40 <- readRDS(system.file("extdata",
   "Ponerinae_deepSTRAPP_biogeo_old_calib_0_40.rds", package = "deepSTRAPP"))
deepSTRAPP_outputs <- Ponerinae_deepSTRAPP_biogeo_old_calib_0_40
## This dataset is only available in development versions installed from GitHub.
# It is not available in CRAN versions.
```

```
# Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

## Explore output
str(deepSTRAPP_outputs, max.level = 1)

# Display test summaries
# Can be passed down to [deepSTRAPP::plot_STRAPP_pvalues_over_time()] to generate a plot
# showing the evolution of the test results across time.
deepSTRAPP_outputs$pvalues_summary_df

# Access bioregeographic range data in a melted data.frame
head(deepSTRAPP_outputs$trait_data_df_over_time)
# Ranges were extracted from densityMaps recording only frequency of states.
# Thus, ranges have been distributed accordingly across 'Dummy_maps'
# to reproduce the recorded frequencies.
table(deepSTRAPP_outputs$trait_data_df_over_time$Map_ID)

# Access the diversification data in a melted data.frame
head(deepSTRAPP_outputs$diversification_data_df_over_time)
# Diversification data includes 1000 BAMM posteriors
table(deepSTRAPP_outputs$diversification_data_df_over_time$BAMM_sample_ID)

# Access details of deepSTRAPP results
str(deepSTRAPP_outputs$STRAPP_results, max.level = 2)

## Visualize updated phylogenies

 # (May take time to plot)
# Plot updated densityMaps for time step n°2 = 10My
densityMaps_10My <- deepSTRAPP_outputs$updated_Maps_over_time[[2]]
plot_densityMaps_overlay(densityMaps_10My$densityMaps)

# Plot diversification rates on updated phylogeny for time step n°2
BAMM_object_10My <- deepSTRAPP_outputs$updated_BAMM_objects_over_time[[2]]
plot_BAMM_rates(BAMM_object = BAMM_object_10My,
  legend = TRUE, labels = FALSE,
  colorbreaks = BAMM_object_10My$initial_colorbreaks$net_diversification)

## Visualize test results

 # (May take time to plot)
# Plot p-values of Mann-Whitney-Wilcoxon tests across all time-steps
plot_STRAPP_pvalues_over_time(
   deepSTRAPP_outputs = deepSTRAPP_outputs,
   alpha = 0.10) # Set a generous significance threshold for the example

# Plot evolution of mean rates through time
plot_rates_through_time(deepSTRAPP_outputs = deepSTRAPP_outputs,
  colors_per_levels = colors_per_ranges, plot_CI = TRUE)

# Plot rates vs. trait values across branches for time step n°2 = 10 My
plot_rates_vs_trait_data_for_focal_time(
   deepSTRAPP_outputs = deepSTRAPP_outputs,
```

```
    focal_time = 10,
    colors_per_levels = colors_per_ranges)

 # Plot histogram of Mann-Whitney-Wilcoxon test stats for time step n°2 = 10My
 plot_histogram_STRAPP_test_for_focal_time(
    deepSTRAPP_outputs = deepSTRAPP_outputs,
   focal_time = 10)

# Plot histograms of Mann-Whitney-Wilcoxon test stats for all time-steps (One plot per time-step)
 plot_histograms_STRAPP_tests_over_time(
    deepSTRAPP_outputs = deepSTRAPP_outputs,
    display_plots = TRUE,
    plot_posthoc_tests = FALSE)
}
```

`select_best_model_from_BioGeoBEARS` *Compare model fits with AICc and Akaike's weights*

Description

Compare models fit with `BioGeoBEARS::bear_optim_run()` using AICc and Akaike's weights. Generate a data.frame summarizing information. Identify the best model and extract its results.

Usage

```
select_best_model_from_BioGeoBEARS(list_model_fits)
```

Arguments

`list_model_fits`
: Named list with the results of a model fit with `BioGeoBEARS::bear_optim_run()` in each element.

Value

The function returns a list with three elements.

- `$model_comparison_df` Data.frame summarizing information to compare model fits. It includes the model name (`$model`), the log-likelihood (`$logLik`), the number of free-parameters (`$k`), the AIC (`$AIC`), the corrected AIC (`$AICc`), the delta to the best/lowest AICc (`$delta_AICc`), the Akaike weights (`$Akaike_weights`), and their rank based on AICc (`$rank`).
- `$best_model_name` Character string. Name of the best model.
- `$best_model_fit` List containing the output of `BioGeoBEARS::bear_optim_run()` for the model with the best fit.

Author(s)

Maël Doré

See Also

BioGeoBEARS::bear_optim_run() BioGeoBEARS::get_LnL_from_BioGeoBEARS_results_object()
BioGeoBEARS::AICstats_2models()

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## The R package 'BioGeoBEARS' is needed for this function to work with biogeographic data.
 # Please install it manually from: https://github.com/nmatzke/BioGeoBEARS.

 # Load phylogeny and tip data
 library(phytools)
 data(eel.tree)
 data(eel.data)

 # Transform feeding mode data into biogeographic data with ranges A, B, and AB.
 eel_data <- stats::setNames(eel.data$feed_mode, rownames(eel.data))
 eel_data <- as.character(eel_data)
 eel_data[eel_data == "bite"] <- "A"
 eel_data[eel_data == "suction"] <- "B"
 eel_data[c(5, 6, 7, 15, 25, 32, 33, 34, 50, 52, 57, 58, 59)] <- "AB"
 eel_data <- stats::setNames(eel_data, rownames(eel.data))
 table(eel_data)

 colors_per_levels <- c("dodgerblue3", "gold")
 names(colors_per_levels) <- c("A", "B")

  # (May take several minutes to run)
 ## Prepare phylo

 # Set path to BioGeoBEARS directory
 BioGeoBEARS_directory_path = tempfile("BioGeoBEARS_directory_")
 dir.create(BioGeoBEARS_directory_path, recursive = TRUE, showWarnings = FALSE)

 # Export phylo
 path_to_phylo <- BioGeoBEARS::np(file.path(BioGeoBEARS_directory_path, "phylo.tree"))
 write.tree(phy = eel.tree, file = path_to_phylo)

 ## Prepare range data

 tip_data <- eel_data
 # Convert tip_data to df
 ranges_df <- as.data.frame(tip_data)
 # Get list of all unique areas
 unique_areas_in_ranges_list <- strsplit(x = tip_data, split = "")
 unique_areas <- unique(unlist(unique_areas_in_ranges_list))
 unique_areas <- unique_areas[order(unique_areas)]

 # Loop per unique area
 for (i in seq_along(unique_areas))
 {
```

```
  # i <- 1

  # Extract unique area
  unique_area_i <- unique_areas[i]
  # Detect presence in ranges
  binary_match_i <- unlist(lapply(X = unique_areas_in_ranges_list,
     FUN = function (x) { unique_area_i %in% x } ))

  # Add to ranges_df
  ranges_df <- cbind(ranges_df, binary_match_i)
}
# Extract binary df of presence/absence
binary_df <- ranges_df[, -1]
# Convert character strings into numeric factors
binary_df_num <- as.data.frame(apply(X = binary_df, MARGIN = 2, FUN = as.numeric))
row.names(binary_df_num) <- names(tip_data)
names(binary_df_num) <- unique_areas

# Produce tipranges object from numeric df
Taxa_bioregions_tipranges_obj <- BioGeoBEARS::define_tipranges_object(tmpdf = binary_df_num)

# Set path to tip ranges object
path_to_tip_ranges <- BioGeoBEARS::np(file.path(BioGeoBEARS_directory_path, "tip_ranges.data"))

# Export tip ranges in Lagrange/PHYLIP format
BioGeoBEARS::save_tipranges_to_LagrangePHYLIP(
  tipranges_object = Taxa_bioregions_tipranges_obj,
  lgdata_fn = path_to_tip_ranges,
  areanames = colnames(Taxa_bioregions_tipranges_obj@df))

## Prepare models

# Prepare DEC model run
DEC_run <- BioGeoBEARS::define_BioGeoBEARS_run(
   num_cores_to_use = 1,
   max_range_size = 2, # To set the maximum number of bioregion encompassed
     # by a lineage range at any time
   trfn = path_to_phylo, # To provide path to the input tree file
  geogfn = path_to_tip_ranges, # To provide path to the LagrangePHYLIP file with binary ranges
   return_condlikes_table = TRUE, # To ask to obtain all marginal likelihoods
     # computed by the model and used to display ancestral states
   print_optim = TRUE)

# Check that starting parameter values are inside the min/max
DEC_run <- BioGeoBEARS::fix_BioGeoBEARS_params_minmax(BioGeoBEARS_run_object = DEC_run)
# Check validity of set-up before run
BioGeoBEARS::check_BioGeoBEARS_run(DEC_run)

# Use the DEC model as template for DEC+J model
DEC_J_run <- DEC_run

# Update status of jump speciation parameter to be estimated
DEC_J_run$BioGeoBEARS_model_object@params_table["j","type"] <- "free"
```

```
# Set initial value of J for optimization to an arbitrary low non-null value
j_start <- 0.0001
DEC_J_run$BioGeoBEARS_model_object@params_table["j","init"] <- j_start
DEC_J_run$BioGeoBEARS_model_object@params_table["j","est"] <- j_start

# Check validity of set-up before run
DEC_J_run <- BioGeoBEARS::fix_BioGeoBEARS_params_minmax(BioGeoBEARS_run_object = DEC_J_run)
invisible(BioGeoBEARS::check_BioGeoBEARS_run(DEC_J_run))

## Run models

# Run DEC model
DEC_fit <- BioGeoBEARS::bears_optim_run(DEC_run)

# Set starting values for optimization of DEC+J model based on MLE in the DEC model
d_start <- DEC_fit$outputs@params_table["d","est"]
e_start <- DEC_fit$outputs@params_table["e","est"]

DEC_J_run$BioGeoBEARS_model_object@params_table["d","init"] <- d_start
DEC_J_run$BioGeoBEARS_model_object@params_table["d","est"] <- d_start
DEC_J_run$BioGeoBEARS_model_object@params_table["e","init"] <- e_start
DEC_J_run$BioGeoBEARS_model_object@params_table["e","est"] <- e_start

# Run DEC+J model
DEC_J_fit <- BioGeoBEARS::bears_optim_run(DEC_J_run)

## Store model outputs
list_model_fits <- list(DEC = DEC_fit, DEC_J = DEC_J_fit)

## Compare models
model_comparison_output <- select_best_model_from_BioGeoBEARS(list_model_fits = list_model_fits)

# Explore output
str(model_comparison_output, max.level = 1)
# Print comparison
print(model_comparison_output$models_comparison_df)
# Print best model fit
print(model_comparison_output$best_model_fit$outputs)

## Clean local files
unlink(BioGeoBEARS_directory_path, recursive = TRUE, force = TRUE)
}
```

`select_best_trait_model_from_geiger`

Compare trait evolutionary model fits with AICc and Akaike's weights

---

Description

Compare evolutionary models fit with `geiger::fitContinuous()` or `geiger::fitDiscrete()` using AICc and Akaike's weights. Generate a data.frame summarizing information. Identify the best model and extract its results.

Usage

```
select_best_trait_model_from_geiger(list_model_fits)
```

Arguments

`list_model_fits`
: Named list with the results of a model fit with `geiger::fitContinuous()` or `geiger::fitDiscrete()` in each element.

Value

The function returns a list with three elements.

- `$model_comparison_df` Data.frame summarizing information to compare model fits. It includes the model name (`$model`), the log-likelihood (`$logLik`), the number of free-parameters (`$k`), the AIC (`$AIC`), the corrected AIC (`$AICc`), the delta to the best/lowest AICc (`$delta_AICc`), the Akaike weights (`$Akaike_weights`), and their rank based on AICc (`$rank`).
- `$best_model_name` Character string. Name of the best model.
- `$best_model_fit` List containing the output of `geiger::fitContinuous()` or `geiger::fitDiscrete()` for the model with the best fit.

Author(s)

Maël Doré

See Also

`geiger::fitContinuous()` `geiger::fitDiscrete()`

Examples

```
# ----- Example 1: Continuous data ----- #

# Load phylogeny and tip data
library(phytools)
data(eel.tree)
data(eel.data)

# Extract body size
eel_data <- stats::setNames(eel.data$Max_TL_cm,
                            rownames(eel.data))

 # (May take several minutes to run)
# Fit BM model
BM_fit <- geiger::fitContinuous(phy = eel.tree, dat = eel_data, model = "BM")
```

```
# Fit EB model
EB_fit <- geiger::fitContinuous(phy = eel.tree, dat = eel_data, model = "EB")
# Fit OU model
OU_fit <- geiger::fitContinuous(phy = eel.tree, dat = eel_data, model = "OU")

# Store models
list_model_fits <- list(BM = BM_fit, EB = EB_fit, OU = OU_fit)

# Compare models
model_comparison_output <- select_best_trait_model_from_geiger(list_model_fits = list_model_fits)

# Explore output
str(model_comparison_output, max.level = 2)

# Print comparison
print(model_comparison_output$models_comparison_df)

# Print best model fit
print(model_comparison_output$best_model_fit)

# ----- Example 2: Categorical data ----- #

# Load phylogeny and tip data
library(phytools)
data(eel.tree)
data(eel.data)

# Extract feeding mode data
eel_data <- stats::setNames(eel.data$feed_mode, rownames(eel.data))
table(eel_data)

 # (May take several minutes to run)
# Fit ER model
ER_fit <- geiger::fitDiscrete(phy = eel.tree, dat = eel_data, model = "ER")
# Fit ARD model
ARD_fit <- geiger::fitDiscrete(phy = eel.tree, dat = eel_data, model = "ARD")

# Store models
list_model_fits <- list(ER = ER_fit, ARD = ARD_fit)

# Compare models
model_comparison_output <- select_best_trait_model_from_geiger(list_model_fits = list_model_fits)

# Explore output
str(model_comparison_output, max.level = 2)

# Print comparison
print(model_comparison_output$models_comparison_df)

# Print best model fit
print(model_comparison_output$best_model_fit)
```

subset_BAMM_object *Subset a BAMM object before a deepSTRAPP run*

Description

Subset a BAMM object of class bammdata while keeping additional information for deepSTRAPP.

The BAMM_object is typically generated directly with prepare_diversification_data() or from external BAMM output files with build_BAMM_object().

This is a wrapper of the original BAMMtools::subsetEventData() function that additionally preserves information on:

- the Marginal Shift Probability (MSP) = the probability of a regime shift to occur along each branch.
- the Maximum A Posteriori probability (MAP) configurations among the posterior samples = the configurations of regime shifts that were sampled most frequently (See BAMMtools::getBestShiftConfiguration
- the Maximum Shift Credibility (MSC) configurations among the posterior samples = the configurations of regime shift location with the highest product of marginal probabilities across branches (See BAMMtools::maximumShiftCredibility()) that are stored in a BAMM_object when built with deepSTRAPP functions.

Those additional elements are used by plot_BAMM_rates() to display regime shift probabilities and locations.

Usage

```
subset_BAMM_object(
  BAMM_object,
  nb_posterior_samples = NULL,
  sample_indices = NULL,
  seed = NULL
)
```

Arguments

BAMM_object	Object of class "bammdata", typically generated with prepare_diversification_data() or build_BAMM_object(), that contains a phylogenetic tree and associated diversification rate mapping across selected posterior samples.
nb_posterior_samples	Integer. Number of posterior samples to extract from BAMM_object. Default = NULL.
sample_indices	Integer or vector of integers. Indices of the posterior samples to extract. Default = NULL.
seed	Integer. Set the seed to ensure reproducibility when sample_indices is not provided and posterior samples are drawn randomly. Default is NULL (a random seed is used).

Value

The function returns a `BAMM_object` of class `"bammdata"` which is a list with at least 23 elements.

Phylogeny-related elements used to plot a phylogeny with `ape::plot.phylo()`:

- `$edge` Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- `$Nnode` Integer. Number of internal nodes.
- `$tip.label` Character vector. Labels of all tips.
- `$edge.length` Numeric vector. Length of edges/branches.
- `$node.label` Character vector. Labels of all internal nodes. (Present only if present in the initial `phylo`)

BAMM internal elements used for tree exploration:

- `$begin` Numeric vector. Absolute time since root of edge/branch start (rootward).
- `$end` Numeric vector. Absolute time since root of edge/branch end (tipward).
- `$downseq` Integer vector. Order of node visits when using a pre-order tree traversal.
- `$lastvisit` ID of the last node visited when starting from the node in the corresponding position in `$downseq`.

BAMM elements summarizing diversification data:

- `$numberEvents` Integer vector. Number of events/macroevolutionary regimes (k+1) recorded in each posterior configuration. k = number of shifts.
- `$eventData` List of data.frames. One per posterior sample. Records shift events and macroevolutionary regimes parameters. 1st line = Background root regime.
- `$eventVectors` List of integer vectors. One per posterior sample. Record regime ID per branch.
- `$tipStates` List of named integer vectors. One per posterior sample. Record regime ID per tip.
- `$tipLambda` List of named numeric vectors. One per posterior sample. Record speciation rates per tip.
- `$tipMu` List of named numeric vectors. One per posterior sample. Record extinction rates per tip.
- `$eventBranchSegs` List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- `$meanTipLambda` Named numeric vector. Mean tip speciation rates across all posterior configurations of tips.
- `$meanTipMu` Named numeric vector. Mean tip extinction rates across all posterior configurations of tips.
- `$type` Character string. Set the type of data modeled with BAMM. Should be "diversification".

Additional elements providing key information for downstream analyses:

- `$expectedNumberOfShifts` Integer. The expected number of regime shifts used to set the prior in BAMM.

- `$MSP_tree` Object of class `phylo`. List of 4 elements duplicating information from the Phylogeny-related elements above, except `$MSP_tree$edge.length` is recording the Marginal Shift Probability of each branch (i.e., the probability of a regime shift to occur along each branch)
- `$MAP_indices` Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- `$MAP_BAMM_object` List of 18 elements of class `"bammdata"` recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.
- `$MSC_indices` Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.
- `$MSC_BAMM_object` List of 18 elements of class `"bammdata"` recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.

Author(s)

Maël Doré

References

For BAMM: Rabosky, D. L. (2014). Automatic detection of key innovations, rate shifts, and diversity-dependence on phylogenetic trees. PloS one, 9(2), e89543. doi:10.1371/journal.pone.0089543. Website: http://bamm-project.org/.

For {BAMMtools}: Rabosky, D. L., Grundler, M., Anderson, C., Title, P., Shi, J. J., Brown, J. W., ... & Larson, J. G. (2014). BAMM tools: an R package for the analysis of evolutionary dynamics on phylogenetic trees. Methods in Ecology and Evolution, 5(7), 701-707. doi:10.1111/2041210X.12199

See Also

Initial function in BAMMtools: `BAMMtools::subsetEventData()`

Associated functions in deepSTRAPP: `prepare_diversification_data()` `build_BAMM_object()`

For a guided tutorial, see this vignette: `vignette("model_diversification_dynamics", package = "deepSTRAPP")`

Examples

```
if (deepSTRAPP::is_dev_version())
{
 ## Load BAMM object
 # data(Ponerinae_BAMM_object_old_calib)
 # This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 # Check structure of BAMM_object
 str(Ponerinae_BAMM_object_old_calib, 1)
```

```
 # Check current number of BAMM posterior samples
 length(Ponerinae_BAMM_object_old_calib$eventData)
 # We have initially 1000 posterior samples in the updated BAMM object

 ## Subset BAMM_object
 BAMM_object_subset <- subset_BAMM_object(
    BAMM_object = Ponerinae_BAMM_object_old_calib,
    nb_posterior_samples = 100,
    seed = 1234)

 # Check structure of the updated BAMM_object
 str(BAMM_object_subset, 1)
 # Check updated number of BAMM posterior samples
 length(BAMM_object_subset$eventData)
 # We have now 100 posterior samples in the updated BAMM object
}
```

`update_rates_and_regimes_for_focal_time` *Update diversification rates/regimes mapped on a phylogeny up to a given time in the past*

Description

Updates an object of class "bammdata" to obtain the diversification rates/regimes found along branches at a specific time in the past (i.e. the `focal_time`). Optionally, the function can update the object to display a mapped phylogeny such that branches overlapping the `focal_time` are shortened to the `focal_time`.

Usage

```
update_rates_and_regimes_for_focal_time(
  BAMM_object,
  focal_time,
  update_rates = TRUE,
  update_regimes = TRUE,
  update_tree = FALSE,
  update_plot = FALSE,
  update_all_elements = FALSE,
  keep_tip_labels = TRUE,
  verbose = TRUE
)
```

Arguments

`BAMM_object`	Object of class "bammdata", typically generated with `prepare_diversification_data()`, that contains a phylogenetic tree and associated diversification rate mapping

across selected posterior samples. The phylogenetic tree must be rooted and fully resolved/dichotomous, but it does not need to be ultrametric (it can include fossils).

`focal_time` Numeric. The time, in terms of time distance from the present, at which the tree and rate mapping must be cut. It must be smaller than the root age of the phylogeny.

`update_rates` Logical. Specify whether diversification rates stored in `$tipLambda` (speciation) and `$tipMu` (extinction) must be updated to summarize rates found along branches at `focal_time`. Default is `TRUE`.

`update_regimes` Logical. Specify whether diversification regimes stored in `$tipStates` must be updated to summarize regimes found along branches at `focal_time`. Default is `TRUE`.

`update_tree` Logical. Specify whether to update the phylogeny such that all branches that are younger than the `focal_time` are cut-off. Default is `FALSE`.

`update_plot` Logical. Specify whether to update the phylogeny AND the elements used by `plot_BAMM_rates()` to plot diversification rates on the phylogeny such that all branches that are younger than the `focal_time` are cut-off. Default is `FALSE`. If set to `TRUE`, it will override the `update_tree` parameter and update the phylogeny.

`update_all_elements`
Logical. Specify whether to update all the elements in the object, including rates/regimes/phylogeny/elements for `plot_BAMM_rates()`/all other elements. Default is `FALSE`. If set to `TRUE`, it will override other `update_*` parameters and update all elements.

`keep_tip_labels`
Logical. Should terminal branches with a single descendant tip retain their initial `tip.label` on the updated phylogeny and diversification rate mapping? Default is `TRUE`. If set to `FALSE`, the tipward node ID will be used as label for all tips.

`verbose` Logical. Should progression be displayed? A message will be printed for every batch of 100 BAMM posterior samples updated. Default is `TRUE`.

Details

The object of class `"bammdata"` (`BAMM_object`) is cut-off at a specific time in the past (i.e. the `focal_time`) and the current diversification rate values of the overlapping edges/branches are extracted.

—— Update diversification rate data ——

If `update_rates = TRUE`, diversification rates of branches overlapping with `focal_time` will be updated. Each cut-off branch forms a new tip present at `focal_time` with updated associated diversification rate data. Fossils older than `focal_time` do not yield any data.

- `$tipLambda` contains speciation rates.
- `$tipMu` contains extinction rates.

If `update_regimes = TRUE`, diversification regimes of branches overlapping with `focal_time` will be updated. Each cut-off branch forms a new tip present at `focal_time` with updated associated

diversification regime ID found in `$tipStates`. Fossils older than `focal_time` do not yield any data.

—— Update the phylogeny ——

If `update_tree = TRUE`, elements defining the phylogeny, as in a `"phylo"` object will be updated such that all branches that are younger than the `focal_time` are cut-off:

- `$edge` defines the tree topology.
- `$Nnode` defines the number of internal nodes.
- `$tip.label` provides the labels of all tips, including fossils older than `focal_time` if present.
- `$edge.length` provides length of all branches.
- `$node.label` provides the labels of all internal nodes. (Optional)

—— Update the plot from `plot_BAMM_rates()` ——

If `update_plot = TRUE`, elements used to plot diversification rates on the phylogeny using `plot_BAMM_rates()` will be updated such that all branches that are younger than the `focal_time` are cut-off:

- `$begin` provides absolute time since root of edge/branch start (rootward).
- `$end` provides absolute time since root of edge/branch end (tipward).
- `$eventVectors` provides regime membership per branch in each posterior sample configuration.
- `$eventBranchSegs` provides regime membership per segment of branches in each posterior sample configuration.
- `$dtrates` provides mean speciation and extinction rates along segments of branches, and resolution fraction (tau) describing the fraction of each segment length compared to the full depth of the initial tree (i.e., the root_age).

Value

The function returns a list as an updated `BAMM_object` of class `"bammdata"`.

Phylogeny-related elements used to plot a phylogeny with `ape::plot.phylo()`:

- `$edge` Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- `$Nnode` Integer. Number of internal nodes.
- `$tip.label` Character vector. Labels of all tips, including fossils older than `focal_time` if present.
  - If `keep_tip_labels = TRUE`, cut-off branches with a single descendant tip retain their initial `tip.label`.
  - If `keep_tip_labels = FALSE`, all cut-off branches are labeled using their tipward node ID.
- `$edge.length` Numeric vector. Length of edges/branches.
- `$node.label` Character vector. Labels of all internal nodes. (Present only if present in the initial `BAMM_object`)

BAMM internal elements used for tree exploration:

- `$begin` Numeric vector. Absolute time since root of edge/branch start (rootward).
- `$end` Numeric vector. Absolute time since root of edge/branch end (tipward).

- $downseq Integer vector. Order of node visits when using a pre-order tree traversal.
- $lastvisit ID of the last node visited when starting from the node in the corresponding position in $downseq.

BAMM elements summarizing diversification data:

- $numberEvents Integer vector. Number of events/macroevolutionary regimes (k+1) recorded in each posterior configuration. k = number of shifts.
- $eventData List of data.frames. One per posterior sample. Records shift events and macroevolutionary regimes parameters. 1st line = Background root regime.
- $eventVectors List of integer vectors. One per posterior sample. Record regime ID per branch.
- $tipStates List of named integer vectors. One per posterior sample. Record regime ID per tip present at focal_time. Updated if update_regimes = TRUE.
- $tipLambda List of named numeric vectors. One per posterior sample. Record speciation rates per tip present at focal_time. Updated if update_rates = TRUE.
- $tipMu List of named numeric vectors. One per posterior sample. Record extinction rates per tip present at focal_time. Updated if update_rates = TRUE.
- $eventBranchSegs List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- $meanTipLambda Named numeric vector. Mean tip speciation rates across all posterior configurations of tips present at focal_time (does not include older fossils).
- $meanTipMu Named numeric vector. Mean tip extinction rates across all posterior configurations of tips present at focal_time (does not include older fossils).
- $type Character string. Set the type of data modeled with BAMM. Should be "diversification".

Additional elements providing key information for downstream analyses:

- $expectedNumberOfShifts Integer. The expected number of regime shifts used to set the prior in BAMM.
- $MSP_tree Object of class phylo. List of 4 elements duplicating information from the Phylogeny-related elements above, except MSP_treeedge.length is recording the Marginal Shift Probability of each branch (i.e., the probability of a regime shift to occur along each branch) whose origin is older than focal_time.
- $MAP_indices Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- $MAP_BAMM_object. List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configuration. All BAMM elements summarizing diversification data hold a single entry describing this the mean diversification history, updated for the focal_time.
- $MSC_indices Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.
- $MSC_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history, updated for the focal_time.

New elements added to provide update information:

- $root_age Integer. Stores the age of the root of the tree.
- $nodes_ID_df Data.frame with two columns. Provides the conversion from the new_node_ID to the initial_node_ID. Each row is a node.
- $initial_nodes_ID Character vector. Provides the initial ID of internal nodes. Used to plot internal node IDs as labels with ape::nodelabels().
- $edges_ID_df Data.frame with two columns. Provides the conversion from the new_edge_ID to the initial_edge_ID. Each row is an edge/branch.
- $initial_edges_ID Character vector. Provides the initial ID of edges/branches. Used to plot edge/branch IDs as labels with ape::edgelabels().
- $dtrates List of three elements.
  - 1/ $dtrates$tau Numeric. Resolution factor describing the fraction of each segment length used in plot_BAMM_rates() compared to the full depth of the initial tree (i.e., the root_age)
  - 2/ $dtrates$rates List of two numeric vectors. Speciation and extinction rates along segments used by plot_BAMM_rates().
  - 3/ $dtrates$tmat Numeric matrix. Start and end times of segments in term of distance to the root.
- $initial_colorbreaks List of three numeric vectors. Rate values of the percentiles delimiting the bins for mapping rates to colors with BAMMtools::plot.bammdata(). Each element provides values for different type of rates ($speciation, $extinction, $net_diversification).
- $focal_time Integer. The time, in terms of time distance from the present, at which the rates/regimes were extracted and the tree was possibly cut.

Author(s)

Maël Doré

See Also

cut_phylo_for_focal_time() plot_BAMM_rates()

Examples

```
# ----- Example 1: Extant whales (87 taxa) ----- #

## Load the BAMM_object summarizing 1000 posterior samples of BAMM
data(whale_BAMM_object, package = "deepSTRAPP")

## Set focal-time to 5 My
focal_time = 5

 # (May take several minutes to run)
## Update the BAMM object
whale_BAMM_object_5My <- update_rates_and_regimes_for_focal_time(
   BAMM_object = whale_BAMM_object,
   focal_time = 5,
   update_rates = TRUE, update_regimes = TRUE,
   update_tree = TRUE, update_plot = TRUE,
   update_all_elements = TRUE,
```

```
   keep_tip_labels = TRUE,
   verbose = TRUE)

# Add "phylo" class to be compatible with phytools::nodeHeights()
class(whale_BAMM_object) <- unique(c(class(whale_BAMM_object), "phylo"))
root_age <- max(phytools::nodeHeights(whale_BAMM_object)[,2])
# Remove temporary "phylo" class
class(whale_BAMM_object) <- setdiff(class(whale_BAMM_object), "phylo")

## Plot initial BAMM_object for t = 0 My
plot_BAMM_rates(whale_BAMM_object, add_regime_shifts = TRUE,
                labels = TRUE, legend = TRUE, cex = 0.5)
abline(v = root_age - focal_time,
      col = "red", lty = 2, lwd = 2)

## Plot updated BAMM_object for t = 5 My
plot_BAMM_rates(whale_BAMM_object_5My, add_regime_shifts = TRUE,
                labels = TRUE, legend = TRUE, cex = 0.8)

# ----- Example 2: Extant Ponerinae (1,534 taxa) ----- #

if (deepSTRAPP::is_dev_version())
{
 ## Load the BAMM_object summarizing 1000 posterior samples of BAMM
 data(Ponerinae_BAMM_object, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Set focal-time to 10 My
 focal_time = 10

  # (May take several minutes to run)
 ## Update the BAMM object
 Ponerinae_BAMM_object_10My <- update_rates_and_regimes_for_focal_time(
    BAMM_object = Ponerinae_BAMM_object,
    focal_time = focal_time,
    update_rates = TRUE, update_regimes = TRUE,
    update_tree = TRUE, update_plot = TRUE,
    update_all_elements = TRUE,
    keep_tip_labels = TRUE,
    verbose = TRUE)

 ## Load results to save time
 data(Ponerinae_BAMM_object_10My, package = "deepSTRAPP")
 ## This dataset is only available in development versions installed from GitHub.
 # It is not available in CRAN versions.
 # Use remotes::install_github(repo = "MaelDore/deepSTRAPP") to get the latest development version.

 ## Extract root age
 # Add temporarily the "phylo" class to be compatible with phytools::nodeHeights()
 class(Ponerinae_BAMM_object) <- unique(c(class(Ponerinae_BAMM_object), "phylo"))
 root_age <- max(phytools::nodeHeights(Ponerinae_BAMM_object)[,2])
```

```
# Remove temporary "phylo" class
class(Ponerinae_BAMM_object) <- setdiff(class(Ponerinae_BAMM_object), "phylo")

## Plot diversification rates on the initial tree
plot_BAMM_rates(Ponerinae_BAMM_object,
                legend = TRUE, labels = FALSE)
abline(v = root_age - focal_time,
       col = "red", lty = 2, lwd = 2)

## Plot diversification rates and regime shifts on the updated tree (cut-off for 10 My)
# Keep the initial color scheme
plot_BAMM_rates(Ponerinae_BAMM_object_10My, legend = TRUE, labels = FALSE,
          colorbreaks = Ponerinae_BAMM_object_10My$initial_colorbreaks$net_diversification)

# Use a new color scheme mapped on the new distribution of rates
plot_BAMM_rates(Ponerinae_BAMM_object_10My, legend = TRUE, labels = FALSE)
}
```

`whale_BAMM_object` *Dataset summarizing 1000 posterior samples of BAMM for extant whales*

Description

An object of class `"bammdata"` containing information on diversification dynamics of extant whales (Cetacea order) modeled with BAMM.

Source: Steeman, M. E., M. B. Hebsgaard, R. E. Fordyce, S. Y. W. Ho, D. L. Rabosky, R. Nielsen, C. Rahbek, H. Glenner, M. V. Sorensen, and E. Willerslev (2009) Radiation of extant cetaceans driven by restructuring of the oceans. Systematic Biology, 58, 573-585.

Usage

```
data(whale_BAMM_object)
```

Format

A list with 24 elements.

Details

An object of class `"bammdata"` containing information on diversification dynamics of extant whales (Cetacea order) modeled with BAMM.

Phylogeny-related elements used to plot a phylogeny with `ape::plot.phylo()`:

- `$edge` Integer matrix. Defines the tree topology by providing rootward and tipward node ID of each edge.
- `$Nnode` Integer. Number of internal nodes.

- $tip.label Character vector. Labels of all tips.
- $edge.length Numeric vector. Length of edges/branches.

BAMM internal elements used for tree exploration:

- $begin Numeric vector. Absolute time since root of edge/branch start (rootward).
- $end Numeric vector. Absolute time since root of edge/branch end (tipward).
- $downseq Integer vector. Order of node visits when using a pre-order tree traversal.
- $lastvisit ID of the last node visited when starting from the node in the corresponding position in $downseq.

BAMM elements summarizing diversification data:

- $numberEvents Integer vector. Number of events/macroevolutionary regimes (k+1) recorded in each posterior configuration. k = number of shifts.
- $eventData List of data.frames. One per posterior sample. Records shift events and macroevolutionary regimes parameters. 1st line = Background root regime.
- $eventVectors List of integer vectors. One per posterior sample. Record regime ID per branch.
- $tipStates List of integer vectors. One per posterior sample. Record regime ID per tip.
- $tipLambda List of numeric vectors. One per posterior sample. Record speciation rates per tip.
- $tipMu List of numeric vectors. One per posterior sample. Record extinction rates per tip.
- $eventBranchSegs List of numeric matrices. One per posterior sample. Record regime ID per segment of branches.
- $meanTipLambda Numeric vector. Mean tip speciation rates across all posterior configurations of tips.
- $meanTipMu Numeric vector. Mean tip extinction rates across all posterior configurations of tips.
- $type Character string. Set the type of data modeled with BAMM. Here, type = "diversification".

Additional elements providing key information for downstream analyses:

- $expectedNumberOfShifts Integer. The expected number of regime shifts used to set the prior in BAMM.
- $MSP_tree Object of class phylo. List of 4 elements duplicating information from the Phylogeny-related elements above, except MSP_treeedge.length is recording the Marginal Shift Probability of each branch (i.e., the probability of a regime shift to occur along each branch)
- $MAP_indices Integer vector. The indices of the Maximum A Posteriori probability (MAP) configurations among the posterior samples.
- $MAP_BAMM_object List of 18 elements of class "bammdata" recording the mean rates and regime shift locations found across the Maximum A Posteriori probability (MAP) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.
- $MSC_indices Integer vector. The indices of the Maximum Shift Credibility (MSC) configurations among the posterior samples.

- `$MSC_BAMM_object` List of 18 elements of class `"bammdata"` recording the mean rates and regime shift locations found across the Maximum Shift Credibility (MSC) configurations. All BAMM elements summarizing diversification data hold a single entry describing this mean diversification history.

References

Steeman, M. E., M. B. Hebsgaard, R. E. Fordyce, S. Y. W. Ho, D. L. Rabosky, R. Nielsen, C. Rahbek, H. Glenner, M. V. Sorensen, and E. Willerslev (2009) Radiation of extant cetaceans driven by restructuring of the oceans. Systematic Biology, 58, 573-585.

See Also

BAMM software website: http://bamm-project.org/

Index

∗ **datasets**
BAMM_template_diversification, 6
eel_biogeo_data, 49
eel_cat_3lvl_data, 50
mammals, 68
Ponerinae_binary_range_table, 124
Ponerinae_biogeo_data_old_calib, 125
Ponerinae_cat_2lvl_data_old_calib, 126
Ponerinae_cat_3lvl_data_old_calib, 127
Ponerinae_trait_cont_tip_data_10My, 128
Ponerinae_trait_tip_data, 129
Ponerinae_tree, 129
Ponerinae_tree_old_calib, 130
whale_BAMM_object, 201

aggregate_contMaps, 3
ape, *28*
ape::ace(), *61*, *158*, *171*
ape::edgelabels(), *30*, *33*, *36*, *39*, *41*, *44*, *47*, *54, 55*, *63*, *199*
ape::nodelabels(), *30*, *33*, *36*, *38*, *41*, *44*, *47*, *54, 55*, *63*, *199*
ape::plot.phylo(), *8*, *135*, *151*, *193*, *197*, *201*

BAMM_template_diversification, 6
BAMMtools::addBAMMshifts(), *69, 70*, *72*, *112*, *119*
BAMMtools::credibleShiftSet(), *71*
BAMMtools::distinctShiftConfigurations(), *135*
BAMMtools::getBestShiftConfiguration(), *7, 8*, *10*, *70*, *111*, *119*, *133*, *135*, *149–151*, *192*
BAMMtools::getEventData, *135*
BAMMtools::getEventData(), *7*, *10*
BAMMtools::marginalShiftProbsTree(), *135*
BAMMtools::maximumShiftCredibility(), *7*, *10*, *70*, *71*, *111*, *119*, *135*, *149*, *151*, *192*
BAMMtools::plot.bammdata(), *69–72*, *112*, *119*, *199*
BAMMtools::plotPrior, *134*
BAMMtools::plotPrior(), *133*
BAMMtools::samplingProbs(), *133*
BAMMtools::setBAMMpriors, *134*
BAMMtools::subsetEventData, *135*
BAMMtools::subsetEventData(), *192*, *194*
BAMMtools::subtreeBAMM(), *149, 150*, *153*
BAMMtools::traitDependentBAMM(), *11*, *14*, *17*, *156*, *170*
build_BAMM_object, 7
build_BAMM_object(), *149, 150*, *153*, *192*, *194*

coda::effectiveSize(), *134*
compute_STRAPP_test_for_focal_time, 11
compute_STRAPP_test_for_focal_time(), *83*, *161–163*, *177*
contsimmap::make.contsimmap(), *4*, *24, 25*, *143*
convert_BSM_to_simmap, 20
convert_BSM_to_simmap(), *20–22*
convert_BSMs_to_simmaps *(*convert_BSM_to_simmap*)*, 20
convert_BSMs_to_simmaps(), *20*, *22*
convert_contsimmap_to_contMaps, 24
convert_contsimmap_to_contMaps(), *4*
convert_simmaps_to_densityMaps, 26
cut_contMap_for_focal_time, 32
cut_contMap_for_focal_time(), *41*, *63*
cut_contMaps_for_focal_time, 29
cut_contMaps_for_focal_time(), *55*
cut_densityMap_for_focal_time, 37
cut_densityMap_for_focal_time(), *36*

cut_densityMaps_for_focal_time, 34
cut_densityMaps_for_focal_time(), *41*, *55*, *63*
cut_phylo_for_focal_time, 40
cut_phylo_for_focal_time(), *30*, *33*, *36*, *39*, *44*, *47*, *55*, *63*, *199*
cut_simmap_for_focal_time, 46
cut_simmaps_for_focal_time, 42
cut_simmaps_for_focal_time(), *55*

dunn.test::dunn.test, *11*, *162*
dunn.test::dunn.test(), *17*

eel_biogeo_data, 49
eel_cat_3lvl_data, 50
extract_all_trait_values_for_focal_time, 51
extract_all_trait_values_for_focal_time(), *12*, *30*, *36*, *39*, *44*, *47*, *62, 63*, *66*, *161*, *163*
extract_diversification_data_melted_df_for_focal_time, 58
extract_diversification_data_melted_df_for_focal_time(), *161–163*, *177*
extract_most_likely_trait_values_for_focal_time, 59
extract_most_likely_trait_values_for_focal_time(), *12*, *17*, *30*, *33*, *36*, *39*, *44*, *47*, *53*, *55*, *66*, *128*, *161*, *163*, *177*
extract_trait_data_melted_df_for_focal_time, 66
extract_trait_data_melted_df_for_focal_time(), *161–163*

geiger::fitContinuous, *128*
geiger::fitContinuous(), *140*, *142*, *145*, *190*
geiger::fitDiscrete, *50*, *126, 127*
geiger::fitDiscrete(), *140*, *142*, *145*, *190*
graphics::abline(), *71*
graphics::title(), *71*
grDevices::colorRampPalette(), *4*, *73*, *90*, *95*, *101*, *111*, *118*, *141*

mammals, 68

OUwie::OUwie(), *145*

par(), *70*, *112*, *119*
phytools::anc.ML(), *61*, *158*, *171*
phytools::brownie.lite(), *145*
phytools::contMap(), *3, 4*, *24, 25*, *29*, *32*, *60*, *143*, *145*, *157*, *171*
phytools::countSimmap(), *22, 23*
phytools::densityMap(), *27, 28*, *35*, *38*, *143*, *145*, *157*
phytools::fastAnc, *143*
phytools::fastAnc(), *61*, *158*, *171*
phytools::make.simmap, *51*
phytools::make.simmap(), *20*, *23*, *27*, *43*, *46*, *52*, *60*, *141*, *157*, *171*
phytools::plot.contMap(), *73, 74*, *112*, *119*
phytools::plot.densityMap(), *76*, *113*, *120*
phytools::plot.simmap(), *119*
phytools::plotSimmap(), *75, 76*, *112*
phytools::rescale(), *143*
phytools::setMap(), *73*
plot_BAMM_rates, 69
plot_BAMM_rates(), *7*, *10*, *110*, *112, 113*, *119, 120*, *131*, *135*, *137*, *151*, *153*, *162*, *175*, *177*, *192*, *196, 197*, *199*
plot_contMap, 73
plot_contMap(), *4*, *110*, *120*, *162*, *175*, *177*
plot_densityMaps_overlay, 75
plot_densityMaps_overlay(), *27, 28*, *74*, *110*, *112, 113*, *119, 120*, *162*, *175*, *177*
plot_histogram_STRAPP_test_for_focal_time, 83
plot_histogram_STRAPP_test_for_focal_time(), *14–16*, *79*, *160*, *174, 175*, *177*
plot_histograms_STRAPP_tests_over_time, 77
plot_histograms_STRAPP_tests_over_time(), *84, 85*, *174, 175*, *177*
plot_rates_through_time, 89
plot_rates_through_time(), *175, 176*
plot_rates_vs_trait_data_for_focal_time, 94
plot_rates_vs_trait_data_for_focal_time(), *102, 103*
plot_rates_vs_trait_data_over_time, 100
plot_rates_vs_trait_data_over_time(), *96, 97*
plot_STRAPP_pvalues_over_time, 107

plot_STRAPP_pvalues_over_time(), *15*, *175*, *176*
plot_traits_vs_rates_on_phylogeny_for_focal_time, 109
plot_traits_vs_rates_on_phylogeny_for_focal_time(), *120*
plot_traits_vs_rates_on_phylogeny_over_time, 117
plot_traits_vs_rates_on_phylogeny_over_time(), *113*
Ponerinae_binary_range_table, 124
Ponerinae_biogeo_data_old_calib, 125
Ponerinae_cat_2lvl_data_old_calib, 126
Ponerinae_cat_3lvl_data_old_calib, 127
Ponerinae_trait_cont_tip_data_10My, 128
Ponerinae_trait_tip_data, 129
Ponerinae_tree, 129
Ponerinae_tree_old_calib, 130
prepare_diversification_data, 131
prepare_diversification_data(), *6*, *7*, *10*, *69*, *71*, *72*, *149*, *150*, *153*, *158*, *172*, *192*, *194*, *195*
prepare_trait_data, 139
prepare_trait_data(), *25*, *27*, *28*, *35*, *49–52*, *60*, *73*, *75*, *125–128*, *137*, *157*, *171*
prune_BAMM_object, 149

RColorBrewer::brewer.pal(), *90*, *95*, *101*
run_deepSTRAPP_for_focal_time, *69*, *137*, *145*, 156
run_deepSTRAPP_for_focal_time(), *7*, *10*, *27*, *72*, *83–85*, *94–97*, *108*, *111–113*, *117*, *131*, *135*, *137*, *175*, *177*
run_deepSTRAPP_over_time, *137*, *145*, 169
run_deepSTRAPP_over_time(), *7*, *10*, *14*, *27*, *69*, *72*, *77–79*, *84*, *89–91*, *94–96*, *101–103*, *107*, *108*, *111*, *113*, *117*, *118*, *120*, *131*, *135*, *137*, *160*

scales::hue_pal(), *90*, *95*, *102*
select_best_model_from_BioGeoBEARS, 186
select_best_trait_model_from_geiger, 189
stats::cor.test, *11*, *162*
stats::cor.test(), *17*
stats::kruskal.test, *11*, *162*
stats::kruskal.test(), *17*
stats::p.adjust(), *14*, *160*, *174*
stats::wilcox.test, *11*, *162*
stats::wilcox.test(), *17*
subset_BAMM_object, 192
subset_BAMM_object(), *153*

update_rates_and_regimes_for_focal_time, 195
update_rates_and_regimes_for_focal_time(), *12*, *17*, *58*, *59*, *67*, *69*, *72*, *135*, *137*, *161–163*, *177*

whale_BAMM_object, 201