Package: ontophylo (via r-universe)

August 24, 2024

Title Ontology-Informed Phylogenetic Comparative Analyses

Version 1.0.0

Description This package provides new phylogenetic tools for analyzing discrete trait data integrating bio-ontologies and phylogenetics. It expands on the previous work of Tarasov et al. (2019). The PARAMO pipeline allows to reconstruct ancestral phenomes treating groups of morphological traits as a single complex character. The pipeline incorporates knowledge from ontologies during the amalgamation of individual character stochastic maps. Here we expand the current PARAMO functionality by adding new statistical methods for inferring evolutionary phenome dynamics using non-homogeneous Poisson process (NHPP). The new functionalities include: (1) reconstruction of evolutionary rate shifts of phenomes across lineages and time; (2) reconstruction of morphospace dynamics through time; and (3) estimation of rates of phenome evolution at different levels of anatomical hierarchy (e.g., entire body or specific regions only). The package also includes user-friendly tools for visualizing evolutionary rates of different anatomical regions using PNG vector images of the organisms of interest.

License MIT + file LICENSE

Encoding UTF-8

Language en-US

URL `https://github.com/diegosasso/ontophylo`

BugReports `https://github.com/diegosasso/ontophylo/issues`

Depends R (>= 3.5.0)

Imports magrittr, dplyr, tidyr, tibble, ggplot2, stringdist, phytools, ontologyIndex, RColorBrewer, grid, truncnorm, fANCOVA

Suggests grImport, rmarkdown, knitr, roxygen2, testthat (>= 3.0.0)

VignetteBuilder knitr

Remotes diegosasso/ontophylo

LazyData true

RoxygenNote 7.2.3

Config/testthat/edition 3

Repository https://diegosasso.r-universe.dev

RemoteUrl https://github.com/diegosasso/ontophylo

RemoteRef HEAD

RemoteSha 49565f518d1057473fe3470d8bc50b67030e0d49

Contents

`add_noise_MD` *Adding noise to MDS from one stochastic character map*

Description

Adds noise to the points in the 2D coordinates in the MDS plot. # The noise is calculated as var(V)*add.noise.

Usage

```
add_noise_MD(MD, add.noise)
```

Arguments

`MD`	tibble. The output of a MD_morpho function.
`add.noise`	numeric. A vector of length 2 indicating the amount of noise to be added to the x and y coordinates.

Value

A list of tibbles as in the output of MD_morpho functions, with noise added.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_mds")
# Get a sample of 5 amalgamated stochastic maps.
tree_list <- hym_stm_mds
tree_list <- merge_tree_cat_list(tree_list)
## Not run:

  # Multidimensional scaling for an arbitrary tree.
  MD <- suppressWarnings(MultiScale.simmap(tree_list[[1]], add.noise = NULL))

  # Add noise.
  add_noise_MD(MD, c(0.3, 0.3))


## End(Not run)
```

`add_pseudodata` *Add pseudodata*

Description

Adds a vector of pseudodata to the path data obtained from the 'make_data_NHPP_KDE_Markov_kernel' function.

Usage

```
add_pseudodata(Edge.groups, Pseudo.data, Path.data)
```

Arguments

`Edge.groups`	list. A list with groups of edge IDs.
`Pseudo.data`	numeric. A vector with values of pdeusodata.
`Path.data`	numeric. A list of path data obtained from the 'make_data_NHPP_KDE_Markov_kernel' function.

Value

A list of path data with the pseudodata added.

Author(s)

Sergei Tarasov

Examples

```
data("hym_hm", "hym_tree")
# Get hamming data from the head characters.
hm <- hym_hm$head
# Make NHPP path data.
nhpp <- make_data_NHPP_KDE_Markov_kernel(hm, add.psd = FALSE)
# Add pseudo data to path data.
psd <- lapply(nhpp, function(x) -x[x < 100] )
edge_groups <- as.list(1:length(hym_tree$edge.length))
nhpp_psd <- add_pseudodata(Edge.groups = edge_groups, Pseudo.data = psd, Path.data = nhpp)
# Check NHPP path data plus pseudodata for an arbitrary branch.
nhpp_psd[[5]]
```

`anat_plot` *Plot Picture*

Description

Wrapper function for making a plot of an object of class 'Picture' using the 'make_pic' function.

Usage

```
anat_plot(picture, anat_layers, plot_stat, color_palette, scale_lim)
```

Arguments

`picture`	grImport object. A vector image imported in R using the 'readPicture' function from grImport.
`anat_layers`	numeric. A named vector with the layer IDs obtained using the 'get_vector_ids_list' function.
`plot_stat`	numeric. A named vector with values corresponding to the branch statistics to be plotted and names matching the names in the layer IDs.
`color_palette`	A character vector or function defining a color palette.
`scale_lim`	numeric. A pair of values defining the lower and upper limits of the scale.

Value

A plot of the object of class 'Picture' with the assigned colors to different anatomical regions.

Author(s)

Diego Porto

Examples

```
data("HAO", "hym_graph", "hym_img", "hym_kde")
# Get picture.
picture <- hym_img
# Get picture layers from three anatomical regions.
terms_list <- as.list(c("HAO:0000397", "HAO:0000576", "HAO:0000626"))
terms_list <- setNames(terms_list, c("head", "mesosoma", "metasoma"))
anat_layers <- get_vector_ids_list(terms = terms_list , ONT = HAO, GR = hym_graph)
# Get mean rates all branches for the three anatomical regions.
plot_stat <- lapply(hym_kde, function(x) unlist(lapply(x$loess.lambda.mean, function(x) mean(x) )) )
plot_stat <- do.call(cbind, plot_stat)
# Select an arbitrary branch.
plot_stat <- plot_stat[50,]
# Set scale.
scale_lim <- range(plot_stat)
# Get color palette.
hm.palette <- colorRampPalette(RColorBrewer::brewer.pal(9, "Spectral"), space = "Lab")

# Plot picture.
## Not run:

  anat_plot(picture, anat_layers, plot_stat, hm.palette(100), scale_lim)


## End(Not run)
```

---

color.bar *Color bar*

Description

Function to plot the color scale bar.

Usage

```
color.bar(
  pal,
  min,
  max = -min,
  nticks = 11,
  ticks = seq(min, max, len = nticks),
  title = ""
)
```

Arguments

pal	character. A vector with color IDs.
min	numeric. Value for lower limit of the scale.

`max`	numeric. Value for upper limit of the scale.
`nticks`	integer. Number of subdivisions of the scale.
`title`	character. A legend for the scale bar.

Value

A plot of the color scale bar.

Author(s)

Sergei Tarasov

Examples

```
stat <- runif(10, 0.25, 1)
hm.palette <- colorRampPalette(RColorBrewer::brewer.pal(9, "Spectral"), space = "Lab")
## Not run:

  color.bar(hm.palette(100), min = min(stat), max = max(stat),
            ticks = round(c(min(stat), max(stat)/2, max(stat)), 2), title = "")


## End(Not run)
```

`derivative_KDE` *Calculate KDE derivative over edges*

Description

Calculates the derivative of the normalized Markov KDE or normalized loess smoothing over edges.

Usage

```
derivative_KDE(tree.discr, Edge.KDE.stat)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Edge.KDE.stat`	list. A list with the estimated normalized or loess smoothing KDEs for each edge.

Value

A list with the distribution of the derivatives calculated for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_tree", "hym_kde")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get smoothing of normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
Edge_KDE_stat <- Edge_KDE$loess.lambda.mean
# Calculate derivatives.
Edge_KDE$loess.lambda.mean.deriv <- derivative_KDE(tree_discr, Edge_KDE_stat)
# Check derivatives of some arbitrary branch.
Edge_KDE$loess.lambda.mean.deriv[[5]]
```

`discr_Simmap` *Reading unsummarized simmap for one tree*

Description

Discretizes tree edges into identical bins given a selected resolution value.

Usage

```
discr_Simmap(tree, res)
```

Arguments

`tree`	simmap or phylo object.
`res`	integer. A resolution value for the discretization of tree edges.

Value

A simmap or phylo object.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm")
tree <- hym_stm[[1]][[1]]
stm_discr <- discr_Simmap(tree, res = 100)
# Check some arbitrary branch.
tree$maps[[10]]
stm_discr$maps[[10]]
```

`discr_Simmap_all` *Reading unsummarized simmap for a list of trees*

Description

Discretizes tree edges of a list of trees.

Usage

```
discr_Simmap_all(tree, res)
```

Arguments

`tree`	multiSimmap or multiPhylo object.
`res`	integer. A resolution value for the discretization of tree edges.

Value

A multiSimmap or multiPhylo object.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm")
tree_list <- hym_stm[[1]]
stm_discr_list <- discr_Simmap_all(tree_list, res = 100)
# Check some arbitrary branch of some arbitrary tree.
tree_list[[1]]$maps[[10]]
stm_discr_list[[1]]$maps[[10]]
```

`edgeplot` *Plot edge profiles and contMap*

Description

Wrapper function for plotting edge profiles and contmap from NHPP.

Usage

```
edgeplot(map_stat, prof_stat, plot.cont = TRUE)
```

Arguments

`map_stat`	contMap object. A contMap obtained using the 'make_contMap_KDE' function.
`prof_stat`	tibble. A tibble with the edgeplot information obtained using the 'edge_profiles4plotting' function.
`plot.cont`	logical. Whether to plot also the contMap or not.

Value

A plot with the edge profiles and contMap of the selected statistic (e.g. branch rates).

Author(s)

Diego Porto

Examples

```
data("hym_tree", "hym_kde")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get smoothing of normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
Edge_KDE_stat <- Edge_KDE$loess.lambda.mean
# Make contmap nhpp data.
map_stat <- make_contMap_KDE(tree_discr, Edge_KDE_stat)
# Make edgeplot nhpp data.
prof_stat <- edge_profiles4plotting(tree_discr, Edge_KDE_stat)
# Plot.
## Not run:

  edgeplot(map_stat, prof_stat)


## End(Not run)
```

`edge_profiles4plotting` *Make edge profiles for plotting*

Description

Gets the information necessary for making an edgeplot, where the tree is plotted in a space where the x axis is the time and y axis the scale of the desired statistics.

Usage

```
edge_profiles4plotting(tree.discr, Edge.KDE.stat)
```

Arguments

`tree.discr`	phylo object. A discretized tree using the 'discr_Simmap' function.
`Edge.KDE.stat`	list. A list with the distributions of the estimated parameter of KDEs for each edge.

Value

A tibble with X and Y coordinates and other information necessary for making an edgeplot.

Author(s)

Sergei Tarasov

Examples

```
data("hym_tree", "hym_kde")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get smoothing of normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
Edge_KDE_stat <- Edge_KDE$loess.lambda.mean
# Make edgeplot nhpp data.
stat_prof <- edge_profiles4plotting(tree_discr, Edge_KDE_stat)
```

`estimate_band_W` *Estimate bandwidth*

Description

Estimate the bandwidth for the Markov KDE.

Usage

```
estimate_band_W(
  tree.discr,
  data.path,
  band.width = c("bw.nrd0", "bw.nrd0", "bw.ucv", "bw.bcv", "bw.SJ")
)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`data.path`	list. A list of path data obtained from the 'make_data_NHPP_KDE_Markov_kernel' function.
`band.width`	character. Bandwidth selectors for the KDE, as in density.

Value

A numeric vector.

Author(s)

Sergei Tarasov

Examples

```
data("hym_hm", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get hamming data from the head characters.
hm <- hym_hm$head
# Make NHPP path data.
nhpp_psd <- make_data_NHPP_KDE_Markov_kernel(hm, add.psd = TRUE)
# Calculate bandwidth.
bdw <- estimate_band_W(tree_discr, nhpp_psd, band.width = "bw.nrd0")
mean(bdw)
```

`estimate_edge_KDE` *Estimate the normalized Markov KDE*

Description

Estimated the normalized Markov KDE for each edge averaged across all possible root-tip paths.

Usage

```
estimate_edge_KDE(tree.discr, Path.data, h)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Path.data`	numeric. A list of path data obtained from the 'make_data_NHPP_KDE_Markov_kernel' function.
`h`	numeric. A value for the bandwidth calculated using the 'estimate_band_W' function.

Value

A list with the estimated unnormalized ($Maps.mean) and normalized ($Maps.mean.norm) KDEs for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_nhpp", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Make NHPP path data.
nhpp <- hym_nhpp$head
# Add pseudo data to path data.
psd <- lapply(nhpp, function(x) -x[x < 100] )
edge_groups <- as.list(1:length(hym_tree$edge.length))
nhpp_psd <- add_pseudodata(Edge.groups = edge_groups, Pseudo.data = psd, Path.data = nhpp)
# Calculate bandwidth.
bdw <- estimate_band_W(tree_discr, nhpp_psd, band.width = "bw.nrd0")
bdw <- mean(bdw)
# Estimate non-normalized and normalized edge KDE.
Edge_KDE <- estimate_edge_KDE(tree_discr, nhpp_psd, h = bdw)
# Check KDE data for normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean.norm[[5]]
# Check KDE data for non-normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean[[5]]
```

`estimate_edge_KDE_Markov_kernel_unnorm`
Estimate the unnormalized Markov KDE

Description

Estimated the unnormalized Markov KDE for each edge averaged across all possible root-tip paths.

Usage

```
estimate_edge_KDE_Markov_kernel_unnorm(tree.discr, Path.data, h = 10)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Path.data`	numeric. A list of path data obtained from the 'make_data_NHPP_KDE_Markov_kernel' function.
`h`	numeric. A value for the bandwidth calculated using the 'estimate_band_W' function.

Value

A list with the estimated unnormalized KDEs ($Maps.mean) for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_nhpp", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Make NHPP path data.
nhpp <- hym_nhpp$head
# Add pseudo data to path data.
psd <- lapply(nhpp, function(x) -x[x < 100] )
edge_groups <- as.list(1:length(hym_tree$edge.length))
nhpp_psd <- add_pseudodata(Edge.groups = edge_groups, Pseudo.data = psd, Path.data = nhpp)
# Calculate bandwidth.
bdw <- estimate_band_W(tree_discr, nhpp_psd, band.width = "bw.nrd0")
bdw <- mean(bdw)
# Estimate non-normalized and normalized edge KDE.
Edge_KDE <- estimate_edge_KDE_Markov_kernel_unnorm(tree_discr, nhpp_psd, h = bdw)
# Check KDE data for non-normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean[[5]]
```

---

get_descendants_chars *Get characters that are the descendants of a selected ontology term*

Description

Returns all characters located (associated) with a given ontology term.

Usage

```
get_descendants_chars(ontology, annotations = "auto", terms, ...)
```

Arguments

ontology	ontology_index object.
annotations	character. Sets which annotations to use: "auto" means automatic annotations, "manual" means manual annotations. Alternatively, any other list of element containing annotations can be specified.
terms	character. IDs of ontology terms for which descendants are queried.
...	other parameters for ontologyIndex::get_descendants() function.

Value

The vector of character IDs.

Author(s)

Sergei Tarasov

Examples

```
data("HAO")
HAO$terms_selected_id <- list("CH1" = c("HAO:0000653"), "CH2" = c("HAO:0000653"))
get_descendants_chars(HAO, annotations = "manual", "HAO:0000653")
```

`get_path_edges` *Get edges IDs from root to a given node.*

Description

Internal function. Not exported.

Usage

```
get_path_edges(tree.merge, node)
```

Author(s)

Sergei Tarasov

`get_rough_state_cols` *Multiple character state colors*

Description

Get state colors for ploting stochastic character maps when there many states.

Usage

```
get_rough_state_cols(tree)
```

Arguments

`tree`	simmap object.

Value

A character vector with colors associated with state names.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg")
# Get one sample of stochastic map from head.
tree <- hym_stm_amalg$head[[5]]
# Plot one amalgamated stochastic map from head.
## Not run:

  phytools::plotSimmap(tree, get_rough_state_cols(tree),  lwd = 3, pts = F,ftype = "off", ylim = c(0,100))


## End(Not run)
```

`get_states_path` *Get state information about a given path.*

Description

Internal function. Not exported.

Usage

```
get_states_path(tree.merge, node)
```

Author(s)

Sergei Tarasov

`get_vector_ids_list` *Wrapper for getting vector layer IDs for multiple terms*

Description

Given an ontology_index object, data.frame with ontology term labels, and data.frame with picture information (see examples), produces a named vector with layer IDs to be used in the 'make_pic' function.

Usage

```
get_vector_ids_list(terms_list, ONT, GR)
```

Arguments

`ONT`	ontology_index object.
`GR`	data.frame. A data.frame with the picture information. It contains the matches between all ontology term labels and layer IDs in the Picture object. The first column corresponds to the ontology term labels, the second to the ontology IDs, and the third to the layer IDs in the Picture object.
`term_list`	list. A named list with ontology terms to get layer IDs for. The first column corresponds to the ontology term labels, the second to the ontology IDs.

Value

A named vector with the layer IDs corresponding to or descending from the ontology term label queried.

Author(s)

Diego S. Porto

Examples

```
data("HAO", "hym_graph")
# Get picture layers from three anatomical regions.
terms_list <- as.list(c("HAO:0000397", "HAO:0000576", "HAO:0000626"))
terms_list <- setNames(terms_list, c("head", "mesosoma", "metasoma"))
get_vector_ids_list(terms = terms_list , ONT = HAO, GR = hym_graph)
```

`get_vector_ids_per_term` *Get vector layer IDs for single term*

Description

Given an ontology_index object, ontology term label, and data.frame with picture information (see examples), produces a named vector with layer IDs to be used in the 'make_pic' function.

Usage

```
get_vector_ids_per_term(term = "HAO:0000349", ONT, GR)
```

Arguments

`term`	character. An ontology term label to get the corresponding layer IDs in the Picture object.
`ONT`	ontology_index object.
`GR`	data.frame. A data.frame with the picture information. It contains the matches between all ontology term labels and layer IDs in the Picture object. The first column corresponds to the ontology term labels, the second to the ontology IDs, and the third to the layer IDs in the Picture object.

Value

A named vector with the layer IDs corresponding to or descending from the ontology term label queried.

Author(s)

Sergei Tarasov

Examples

```
data("HAO", "hym_graph")
# Get picture layers from head.
get_vector_ids_per_term(term = "HAO:0000397", ONT = HAO, GR = hym_graph)
```

`integrate_edge_KDE` *Calculate KDE integral over edges*

Description

Checks the integral of normalized Markov KDE or normalized loess smoothing over edges.

Usage

```
integrate_edge_KDE(tree.discr, Edge.KDE.list)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Edge.KDE.list`	list. A list with the normalized KDEs or loess smoothing for each edge.

Value

A numeric value for the integral over all edges.

Author(s)

Sergei Tarasov

Examples

```
data("hym_kde", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get non-normalized and normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
# Check integrals.
integrate_edge_KDE(tree_discr, Edge_KDE$Maps.mean.norm)
integrate_edge_KDE(tree_discr, Edge_KDE$Maps.mean.loess.norm)
```

`join_edges` *Join neighboring edges in edge profiles.*

Description

Internal function. Not exported.

Usage

```
join_edges(tree.discr, edge.profs)
```

Author(s)

Sergei Tarasov

`KDE_unnormalized_scalar_Markov_kernel`
KDE for unnormalized Markov kernel.

Description

Internal function. Not exported.

Usage

```
KDE_unnormalized_scalar_Markov_kernel(x, h, dat)
```

Author(s)

Sergei Tarasov

`KDE_unnorm_trunc_Markov`
KDE for unnormalized Markov kernel vectorized.

Description

Internal function. Not exported.

Usage

```
KDE_unnorm_trunc_Markov(x, h, dat)
```

Author(s)

Sergei Tarasov

list2edges *Convert list to edge matrix*

Description

Takes a list of charater annotations and creates an edge matrix comprising two columns: from and to. The list to table conversion can be done using ldply function from plyr package: plyr::ldply(list, rbind).

Usage

```
list2edges(annotated.char.list, col_order_inverse = FALSE)
```

Arguments

`annotated.char.list`	character list. A character list with ontology annotations.
`col_order_inverse`	logical. The default creates the first columns consisting of character IDs and the second columns consisting of ontology annotations. The inverse order changes the columns order.

Value

Two-column matrix.

Author(s)

Sergei Tarasov

Examples

```
annot_list <- list("CH1" = c("HAO:0000933", "HAO:0000958"), "CH2" = c("HAO:0000833", "HAO:0000258"))
list2edges(annot_list)
```

loess_smoothing_KDE *Get loess smoothing for the unnormalized Markov KDE*

Description

Calculates loess smoothing for the unnormalized Markov KDE obtained from the 'estimate_edge_KDE_Markov_kernel_unn
function.

Usage

```
loess_smoothing_KDE(tree.discr, Edge.KDE)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Edge.KDE`	list. A list with the estimated unnormalized KDEs ($Maps.mean) for each edge.

Value

A list with the loess smoothing calculated for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_kde", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get non-normalized and normalized edge KDE data.
Edge_KDE <- hym_kde$head
# Calculate smoothing of edge KDE data.
Edge_KDE$Maps.mean.loess <- suppressWarnings(loess_smoothing_KDE(tree_discr, Edge_KDE))
# Check smoothing of KDE data for normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean.loess.norm[[5]]
# Check smoothing of KDE data for non-normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean.loess[[5]]
```

`make_colors` *Make color palette for image plotting*

Description

Produces a color scale for a given statistic of evolutionary rate.

Usage

```
make_colors(Stat, palette)
```

Arguments

`Stat`	numeric. A named vector where values are the statistics, and names are ontology term labels.
`palette`	A character vector or function defining a color palette.

Value

A character vector where elements are color IDs and names are the input ontology term labels.

Author(s)

Sergei Tarasov

Examples

```
stat <- setNames(runif(5, 0.1, 10), c("cranium", "fore_wing", "hind_wing", "pronotum", "propectus") )
hm.palette <- colorRampPalette(RColorBrewer::brewer.pal(9, "Spectral"), space = "Lab")
cols.maps <- make_colors(stat, palette = hm.palette(100))
cols.maps
```

---

make_colors_relative_scale

Make color palette for image plotting with relative scale

Description

Produces a relative color scale for a given statistic of evolutionary rate.

Usage

```
make_colors_relative_scale(Stat, palette, lims)
```

Arguments

`Stat`	numeric. A named vector where values are the statistics, and names are ontology term labels.
`palette`	A character vector or function defining a color palette.
`lims`	numeric. A pair of values defining the lower and upper limits of the scale.

Value

A character vector where elements are color IDs and names are the input ontology term labels.

Author(s)

Sergei Tarasov

Examples

```
stat <- setNames(runif(5, 0.1, 10), c("cranium", "fore_wing", "hind_wing", "pronotum", "propectus") )
hm.palette <- colorRampPalette(RColorBrewer::brewer.pal(9, "Spectral"), space = "Lab")
cols.maps <- make_colors_relative_scale(stat, palette = hm.palette(100), lims = c(min(stat), max(stat)))
cols.maps
```

`make_contMap_KDE` *Make contMap KDE object*

Description

Produces a contMap object for plotting the NHPP.

Usage

```
make_contMap_KDE(tree.discr, Edge.KDE.stat)
```

Arguments

`tree.discr`	phylo object. A discretized tree using the 'discr_Simmap' function.
`Edge.KDE.stat`	list. A list with the distributions of the estimated parameter of KDEs for each edge.

Value

A 'contMap' object.

Author(s)

Sergei Tarasov

Examples

```
data("hym_tree", "hym_kde")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get smoothing of normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
Edge_KDE_stat <- Edge_KDE$loess.lambda.mean
# Make contmap nhpp data.
nhpp_map <- make_contMap_KDE(tree_discr, Edge_KDE_stat)
# Plot contmap.
## Not run:

  phytools::plot.contMap(nhpp_map, lwd = 3, outline = F, legend = F, ftype = "off", plot = F)


## End(Not run)
```

make_data_NHPP_KDE_Markov_kernel

Get NHPP data for all edges (Markov KDE)

Description

Gets data on changing times between states for all edges of a given sample of trees for the Markov kernel density estimator (KDE).

Usage

```
make_data_NHPP_KDE_Markov_kernel(Tb.trees, add.psd = TRUE)
```

Arguments

`Tb.trees`	data.frame. A tibble obtained with the 'path_hamming_over_trees_KDE' function.
`add.psd`	logical. Whether to add pseudodata or not. Default is TRUE.

Value

A list with changing times between states for all edges of a given sample of trees.

Author(s)

Sergei Tarasov

Examples

```
data("hym_hm")
# Get hamming data from the head characters.
hm <- hym_hm$head
# Make NHPP path data.
nhpp <- make_data_NHPP_KDE_Markov_kernel(hm)
# Check NHPP path data for an arbitrary branch.
nhpp[[5]]
```

`make_data_NHPP_over_edge_MarkovKDE` *Get NHPP data for a given edge (Markov KDE)*

Description

Gets data on changing times between states for a given edge of a given sample of trees for the Markov kernel density estimator (KDE).

Usage

```
make_data_NHPP_over_edge_MarkovKDE(Tb.trees, Focal.Edge)
```

Arguments

`Tb.trees`	data.frame. A tibble obtained with the 'path_hamming_over_trees_KDE' function.
`Focal.Edge`	integer. A value indicating the edge ID.

Value

A numeric vector with changing times between states for a given edge.

Author(s)

Sergei Tarasov

Internal function. Not exported.

`make_pic` *Assign colors to picture ID layers*

Description

Assigns colors to picture ID layers (@paths) of an object of class 'Picture'. The object should be a PS or ESP vector illustration imported using the grImport package. Colors are taken from cols.maps argument were the palette indicates the scale of the desired statistics for the evolutionary rates.

Usage

```
make_pic(picture, layers, cols.maps)
```

Arguments

picture	grImport object. A vector image imported in R using the 'readPicture' function from grImport.
layers	numeric. A named vector where values indicate the layer IDs in the Picture object and names indicate the anatomy ontology term labels.
cols.maps	character. A named vector where elements correspond to color IDs and names indicate the anatomy ontology term labels.

Value

An object of class 'Picture' with the assigned colors to different anatomical regions.

Author(s)

Sergei Tarasov

Examples

```
data("HAO", "hym_graph", "hym_img", "hym_kde")
# Get picture.
picture <- hym_img
# Get picture layers from three anatomical regions.
terms_list <- as.list(c("HAO:0000397", "HAO:0000576", "HAO:0000626"))
terms_list <- setNames(terms_list, c("head", "mesosoma", "metasoma"))
anat_layers <- get_vector_ids_list(terms = terms_list , ONT = HAO, GR = hym_graph)
# Get mean rates all branches for the three anatomical regions.
plot_stat <- lapply(hym_kde, function(x) unlist(lapply(x$loess.lambda.mean, function(x) mean(x) )) )
plot_stat <- do.call(cbind, plot_stat)
# Select an arbitrary branch.
plot_stat <- plot_stat[50,]
# Set scale.
scale_lim <- range(plot_stat)
# Get color palette.
hm.palette <- colorRampPalette(RColorBrewer::brewer.pal(9, "Spectral"), space = "Lab")
cols_maps <- make_colors_relative_scale(plot_stat, palette = hm.palette(100),
                                        lims = scale_lim)
# Plot picture.
new_pic <- make_pic(picture, anat_layers, cols_maps)
## Not run:

  grImport::grid.picture(new_pic)


## End(Not run)
```

`make_postPois_KDE` *Make posterior distribution of NHPP*

Description

Produces a posterior distribution from a given list of statistics calculated with the 'posterior_lambda_KDE' function.

Usage

```
make_postPois_KDE(Edge.KDE.stat, lambda.post, lambda.post.stat = "Mean")
```

Arguments

`Edge.KDE.stat`	list. A list with the estimated normalized or loess smoothing KDEs for each edge.
`lambda.post`	list. A list with the distribution statistics calculated with the 'posterior_lambda_KDE' function.
`lambda.post.stat`	character. A value with the statistic to be used.

Value

A list with the distribution of the selected statistic for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg", "hym_kde")
# Get a sample of ten stochastic maps from head.
tree_list <- hym_stm_amalg$head[1:10]
tree_list <- merge_tree_cat_list(tree_list)
# Calculate posterior poisson statistics.
lambda_post <- posterior_lambda_KDE(tree_list)
# Get smoothing of normalized edge KDE data for mean rates.
Edge_KDE <- hym_kde$head
Edge_KDE_stat <- Edge_KDE$Maps.mean.loess.norm
# Make posterior poisson distribution.
Edge_KDE$lambda.mean <- make_postPois_KDE(Edge_KDE_stat, lambda_post, lambda.post.stat = "Mean")
# Check posterior poisson of some arbitrary branch.
## Not run:

  plot(density(Edge_KDE$lambda.mean[[5]]), main = "", xlab = "Rates")


## End(Not run)
```

mds_plot *Plot morphospace from MDS*

Description

Wrapper function for plotting morphospaces obtained using the MultiScale.simmap' function.

Usage

```
mds_plot(MD, Tslice = max(MD$Points$time))
```

Arguments

MD	list. A list with the morphospace information obtained using the 'MultiScale.simmap' function.
Tslice	numeric. The value for the temporal slices to be plotted, from root to tip. For example, if Tslice = 25, then all points in the morphospaces from time 0 (root) to 25 will be plotted.

Author(s)

Diego Porto

Examples

```
data("hym_stm_mds")
# Get a sample of 5 amalgamated stochastic maps.
tree_list <- hym_stm_mds
tree_list <- merge_tree_cat_list(tree_list)
# Multidimensional scaling for an arbitrary tree.
## Not run:

  MD <- suppressWarnings(MultiScale.simmap(tree_list[[1]], add.noise = c(0.3,0.3)))

  MD_plot <- mds_plot(MD, Tslice = 10)
  MD_plot
  MD_plot <- mds_plot(MD, Tslice = 50)
  MD_plot
  MD_plot <- mds_plot(MD, Tslice = 200)
  MD_plot
  MD_plot <- mds_plot(MD, Tslice = 280)
  MD_plot


## End(Not run)
```

merge_branch_cat *Merge state bins over branch*

Description

Merges identical state bins over the same branch in the discretized stochastic map.

Usage

```
merge_branch_cat(br)
```

Arguments

br	numeric or character vector. The branches of the tree.

Value

A numeric or character vector with merged identical bins.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm")
tree <- hym_stm[[1]][[1]]
stm_discr <- discr_Simmap(tree, res = 100)
# Check some arbitrary branch.
br1 <- stm_discr$maps[[50]]
br1
br2 <- merge_branch_cat(br1)
br2
sum(br1) == br2
```

merge_tree_cat *Merge state bins over a tree*

Description

Merges identical state bins over a tree in the discretized stochastic map.

Usage

```
merge_tree_cat(tree)
```

Arguments

`tree`	simmap object.

Value

A tree with merged identical bins.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm")
tree <- hym_stm[[1]][[1]]
tree <- discr_Simmap(tree, res = 100)
stm_merg <- merge_tree_cat(tree)
# Check some arbitrary branch.
br1 <- tree$maps[[50]]
br1
br2 <- stm_merg$maps[[50]]
br2
sum(br1) == br2
```

`merge_tree_cat_list` *Merge state bins over a tree list*

Description

A wrapper function to merge identical state bins over a tree list.

Usage

```
merge_tree_cat_list(tree.list)
```

Arguments

`tree.list`	multiSimmap object.

Value

A list of trees with merged identical bins.

Author(s)

Diego S. Porto

Examples

```
data("hym_stm")
tree_list <- hym_stm[[1]]
tree_list <- discr_Simmap_all(tree_list, res = 100)
stm_merg_list <- merge_tree_cat_list(tree_list)
# Check some arbitrary branch of some arbitrary tree.
br1 <- tree_list[[1]]$maps[[50]]
br1
br2 <- stm_merg_list[[1]]$maps[[50]]
br2
sum(br1) == br2
```

`MultiScale.simmap` *Multidimensional scaling of character states from one stochastic character map*

Description

Performs multidimensional scaling (MDS) based on hamming distances among character state vectors from one stochastic character map.

Usage

```
MultiScale.simmap(tree.merge, add.noise = NULL)
```

Arguments

`tree.merge`	simmap object. A stochastic character map after being processed by the discr_Simmap and merge_tree_cat functions.
`add.noise`	numeric. A vector of length 2 or NULL. Indicates if noise should be added or not. Useful if there are many identical states occupying the same points in the 2D coordinates of the morphospace plot. The noise is calculated as var(V)*add.noise.

Value

A list of tibbles – Points, Lines, and Edge.map – correponding to tree branch information to plot.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_mds")
# Get a sample of 5 amalgamated stochastic maps.
tree_list <- hym_stm_mds
tree_list <- merge_tree_cat_list(tree_list)
# Multidimensional scaling for an arbitrary tree.
## Not run:

  MD <- suppressWarnings(MultiScale.simmap(tree_list[[1]], add.noise = c(0.3,0.3)))


## End(Not run)
```

`normalize_KDE` *Normalize loess smoothing*

Description

Normalizes the loess smoothing for the Markov KDE.

Usage

```
normalize_KDE(tree.discr, Maps.mean.loess)
```

Arguments

`tree.discr`	simmap or phylo object. A discretized tree using the 'discr_Simmap' function.
`Maps.mean.loess`	list. A list with the loess smoothing calculated for each edge using the 'loess_smoothing_KDE' function.

Value

A list with the normalized loess smoothing calculated for each edge.

Author(s)

Sergei Tarasov

Examples

```
data("hym_kde", "hym_tree")
# Get reference tree.
tree_discr <- discr_Simmap(hym_tree, res = 4000)
# Get non-normalized and normalized edge KDE data.
Edge_KDE <- hym_kde$head
# Calculate smoothing of edge KDE data.
Edge_KDE$Maps.mean.loess <- suppressWarnings(loess_smoothing_KDE(tree_discr, Edge_KDE))
```

```
# Normalize smoothing edge KDE data.
Edge_KDE$Maps.mean.loess.norm <- normalize_KDE(tree_discr, Edge_KDE$Maps.mean.loess)
# Check smoothing of KDE data for non-normalized mean rates from an arbitrary branch.
Edge_KDE$Maps.mean.loess[[5]]
```

`paramo` *PARAMO*

Description

Wrapper function to perform the final paramo stacking of maps for a set of anatomy ontology terms.

Usage

```
paramo(rac_query, tree.list, ntrees)
```

Arguments

`rac_query`	character list. Named list obtained from the RAC_query function.
`tree.list`	multiSimmap or multiPhylo object. Named list with stochastic character maps.
`ntrees`	integer. Number of trees to stack.

Value

A list of stacked stochastic character maps.

Author(s)

Diego S. Porto

Examples

```
char_info <- hym_annot[1:2]
# Query for three anatomical regions.
terms <- c("head", "mesosoma", "metasoma")
query <- RAC_query(char_info, HAO, terms)
# Select the first three characters for each anatomical region.
query <- lapply(query, function(x) x[1:3])
# Subset the list of multiple maps.
tree_list <- hym_stm[unname(unlist(query))]
tree_list <- lapply(tree_list, function(x) discr_Simmap_all(x, res = 100))
tree_list_amalg <- paramo(query, tree_list, ntrees = 100)
tree_list_amalg <- lapply(tree_list_amalg, function(x) do.call(c,x) )
# Get one sample of map from head.
stm_hd <- tree_list_amalg$head[[1]]
# Get one sample of map from mesosoma.
stm_ms <- tree_list_amalg$mesosoma[[1]]
# Get one sample of map from metasoma.
```

```
stm_mt <- tree_list_amalg$metasoma[[1]]
# Plot one amalgamated stochastic map from each anatomical region.
## Not run:

  phytools::plotSimmap(stm_hd, get_rough_state_cols(stm_hd),  lwd = 3, pts = F,ftype = "off", ylim = c(0,100))
  phytools::plotSimmap(stm_ms, get_rough_state_cols(stm_ms),  lwd = 3, pts = F,ftype = "off", ylim = c(0,100))
  phytools::plotSimmap(stm_mt, get_rough_state_cols(stm_mt),  lwd = 3, pts = F,ftype = "off", ylim = c(0,100))


## End(Not run)
```

paramo.list *Stack multiple discrete stochastic character map lists*

Description

Performs the final stacking of maps for a set of stochastic character maps stored in a list.

Usage

```
paramo.list(cc, tree.list, ntrees = 1)
```

Arguments

`cc`	character. Characters IDs to stack.
`tree.list`	multiSimmap or multiPhylo object. Named list with stochastic character maps.
`ntrees`	integer. Number of trees to stack.

Value

A list of stacked stochastic character maps.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm")
# Select the first five characters.
tree_list <- hym_stm[1:5]
tree_list <- lapply(tree_list, function(x) discr_Simmap_all(x, res = 100))
tree_list_amalg <- paramo.list(names(tree_list), tree_list, ntrees = 100)
tree_list_amalg <- do.call(c, tree_list_amalg)
# Plot one amalgamated stochastic map.
## Not run:

  phytools::plotSimmap(tree_list_amalg[[1]], get_rough_state_cols(tree_list_amalg[[1]]),  lwd = 3, pts = F,ftype =
## End(Not run)
```

`path_hamming` *Path hamming*

Description

Calculates the hamming distance between states for a given path.

Usage

```
path_hamming(Path)
```

Arguments

`Path`	data.frame. A tibble with state information about a given path (from root to a given node). The tibble is the output obtained from the get_states_path function. The columns give information on state changes, time spent on each state, and edge IDs.

Value

The input tibble with two additional columns giving information on absolute and normalized hamming distances.

Author(s)

Sergei Tarasov

Internal function. Not exported.

`path_hamming_over_all_edges`
Hamming distances for a tree

Description

Calculates hamming distances for all paths in a given discretized tree.

Usage

```
path_hamming_over_all_edges(tree.merge)
```

Arguments

`tree.merge`	simmap object. A discretized simmap using the 'discr_Simmap' function where identical state bins were merged using the 'merge_tree_cat' function.

Value

A tibble with information on state changes, time spent on each state, edge IDs, absolute and normalized hamming distances for all edges in a tree.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg")
# Get one sample of stochastic maps from head.
tree <- hym_stm_amalg$head[[1]]
tree <- merge_tree_cat(tree)
# Calculate hamming distances.
ph <- suppressWarnings(path_hamming_over_all_edges(tree))
ph
```

`path_hamming_over_trees_KDE`

Hamming distances for a list of trees

Description

Calculates hamming distances for all paths in each discretized tree of a list.

Usage

```
path_hamming_over_trees_KDE(tree.list)
```

Arguments

`tree.list`	multiSimmap object.

Value

A tibble with information on state changes, time spent on each state, edge IDs, absolute and normalized hamming distances for all edges and all trees in a list.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg")
# Get ten samples of stochastic maps from head.
tree_list <- hym_stm_amalg$head[1:10]
tree_list <- merge_tree_cat_list(tree_list)
# Calculate hamming distances.
ph <- suppressWarnings(path_hamming_over_trees_KDE(tree_list))
ph
```

---

pNHPP *Phylogenetic Non-Homogeneous Poisson Process (pNHPP) method*

Description

Wrapper function for applying the pNHPP method.

Usage

```
pNHPP(
  stm_amalg,
  tree = tree,
  res = res,
  add.psd = TRUE,
  band.width = c("bw.nrd0", "bw.nrd0", "bw.ucv", "bw.bcv", "bw.SJ"),
  lambda.post.stat = "Mean"
)
```

Arguments

`stm_amalg`	multiSimmap object. A list of amalgamated stochastic maps.
`tree`	simmap or phylo object. A reference tree for discretization.
`res`	integer. A resolution value for the discretization of tree edges.
`add.psd`	logical. Whether to add pseudodata or not in the 'make_data_NHPP_KDE_Markov_kernel' function. Default is TRUE.
`band.width`	character. Bandwidth selectors for the KDE in the 'estimate_band_W' function.
`lambda.post.stat`	character. A value with the statistic to be used in the 'make_postPois_KDE' function.

Value

A list with the estimated Markov KDE for all edges, the contMap object for plotting the NHPP, and the information necessary for making the edgeplot.

Author(s)

Diego Porto

Examples

```
## Not run:

  # Load data.
  data("hym_stm", "hym_stm_amalg")
  # Get a reference tree for discretization.
  tree <- hym_stm[[1]][[1]]
  # Get ten samples of stochastic maps from head.
  tree_list <- hym_stm_amalg$head[1:10]
  # Run the pNHPP method.
 nhpp_test <- pNHPP(tree_list, tree, res = 500, add.psd = TRUE, band.width = 'bw.nrd', lambda.post.stat = 'Mean')


## End(Not run)
```

`posterior_lambda_KDE` *Get analytical posterior*

Description

Calculates the required statitics for the posterior distribution of number of state changes across all branches of all trees.

Usage

```
posterior_lambda_KDE(tree.list)
```

Arguments

`tree.list`	multiSimmap object.

Value

A list with mean ($Mean), standard deviation ($SD), and 95HPD interval ($Q_2.5 and $Q_97.5) calculated for the posterior distribution.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg")
# Get a sample of ten stochastic maps from head.
tree_list <- hym_stm_amalg$head
tree_list <- merge_tree_cat_list(tree_list[1:10])
# Calculate posterior poisson statistics.
posterior_lambda_KDE(tree_list)
```

`posterior_lambda_KDE_Distr` *Get distributions of analytical posterior*

Description

Simulates a distribution of number of state changes across all branches of all trees

Usage

```
posterior_lambda_KDE_Distr(tree.list, n.sim = 10, BR.name)
```

Arguments

`tree.list`	multiSimmap object.
`n.sim`	integer. Number of simulations.
`BR.name`	character. A label name for the anatomical region.

Value

A tibble with the simulated distribution.

Author(s)

Sergei Tarasov

Examples

```
data("hym_stm_amalg")
# Get a sample of ten stochastic maps from head.
tree_list <- hym_stm_amalg$head[1:10]
tree_list <- merge_tree_cat_list(tree_list)
# Simulate posterior poisson distribution.
posterior_lambda_KDE_Distr(tree_list, n.sim = 10, BR.name = "head")
```

RAC_query *Retrieve all characters under a given set of terms*

Description

Returns a named list aggregating characters under a specified set of terms (e.g., body regions).

Usage

```
RAC_query(char_info, ONT, terms)
```

Arguments

`char_info`	data.frame. A data.frame with two columns: the first column with character IDs and the second column with ontology IDs.
`ONT`	ontology_index object.
`terms`	character. The set of terms to aggregate characters.

Value

A named list with character groups.

Author(s)

Sergei Tarasov

Examples

```
data("HAO", "hym_annot")
char_info <- hym_annot[1:2]
# Query for three anatomical regions.
terms <- c("head", "mesosoma", "metasoma")
query <- RAC_query(char_info, HAO, terms)
```

read_Simmap_Rev *Reading stochastic character maps file from ReVBayes*

Description

Imports stochastic character maps file from RevBayes into R.

Usage

```
read_Simmap_Rev(file, start = 1, end = 1, save = NULL)
```

Arguments

`file`	character. Path to the RevBayes file.
`start`	integer. First tree of the sample to start reading the RevBayes file.
`end`	integer. Last tree of the sample to finish reading the RevBayes file.
`save`	character. Name to save output file.

Value

A tree in 'phylip' format.

Author(s)

Sergei Tarasov

Examples

```
## Not run:

  stm <- read_Simmap_Rev("revbayes.stm", start = 200, end = 500, save = NULL)
  stm <- phytools::read.simmap(text = stm, format = "phylip")
  phytools::plotSimmap(stm[[1]])


## End(Not run)
```

`stack2` *Stack two discrete stochastic character map lists; x and y are the list of state names (i.e. maps).*

Description

Internal function. Not exported.

Usage

```
stack2(x, y)
```

Author(s)

Sergei Tarasov

stack_stm *Stack two discrete stochastic character maps.*

Description

Internal function. Not exported.

Usage

```
stack_stm(stm.list)
```

Author(s)

Sergei Tarasov

Index