Package: POMS (via r-universe)

September 1, 2024

Title Phylogenetic Organization of Metagenomic Signals

Version 1.0.1

Description Code to identify functional enrichments across diverse taxa in phylogenetic tree, particularly where these taxa differ in abundance across samples in a non-random pattern. The motivation for this approach is to identify microbial functions encoded by diverse taxa that are at higher abundance in certain samples compared to others, which could indicate that such functions are broadly adaptive under certain conditions. See 'GitHub' repository for tutorial and examples: <https://github.com/gavinmdouglas/POMS/wiki>. Citation: Gavin M. Douglas, Molly G. Hayes, Morgan G. I. Langille, Elhanan Borenstein (2022) <doi:10.1093/bioinformatics/btac655>.

License GPL-3

Imports ape (>= 3.0), data.table, MASS, parallel (>= 3.3.0), phangorn (>= 2.0.0), phylolm (>= 2.6), utils, XNomial (>= 1.0.4)

Suggests testthat (>= 3.0.0)

Config/testthat/edition 3

Encoding UTF-8

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.3

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

RemoteUrl https://github.com/gavinmdouglas/poms

RemoteRef HEAD

RemoteSha f1eca60b651275780e48cb5f2bd5b76884d6dfff

Contents

`abun_isometric_log_ratios` *Compute isometric log ratio based on abundance of feature sets*

Description

Computes isometric log ratio between two sets of feature abundances, for each sample separately. Requires an abundance table, with two sets of features for which the ratio will be computed.

Usage

```
abun_isometric_log_ratios(
  abun_table,
  set1_features,
  set2_features,
  pseudocount = NULL
)
```

Arguments

`abun_table`	Abundance table, e.g., read counts or relative abundance. Should be dataframe with column names correspond to sample names and row names corresponding to the feature ids. No 0's are permitted unless the "pseudocount" option is set.
`set1_features`	Features (rows of abundance table) that make up one side of the ratio to be computed (numerator).
`set2_features`	Same as "set1_features", but for the other side of the ratio (denominator).
`pseudocount`	Constant to add to all abundance values, to ensure that there are only non-zero values. For read count data this would typically be 1.

Value

Numeric vector of the computed isometric log ratio for each sample (where samples are taken to be each column in the input table).

compute_node_balances *Compute balances at tree nodes.*

Description

Computes balances (i.e., isometric log ratios, for each sample separately) of feature abundances at each non-negligible node in the tree.

Usage

```
compute_node_balances(
  tree,
  abun_table,
  min_num_tips = 10,
  ncores = 1,
  pseudocount = NULL,
  derep_nodes = FALSE,
  jaccard_cutoff = 0.75,
  subset_to_test = NULL
)
```

Arguments

tree	Phylo object with tip labels matching row names of input abundance table. Note that node labels are required.
abun_table	Abundance table, e.g., read counts or relative abundance. Should be dataframe with column names correspond to sample names and row names corresponding to the tips of the tree. No 0's are permitted unless the "pseudocount" option is set.
min_num_tips	Minimum number of tips that must be found on each side of a node for it to be included (i.e., to be considered non-negligible).
ncores	Number of cores to use for steps of function that can be run in parallel.
pseudocount	Optional constant to add to all abundance values, to ensure that there are only non-zero values. For read count data this would typically be 1.
derep_nodes	Boolean setting to specify whether nodes should be dereplicated based on the Jaccard similarity of the underlying tips. When TRUE, nodes with pairwise Jaccard similarity >= jaccard_cutoff will be collapsed into the same cluster. A node will be added to a cluster if it is adequately similar to any nodes in a cluster. One representative per cluster will be retained, which will correspond to the node with the fewest underlying tips. Note that this step is performed after the step involving the min_num_tips screening.
jaccard_cutoff	Numeric vector of length 1. Must be between 0 and 1 (inclusive). Corresponds to the Jaccard cut-off used for clustering nodes based on similar sets of underlying tips.

`subset_to_test` Optional vector of node labels (*not indices*) that correspond to the subset of nodes that should be considered. Note that balances will still only be computed at each of these nodes if they have a sufficient number of underlying tips (as specified by the "min_num_tips" argument). If this argument is not specified then all nodes will be considered.

Value

List containing three objects:

"tips_underlying_nodes": the tips on the left-hand side (lhs; the numerator) and right-hand side (rhs; the denominator) of each node. Note that which side of the node is denoted as the left-hand or right-hand side is arbitrary.

"balances": list with each non-negligible node as a separate element. The sample balances for each node are provided as a numeric vector within each of these elements.

"negligible_nodes": character vector of node labels considered negligible. This is defined as those with fewer tips on either side of the node than specified by the "min_num_tips" argument.

When derep_nodes = TRUE, additional elements will also be returned:

"ignored_redundant_nodes": character vector of (non-negligible) node labels ignored due to being in sharing high Jaccard similarity with at least one other node.

"node_pairwise_jaccard": dataframe of pairwise Jaccard similarity for all non-negligible nodes.

"node_clusters": list with the node labels clustered into each unique cluster of nodes based on Jaccard similarities. Each list element is a separate cluster for which only one node was selected as a representative (whichever one had the fewest underlying tips).

`filter_rare_table_cols` *Filters out columns of dataframe based on number of proportion of non-zero cells*

Description

Filters dataframe columns with either a low absolute count of non-zero values or a low proportion of rows with non-zero counts. Note that this function is intended for positively-bounded data only (e.g., the function or taxon abundance tables), and will not work properly if the table contains negative values. Included in package simply to make running workflow easier.

Usage

```
filter_rare_table_cols(
  in_tab,
  min_nonzero_count,
  min_nonzero_prop,
  drop_missing_rows = TRUE,
  verbose = TRUE
)
```

Arguments

in_tab	input dataframe
min_nonzero_count	minimum number of cells in column that must be non-zero for column to be retained.
min_nonzero_prop	minimum proportion of cells in column that must be non-zero for column to be retained.
drop_missing_rows	boolean flag to indicate whether rows with all zero values (after dropping columns based on specified cut-offs) should be removed.
verbose	boolean flag to indicate that the number of columns removed should be written to the console.

Value

dataframe with columns that did not meet the min_nonzero_count and/or min_nonzero_prop options removed (and potentially rows dropped too if drop_missing_rows=TRUE).

---

genome_content_phylo_regress

Phylogenetic regression of input vector against function presence/absence.

Description

Runs phylogenetic regression with phylolm on each function (or trait) in the specified function table.

Usage

```
genome_content_phylo_regress(y, func, in_tree, ncores = 1, model_type = "BM")
```

Arguments

y	variable to use for y component of model. Typically would be either a binary vector indicating which taxa are significantly different, or the normalized specicity or normalized prevalence values. Must be a named numeric vector with names matching the rows of the func dataframe. These names also must match the tree tip labels, but they can be a subset and any missing tips will be dropped.
func	dataframe of the number of copies of each function that are encoded by each input taxon. This pipeline only considers the presence/absence of functions across taxa. Taxa (with row names intersecting with the "abun" table) should be the rows and the functions should be the columns.
in_tree	phylo object. Tip labels must include the row names of the func dataframe and the names of the y input vector.

ncores	integer specifying how many cores to use for parallelized sections of pipeline.
model_type	length-one character vector specifying which phylogenetic model to use (must be a possible setting of the model argument to the phylolm function).

Value

Dataframe summarizing the phylolm coefficients and model p-values for each y ~ function comparison. Will include the intercept, slope, and p-value for each case. Row names will be function ids.

---

node_taxa *Determine taxa labels of tips on each side of a node*

Description

Takes in a tree, a table of taxa labels per tip, and either a node label or index.

Usage

```
node_taxa(
  in_tree,
  taxon_labels,
  node_label = NULL,
  node_index = NULL,
  threshold = 0.75,
  combine_labels = TRUE
)
```

Arguments

in_tree	Phylo object
taxon_labels	Dataframe of taxa labels for tips in tree. All tips underlying the specified node must be present, although typically all tips in the tree would be present. Row names must be the tip labels. The column names correspond to each taxonomic level, such as Kingdom, Phylum, etc. The actual column names do not matter: it is just important that the order of the taxonomic levels goes from the highest taxonomic level present (e.g., Kingdom), to the lowest taxonomic level present (e.g., Species).
node_label	Optional label of node for which the representative taxon label will be determined. Either this option or the node_index option must be specified, but not both.
node_index	As above for the node_label option, but to specify a node by index rather than by label.

threshold	Float > 0.5 and <= 1.0 specifying the proportion of tips that must share a taxon label for it to be considered representative.
combine_labels	Boolean flag for whether taxon labels should be combined, so that all higher taxonomic labels are included. Specifically, when TRUE, all higher labels are concatenated and delimited by "; ". E.g., rather than just the genus "Odoribacter" the label would be "Bacteria; Bacteroidetes; Bacteroidia; Bacteroidales; Porphyromonadaceae; Odoribacter", given that those were the labels of the higher taxonomic levels of that genus.

Details

The format of the taxa label table is very important to note: it must have the tips as the rownames and taxonomic levels (ranging from highest to lowest) as the column names.

For each side of the specified node separately, this function returns the lowest possible taxon label shared by at least the specified proportion of tips (set by the "threshold" variable). Will return "Unclear" if there is no applicable taxon.

To clarify, the taxon that meets the threshold at the lowest possible taxonomic level will be used as the representative label. For example, if all the tips on one side of the node are members of the Pseudomonas genus, but only 60% are members of the Pseudomonas aeruginosa species specifically, then Pseudomonas will be used as the representative label (based on a threshold of 0.75 or higher and assuming that species are the last column in the table).

Value

Character vector of size two with the representative taxon for tips on each side of the specified node.

---

phylolm_summary *Wrapper for running phylogenetic regression with phylolm*

Description

Runs basic case of single x and y variables (dummy or continuous). Note that the ordering of the input vectors and the tree tip labels needs to be checked by the user beforehand: this script does not require that the y and x variables are named, and so no name check is performed.

Usage

```
phylolm_summary(y, x, in_tree, model_type = "BM")
```

Arguments

y	variable to use for y component of model.
x	variable to use for x component of model.
in_tree	phylo object. Tip label order is assumed to match the y and x variables.
model_type	length-one character vector specifying which phylogenetic model to use (must be a possible setting of the model argument to the phylolm function).

Value

Numeric vector of length three, providing the estimated coefficients for the intercept and slope, along with the p-value.

`POMS_pipeline` *Main function to run POMS pipeline*

Description

See details below.

Usage

```
POMS_pipeline(
  abun,
  func,
  tree,
  group1_samples = NULL,
  group2_samples = NULL,
  ncores = 1,
  pseudocount = 1,
  manual_BSNs = NULL,
  manual_balances = NULL,
  manual_BSN_dir = NULL,
  min_num_tips = 10,
  min_func_instances = 10,
  min_func_prop = 0.001,
  multinomial_min_FSNs = 5,
  derep_nodes = FALSE,
  jaccard_cutoff = 0.75,
  BSN_p_cutoff = 0.05,
  BSN_correction = "none",
  FSN_p_cutoff = 0.05,
  FSN_correction = "none",
  func_descrip_infile = NULL,
  multinomial_correction = "BH",
  detailed_output = FALSE,
  verbose = FALSE
)
```

Arguments

abun	dataframe of taxa abundances that are at the tips of the input tree. These taxa are usually individual genomes. The taxa need to be the rows and the samples the columns.

`func` dataframe of the number of copies of each function that are encoded by each input taxon. This pipeline only considers the presence/absence of functions across taxa. Taxa (with row names intersecting with the "abun" table) should be the rows and the functions should be the columns.

`tree` phylo object with tip labels that match the row names of the "abun" and "func" tables. This object is usually based on a newick-formatted tree that has been read into R with the ape R package.

`group1_samples` character vector of column names of "abun" table that correspond to the first sample group. This grouping is used for testing for significant sample balances at each node. Required unless the "manual_BSN_dir" argument is set (i.e., if the binary directions of BSNs are specified manually).

`group2_samples` same as "group1_samples", but corresponding to the second sample group.

`ncores` integer specifying how many cores to use for parallelized sections of pipeline.

`pseudocount` number added to all cells of "abun" table to avoid 0 values. Set this to be 0 if this is not desired. Note that there will be issues with the balance tree approach if any 0's are present.

`manual_BSNs` optional vector of node names that match node labels of input tree. These nodes will be considered the set of balance-significant nodes, and the Wilcoxon tests will not be run. The group means of the balances at each node will still be used to determine which group has higher values. Note this requires that the "manual_balances" argument is also specified.

`manual_balances`
optional list of balance values which represent the balances at all tested nodes that resulted in the input to the manual_BSNs vector. This list must include balances for all nodes in the manual_BSNs vector, but also all non-significant tested nodes as well. These node labels must all be present in the input tree. The required list format is the "balances" object in the output of compute_node_balances. Note, however, that any approach for computing balances could be used, as long as they are in this list format.

`manual_BSN_dir` optional character vector specifying "group1" or "group2", depending on the direction of the BSN difference. This must be a named vector, with all names matching the set of nodes specified by the manual_BSNs argument. Although this requires that the exact labels "group1" or "group2" are specified, these categories could represent different binary divisions rather than strict sample groups. For instance, "group1" could be used to represent nodes where sample balances are positively associated with a continuous variable (rather than a discrete grouping), whereas "group2" could represent nodes where sample balances are negatively associated.

`min_num_tips` minimum number of tips on each side of the nodes that is required for them to be retained in the analysis. This argument is ignored if significant nodes are specified manually.

`min_func_instances`
minimum number of tips that must encode the function for it to be retained for the analysis.

`min_func_prop` minimum proportion of tips that must encode the function for it to be retained for the analysis.

multinomial_min_FSNs
: The minimum number of FSNs required to run a multinomial test for a given function.

derep_nodes
: boolean value specifying whether nodes should be dereplicated based on similar sets of underlying tips (EXPERIMENTAL setting). More specifically, whether nodes should be clustered based on how similar their underlying tips are (given a Jaccard index cut-off, specified as separately), and then only retaining the node with the fewest underlying tips per cluster.

jaccard_cutoff
: Numeric vector of length 1. Must be between 0 and 1 (inclusive). Corresponds to the Jaccard cut-off used for clustering nodes based on similar sets of underlying tips (when derep_nodes = TRUE).

BSN_p_cutoff
: significance cut-off for identifying BSNs.

BSN_correction
: multiple-test correction to use on Wilcoxon test p-values when identifying BSNs. Must be in p.adjust.methods.

FSN_p_cutoff
: significance cut-off for identifying FSNs.

FSN_correction
: multiple-test correction to use on Fisher's exact test p-values when identifying FSNs. Must be in p.adjust.methods.

func_descrip_infile
: optional path to mapfile of function ids (column 1) to descriptions (column 2). This should be tab-delimited with no header and one function per line. If this option is specified then an additional description column will be added to the output table.

multinomial_correction
: multiple-test correction to use on raw multinomial test p-values. Must be in p.adjust.methods.

detailed_output
: boolean flag to indicate that several intermediate objects should be included in the final output. This is useful when troubleshooting issues, but is not expected to be useful for most users.
The additional results include:
  - balance_comparisons (summary of Wilcoxon tests on balances)
  - func_enrichments (Fisher's exact test output for all functions at each node)
  - input_param (a list containing the specified input parameters)

verbose
: boolean flag to indicate that log information should be written to the console.

Details

Identifies significant nodes based on sample balances, using a Wilcoxon test by default. Alternatively, significant nodes can be manually specified. Either way, significant nodes based on sample balances are referred to as Balance-Significant Nodes (BSNs).

Fisher's exact tests are run at each node in the tree with sufficient numbers of underlying tips on each side to test for functional enrichment. Significant nodes based on this test are referred to as Function-Significant Nodes (FSNs). The set of FSNs is determined independently for each tested function.

The key output is the tally of the intersecting nodes based on the sets of BSNs and FSNs.

Each FSN can be categorized in one of three ways:

- It does not intersect with any BSN.
- It intersects with a BSN and the functional enrichment is within the taxa that are relatively more abundant in group 1 samples.
- Same as the second point, but enriched within taxa that are relatively more abundant in group 2 samples.

A multinomial test is run to see if the number of FSNs of each type is significantly different from the random expectation.

Value

list containing (at minimum) these elements:

- results: dataframe with each tested function as a row and the numbers of FSNs of each type as columns, as well as the multinomial test output.
- balance_info: list containing the tips underlying each node, which were what the balances are based on, the balances themselves at each tested node, and the set of nodes that were determined to be negligible due to having too few underlying tips. Note that the balances and underlying tips are provided for all non-negligible (i.e., tested) nodes, not just those identified as BSNs. Additional information on the dereplication and Jaccard similarity of nodes is returned as well when derep_nodes = TRUE.
- BSNs: character vector with BSNs as names and values of "group1" and "group2" to indicate for which sample group (or other binary division) the sample balances were higher.
- FSNs_summary: list containing each tested function as a separate element. The labels for nodes in each FSN category of the multinomial test are listed per function (or are empty if there were no such FSNs).
- tree: the prepped tree used by the pipeline, including the added node labels if a tree lacking labels was provided. This tree will also have been subset to only those tips found in the abundance table, and midpoint rooted (if it was not already rooted).
- multinomial_exp_prop: expected proportions of the three FSN categories used for multinomial test.

---

prep_func_node_info *Get node indices of FSN and BSN categories across tree for a given function*

Description

Parse POMS_pipeline output to look at FSNs for a specific function (e.g., a specific gene family). Will also parse BSN information (which is not dependent on a particular function). This is convenient to do before plotting the distribution of FSNs and BSNs across the tree with the ggtree R package for instance. When a taxa label table is specified, labels of tested nodes in the tree (found in the POMS_pipeline output object) will be renamed to be the representative taxa on each side.

Usage

```
prep_func_node_info(
  POMS_output,
  func_id,
  taxa_table = NULL,
  taxa_threshold = 0.75,
  full_taxon_label = FALSE
)
```

Arguments

POMS_output	output object from POMS_pipeline function.
func_id	label of function for which should FSNs should be parsed. Must be present in POMS_output$FSNs_summary.
taxa_table	optional dataframe containing taxa labels for each tip of tree. Must be in same format as expected for node_taxa function.
taxa_threshold	float > 0.5 and <= 1.0 specifying the proportion of tips that must share a taxon label for it to be considered representative. Only relevant if taxa_table specified.
full_taxon_label	boolean flag for whether taxon labels should be combined, so that all higher taxonomic labels are included. Specifically, when TRUE, all higher labels are concatenated and delimited by "; ". E.g., rather than just the genus "Odoribacter" the label would be "Bacteria; Bacteroidetes; Bacteroidia; Bacteroidales; Porphyromonadaceae; Odoribacter". Only relevant if taxa_table specified.

Value

List containing final tree as well as indices of nodes corresponding to different FSN and BSN categories. If taxa_table was specified, then node labels in tree will correspond to representative taxa on each side of the nodes that were tested (i.e., those that were non-negligible).

---

prevalence_norm_logit *Compute additive smoothed prevalence of features (e.g, taxa), restricted to samples of a particular metadata category.*

Description

This code replicates the prevalence score introduced in phylogenize. The code here is modified from the phylogenize code base (https://bitbucket.org/pbradz/phylogenize/src/master/package/phylogenize/R/; commit 6f1bdba9c5a9ff04e90a8ad77bcee8ec9281730d).

Usage

```
prevalence_norm_logit(
  abun_table,
  meta_table,
  focal_var_level,
  var_colname,
  sample_colname,
  silence_citation = FALSE
)
```

Arguments

abun_table	abundance table to use for computing prevalence. Features must be rows and samples columns. All values greater than 0 will be interpreted as present.
meta_table	dataframe object containing metadata for all samples. Must include at least one column corresponding to the sample ids and one column containing the metadata of interest that will be focused on when computing prevalence.
focal_var_level	length-one character vector specifying the variable value to restrict inferences of prevalence to. In other words, prevalence will be computed based on the sample set that contain this value of the variable of interest in the metadata table.
var_colname	length-one character vector specifying the name of column in the metadata table that contains the metadata of interest (e.g., where focal_var_level can be found).
sample_colname	length-one character vector specifying the name of column in the metadata table that contains the sample ids.
silence_citation	length-one Boolean vector specifying whether to silence message notifying user about phylogenize package and paper.

Details

This algorithm is descibed in detail in Bradley et al. 2018. Phylogeny-corrected identification of microbial gene families relevant to human gut colonization. PLOS Computational Biology.

Value

Numeric vector with the normalized prevalence score for each input feature (i.e., for each row of abun_table).

`specificity_scores` *Compute shrunken specificity score of a feature, which represents how the presence of a feature is associated with a given sample grouping.*

Description

This code replicates the environmental specificity score introduced in phylogenize. The code here is modified from the phylogenize code base (https://bitbucket.org/pbradz/phylogenize/src/master/package/phylogenize/R/; commit 6f1bdba9c5a9ff04e90a8ad77bcee8ec9281730d).

Usage

```
specificity_scores(
  abun_table,
  meta_table,
  focal_var_level,
  var_colname,
  sample_colname,
  silence_citation = FALSE
)
```

Arguments

abun_table	abundance table to use for computing specificity Features must be rows and samples columns. All values greater than 0 will be interpreted as present.
meta_table	dataframe object containing metadata for all samples. Must include at least one column corresponding to the sample ids and one column containing the metadata of interest that will be focused on.
focal_var_level	length-one character vector specifying the variable value to restrict inferences of prevalence to. In other words, prevalence will be computed based on the sample set that contain this value of the variable of interest in the metadata table.
var_colname	length-one character vector specifying the name of column in the metadata table that contains the metadata of interest (e.g., where focal_var_level can be found).
sample_colname	length-one character vector specifying the name of column in the metadata table that contains the sample ids.
silence_citation	length-one Boolean vector specifying whether to silence message notifying user about phylogenize package and paper.

Details

This algorithm is descibed in detail in Bradley et al. 2018. Phylogeny-corrected identification of microbial gene families relevant to human gut colonization. PLOS Computational Biology.

Note thee can be some random fluctuations between re-runs of this function. The differences are usually minor, but users are strongly suggested to set a random seed before use to ensure their workflow is reproducible.

Value

Numeric vector with the specificity score for each input feature (i.e., for each row of abun_table).

`subset_by_col_and_filt` *Subset dataframe by column names and then post-filter*

Description

Subset table by set of column names. After doing this, it will remove any rows and columns that are all 0's.

Usage

```
subset_by_col_and_filt(in_tab, col2keep, verbose = TRUE)
```

Arguments

`in_tab`	input dataframe
`col2keep`	column names to retain in output (as long as they have at least one non-zero value).
`verbose`	flag to indicate that the final number of rows and columns (as well as the number removed) should be reported.

Value

dataframe with subset of specified columns (if they have at least one non-zero value), also with rows that only contain 0's removed.

Index