Package 'metricTester'

July 27, 2016

Title Test Metric and Null Model Statistical Performance

Description Explore the behavior and performance of phylogenetic metrics and null models.

Version 1.3.0

Date 2016-7-26

Author Eliot Miller, Chris Trisos & Damien Farine

Maintainer Eliot Miller <eliot.isaac@gmail.com>

URL https://github.com/eliotmiller/metricTester

BugReports https://github.com/eliotmiller/metricTester/issues

Depends R (>= 3.1.1), ape

Imports methods, geiger, spacodiR, dplyr, picante, foreach, doParallel, MASS, plotrix

Suggests testthat

License GPL-3

LazyData true

RoxygenNote 5.0.1

NeedsCompilation no

Repository CRAN

Date/Publication 2016-07-27 18:57:55

R topics documented:

`abundanceVector` *Generate regional abundance vector*

Description

Given a community data matrix of sites by species, extract the column-wise sums (the total number of individuals of each species) and expand to create a regional abundance vector.

Usage

```
abundanceVector(picante.cdm)
```

Arguments

`picante.cdm`	Community data matrix in picante format

Details

Simple function to create a regional abundance vector given a "regional" community data matrix.

Value

A character vector in the form "s1, s1, s1, s2, s2, s3, etc".

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

abund <- abundanceVector(cdm)
```

`alphaMetricSims` *Calculate alpha metrics under specified tree and community parameters*

Description

Takes a specified set of tree size, shape, community richness and abundance parameters, and calculates the alpha-level phylogenetic community structure metrics.

Usage

```
alphaMetricSims(tree.size, richness.vector, delta, abundances)
```

Arguments

`tree.size`	Number of species desired in the total tree.
`richness.vector`	Number of species to be placed in each plot. See details.
`delta`	A value for the delta transformation (Pagel 1999). Values greater than 1 push the branching events towards the root, while values less than 1 push the branching events closer to the tips. See details for particularly low delta values.
`abundances`	Vector of abundances, e.g. a repeated series of 1s for a presence/absence community data matrix, a log-normal distribution, etc. See examples.

Details

The richness.vector (number of species to be placed into each plot) is flexible. For instance, one might want give it 10:19, which would create 10 plots of species richness 10, 11, ... 19. But one could also provide rep(10, 10) to create 10 plots of 10 species each. If given a small value, e.g. 0.1, the delta parameter (tree shape) can occasionally result in oddly formatted trees that would cause errors. To deal with this, there is an internal check that will recreate a new tree and re-scale it with the desired delta. This has not been tested at delta < 0.1, and is currently programmed with a while loop. Care should be taken not to get R stuck in an indefinite loop at delta values even lower than 0.1

Value

A data frame of calculated alpha metrics, and the associated species richness of each plot.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
test <- alphaMetricSims(tree.size=50, richness.vector=30:40, delta=1,
abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1)
```

---

arenaTest *Calculate SES of each observed metric + null model combination*

Description

Given a table of results, where means, SDs, and CIs are bound to the observed scores at the corresponding richness or plot, this function calculates standardized effect scores for each observed metric + null model combination. This is intended to be used to test whether observed values deviate beyond expectations based on the distribution of SES per arena.

Usage

```
arenaTest(results.table, concat.by)
```

Arguments

`results.table`	Data frame of observed metrics with expected mean, SD and CI bound in. See example.
`concat.by`	Whether to concatenate results by richness, plot or both. If richness, observed scores are compared to all randomized scores where the plot had the corresponding richness. If plot, observed scores (e.g. those from plot 1) are compared to all randomized plot 1 scores. If both, both are run and each is saved as a separate data frame in a single list.

Details

Given a table of results, where means, SDs, and CIs are bound to the observed scores at the corresponding richness or plot, this function calculates standardized effect scores for each observed metric + null model combination. Previously the metrics being passed to the function needed to be explicitly specified, but the function now attempts to determine the names of the metrics via the results.table input.

Value

A data frame of standardized effect scores.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate a log normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community of varying richness
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#below not run for example timing issues on CRAN

#run the metrics and nulls combo function
rawResults <- metricsNnulls(tree=tree, picante.cdm=cdm, randomizations=2, cores="seq",
nulls=c("richness","frequency"), metrics=c("richness","NAW_MPD"))

#reduce the randomizations to a more manageable format
reduced <- reduceRandomizations(rawResults)

#calculate the observed metrics from the input CDM
observed <- observedMetrics(tree, cdm, metrics=c("richness","NAW_MPD"))

#summarize the means, SD and CI of the randomizations
summarized <- lapply(reduced, summaries, concat.by="richness")

#merge the observations and the summarized randomizations to facilitate significance
#testing
merged <- lapply(summarized, merge, observed)

#calculate the standardized scores of each observed metric as compared to the richness
#null model randomization.
arenaTest(merged$richness, "richness")

#do the same as above but across all null models. not run
#temp <- lapply(1:length(merged), function(x) arenaTest(merged[[x]], "richness"))
```

---

betaErrorChecker *Wrapper for summarizing error rates of beta metric randomizations*

Description

Given the results of a single iteration of the betaLinker function, returns a list of data frames summarizing the type I and II error rates of metrics both at the single plot and the entire arena level.

Usage

```
betaErrorChecker(single.iteration)
```

Arguments

single.iteration	Results of a run of the betaLinker function.

Details

This function wraps a number of smaller functions into a useful type I and II error checker. It takes a reduced list of randomizations such as those reduced from metricsNnulls with reduceRandomizations, summarizes the mean, SD, and CI of each metric plus null model either at the richness or plot level, then compares the observed metric scores to those summarized metrics. It return a list with two elements. The first is a list of data frames, where each corresponds to the standardized effect scores of the observed metrics for a given null model. The second is a list of data frames, where each corresponds to whether a given plot deviates beyond CI. For the latter, 0 corresponds to within CI, 1 corresponds to less than the CI, and 2 corresponds to greater than the CI.

Value

A list of lists of matrices. The first element of the lists refers to the results from a spatial simulation. Within each of these elements is a list of matrices, where each matrix tabulates the error rate of all tested beta diversity metrics with a given null model.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#run the betaLinker function
#below not run for timing issues on CRAN
#system.time(ex <- betaLinker(no.taxa=50, arena.length=300, mean.log.individuals=2,
#length.parameter=5000, sd.parameter=50, max.distance=30, proportion.killed=0.2,
#competition.iterations=3, no.plots=15, plot.length=30,
#randomizations=3, cores="seq",
#nulls=c("richness", "frequency")))

#test <- betaErrorChecker(ex)
```

`betaErrorWrapper` *Read in and calculate type I and II error rates of a set of beta metric tests*

Description

Reads in the results of a set of betaMultiLinker results and summarizes the type I and II error rates of the various metric-null combinations under different assembly processes.

Usage

```
betaErrorWrapper(working.directory)
```

Arguments

`working.directory`
: Optional character string specifying the working directory. If missing, the current working directory will be used.

Details

The internal Reduce call in this function is not formatted to my liking (ETM). Ideally, it would return a list of the length of the number of spatial simulations, then a data frame for each spatial simulation. Instead, consolidates all to a single data frame. It would be easy to make it work with for loops, but it should also be possible to revise the Reduce call and make it return results properly formatted.

Value

A data frame of spatial simulations, null models, and metric combinations, summarizing the results of all the betaMultiLinker runs in the working directory. Numbers refer to total number of runs that resulted in an error. Errors for the random spatial simulation refer to type I errors, errors for habitat filtering simulation refer to type II errors for detecting filtering, and errors for competitive exclusion refer to type II errors for detecting competition.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

betaLinker *Run spatial simulations, null and beta metric calculations*

Description

This function wraps a number of wrapper functions into one big metric + null tester function. Only a single test is performed, with results saved into memory.

Usage

```
betaLinker(no.taxa, arena.length, mean.log.individuals, length.parameter,
  sd.parameter, max.distance, proportion.killed, competition.iterations,
  no.plots, plot.length, randomizations, cores, simulations, nulls, metrics)
```

Arguments

no.taxa	The desired number of species in the input phylogeny
arena.length	A numeric, specifying the length of a single side of the arena
mean.log.individuals	Mean log of abundance vector from which species abundances will be drawn
length.parameter	Length of vector from which species' locations are drawn. Large values of this parameter dramatically decrease the speed of the function but result in nicer looking communities
sd.parameter	Standard deviation of vector from which species' locations are drawn
max.distance	The geographic distance within which neighboring indivduals should be considered to influence the individual in question
proportion.killed	The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half)
competition.iterations	Number of generations over which to run competition simulations
no.plots	Number of plots to place
plot.length	Length of one side of desired plot
randomizations	The number of randomized CDMs, per null, to generate. These are used to compare the significance of the observed metric scores.
cores	The number of cores to be used for parallel processing.
simulations	Optional. If not provided, defines the simulations as all of those in defineSimulations. If only a subset of those simulations is desired, then simulations should take the form of a character vector corresponding to named functions from defineSimulations. The available simulations can be determined by running names(defineSimulations()). Otherwise, if the user would like to define a new simulation on the fly, the argument simulations can take the form of a named list of new functions (simulations).

nulls	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).
metrics	Optional. If not provided, defines the metrics as all of those in defineBetaMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineBetaMetrics. The available metrics can be determined by running names(defineBetaMetrics()). If the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function wraps a number of other wrapper functions into one big beta metric + null performance tester function. Only a single test is run, with results saved into memory. To perform multiple complete tests, use the multiLinker function, which saves results to file.

Value

A list with two elements. The first is a list of data frames, with one for each spatial simulation. These provide the observed beta metric scores for each spatial simulation. The second level is a list of lists, one for each spatial simulation. Each of these is a list of data frames. There is one data frame per null model, and it summarizes the randomized metric scores for that null model for that spatial simulation. Note that this is slightly different than the regular linker() function, which does not output these raw metric scores (that function calculates SES and CI as outputs).

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
system.time(test <- betaLinker(no.taxa=50, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=30, proportion.killed=0.2,
competition.iterations=3, no.plots=15, plot.length=30,
randomizations=3, cores="seq", metrics=c("Pst", "Bst"),
nulls=c("richness", "frequency")))
```

---

betaMetricSims *Calculate beta metrics under specified tree and community parameters*

Description

Takes a specified set of tree size, shape, community richness and abundance parameters, and calculates the beta-level phylogenetic community structure metrics.

Usage

```
betaMetricSims(tree.size, richness.vector, delta, abundances, beta.iterations)
```

Arguments

`tree.size`	Number of species desired in the total tree.
`richness.vector`	Number of species to be placed in each plot. See details.
`delta`	A value for the delta transformation (Pagel 1999). Values greater than 1 push the branching events towards the root, while values less than 1 push the branching events closer to the tips. See details for particularly low delta values.
`abundances`	Vector of abundances, e.g. a repeated series of 1s for a presence/absence community data matrix, a log-normal distribution, etc. See examples.
`beta.iterations`	Because the type of beta-level phylogenetic community structure metrics used here return a single value per community data matrix, it is not possible to look for inter-metric correlations with only a single matrix and tree. To deal with this, the same tree can be used with different community data matrices. This argument specifies the number of matrices to be used per tree.

Details

The richness.vector (number of species to be placed into each plot) is flexible. For instance, one might want give it 10:19, which would create 10 plots of species richness 10, 11, ... 19. But one could also provide rep(10, 10) to create 10 plots of 10 species each. If given a small value, e.g. 0.1, the delta parameter (tree shape) can occasionally result in oddly formatted trees that would cause errors. To deal with this, there is an internal check that will recreate a new tree and re-scale it with the desired delta. This has not been tested at delta < 0.1, and is currently programmed with a while loop. Care should be taken not to get R stuck in an indefinite loop at delta values even lower than 0.1

Value

A data frame of calculated alpha metrics, and the associated species richness of each plot.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
test <- betaMetricSims(tree.size=50, richness.vector=30:40, delta=1,
abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1, beta.iterations=10)
```

`betaMetricsNnulls` *Parallelized function that calculates beta metrics on randomized matrices*

Description

This function sends out jobs to as many cores as are specified. Each randomizes the input CDM according to all defined null models, then calculates each observed beta metric on each randomized matrix.

Usage

```
betaMetricsNnulls(tree, picante.cdm, optional.dists = NULL,
  regional.abundance = NULL, distances.among = NULL, randomizations = 2,
  cores = "seq", nulls, metrics)
```

Arguments

`tree`	Phylo object
`picante.cdm`	A picante-style community data matrix with sites as rows, and species as columns
`optional.dists`	A symmetric distance matrix can be directly supplied. This option is experimental. Behavior depends on metric being used. If the metric in question relies on the phylogenetic distance matrix from a call to cophenetic(tree), then this optional distance matrix will be inserted instead.
`regional.abundance`	A character vector in the form "s1, s1, s1, s2, s2, s3, etc". Optional, will be generated from the input CDM if not provided.
`distances.among`	A symmetric distance matrix, summarizing the distances among all plots from the cdm. Optional, only used by some null models.
`randomizations`	The number of times the input CDM should be randomized and the metrics calculated across it.
`cores`	This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially.
`nulls`	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).
`metrics`	Optional. If not provided, defines the metrics as all of those in defineBetaMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineBetaMetrics. The available metrics can be determined by running names(defineBetaMetrics()). If

the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function sends out jobs to as many cores as are specified. Each randomizes the input CDM according to all defined null models, then calculates each observed metric on each randomized matrix.

Value

A list of lists of vectors. The first level has as many elements as there are randomizations. The second level has one list for each null model. Each element of this second level is a named vector corresponding to the calculated metric at each plot.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

rawResults <- betaMetricsNnulls(tree, cdm, randomizations=3,
nulls=c("richness", "frequency"))
```

---

betaMultiLinker *Run multiple simulations and calculations to test beta metric + null performance*

Description

This function runs multiple iterations of the linker function, saving results to file.

Usage

```
betaMultiLinker(no.taxa, arena.length, mean.log.individuals, length.parameter,
  sd.parameter, max.distance, proportion.killed, competition.iterations,
  no.plots, plot.length, randomizations, cores, iterations, prefix, simulations,
  nulls, metrics)
```

Arguments

no.taxa	The desired number of species in the input phylogeny
arena.length	A numeric, specifying the length of a single side of the arena
mean.log.individuals	Mean log of abundance vector from which species abundances will be drawn
length.parameter	Length of vector from which species' locations are drawn. Large values of this parameter dramatically decrease the speed of the function but result in nicer looking communities
sd.parameter	Standard deviation of vector from which species' locations are drawn
max.distance	The geographic distance within which neighboring indivduals should be considered to influence the individual in question
proportion.killed	The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half)
competition.iterations	Number of generations over which to run competition simulations
no.plots	Number of plots to place
plot.length	Length of one side of desired plot
randomizations	The number of randomized CDMs, per null, to generate. These are used to compare the significance of the observed metric scores.
cores	The number of cores to be used for parallel processing.
iterations	The number of complete tests to be run. For instance, 1 iteration would be considered a complete cycle of running all spatial simulations, randomly placing plots in the arenas, sampling the contents, creating a community data matrix, calculating observed metric scores, then comparing these to the specified number of randomizations of the original CDMs.
prefix	Optional character vector to affix to the output RData file names, e.g. "test1".
simulations	Optional. If not provided, defines the simulations as all of those in defineSimulations. If only a subset of those simulations is desired, then simulations should take the form of a character vector corresponding to named functions from defineSimulations. The available simulations can be determined by running names(defineSimulations()). Otherwise, if the user would like to define a new simulation on the fly, the argument simulations can take the form of a named list of new functions (simulations).
nulls	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).
metrics	Optional. If not provided, defines the metrics as all of those in defineBetaMetrics. If only a subset of those is desired, then metrics should take the form of a

character vector corresponding to named functions from defineBetaMetrics. The available metrics can be determined by running names(defineBetaMetrics()). If the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function wraps a number of other wrapper functions into one big beta metric + null performance tester function. Unlike the basic betaLinker function, multiple tests can be run, with results saved as RDS files.

Value

Multiple iterations of the the betaLinker function.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#system.time(betaMultiLinker(no.taxa=50, arena.length=300, mean.log.individuals=3.2,
#length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.3,
#competition.iterations=2, no.plots=20, plot.length=30,
#randomizations=3, cores="seq", iterations=2, prefix="test",
#nulls=c("richness", "frequency")))
```

---

calcBetaMetrics *Calculate phylogenetic community structure beta metrics*

Description

Given a prepped metrics.input object, calculate all phylogenetic community structure metrics of interest.

Usage

```
calcBetaMetrics(metrics.input, metrics, new_ = FALSE)
```

Arguments

metrics.input	Prepped metrics.input object

metrics	Optional. If not provided, defines the metrics as all of those in defineBetaMetrics. If only a subset of those metrics is desired, then metrics should take form of a character vector corresponding to named functions from defineBetaMetrics. The available metrics can be determined by running names(defineBetaMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument metrics can take the form of a named list of new functions (metrics). If the latter, new_ must be set to TRUE.
new_	Whether or not new metrics are being defined on the fly. Default is FALSE. Set to TRUE if a new metric is being used.

Details

This function first confirms that the input is of class metrics.input and, if so, then confirms that the metrics to be calculated are in a named list (via checkMetrics), then lapplies all metric functions to the input metrics.input object.

Value

A data frame with the calculated metrics and the associated species richness and total abundance of all input "communities".

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1))

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

prepped <- prepData(tree, cdm)

results <- calcBetaMetrics(prepped)

#an example of how to define ones own metrics for use in the metricTester framework
#this "metric" simply calculates the richness of each plot in the CDM
exampleMetric <- function(metrics.input)
{
output <- mean(apply(metrics.input$picante.cdm, 1, lengthNonZeros))
output
}

calcBetaMetrics(prepped, metrics=list("example"=exampleMetric), new_=TRUE)
```

`calcField` *Calculate phylogenetic and trait fields*

Description

Given a prepped field.input object, calculate all fields of interest.

Usage

```
calcField(field.input, metrics)
```

Arguments

`field.input`	Prepped field.input object.
`metrics`	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

Currently we are calculating 19 phylogenetic community structure metrics. This function first confirms that the input is of class metrics.input and, if so, then confirms that the metrics to be calculated are in a named list (via checkMetrics), then lapplies all metric functions to the input metrics.input object.

Value

A data frame with the calculated metrics and the associated species richness of all input "communities".

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#in this example, occasionally some species are not in the CDM, so prune the tree
```

```
#accordingly so as not to throw any errors
tree <- drop.tip(tree, setdiff(tree$tip.label, colnames(cdm)))

prepped <- prepFieldData(tree=tree, picante.cdm=cdm)

results <- calcField(prepped)
```

`calcMetrics` *Calculate phylogenetic community structure metrics*

Description

Given a prepped metrics.input object, calculate all phylogenetic community structure metrics of interest.

Usage

```
calcMetrics(metrics.input, metrics, new_ = FALSE)
```

Arguments

`metrics.input`	Prepped metrics.input object
`metrics`	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those metrics is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument metrics can take the form of a named list of new functions (metrics). If the latter, new_ must be set to TRUE.
`new_`	Whether or not new metrics are being defined on the fly. Default is FALSE. Set to TRUE if a new metric is being used.

Details

Determine which metrics will be calculated by running names(defineMetrics()). If only a subset of these is desired, supply metrics with a character vector of the named, available metrics. IMPORTANTLY, note that some downstream functions expect the first column returned from this function to be the species richness of each plot. It is best practice therefore to always pass "richness" along as the first metric, even when only a subset of metrics is being calculated. It is possible to provide this function with both predefined metrics and metrics that are defined on the fly, but the call is rather convoluted. See examples.

Value

A data frame with the calculated metrics of all input "communities".

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1))

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

prepped <- prepData(tree, cdm)

results <- calcMetrics(prepped)

#an example of how to only calculate a subset of pre-defined metrics
results2 <- calcMetrics(prepped, metrics=c("richness","NAW_MPD"))

#an example of how to define ones own metrics for use in the metricTester framework
#this "metric" simply calculates the richness of each plot in the CDM
exampleMetric <- function(metrics.input)
{
output <- apply(metrics.input$picante.cdm, 1, lengthNonZeros)
output
}

calcMetrics(prepped, metrics=list("richness"=metricTester:::my_richness,
"example"=exampleMetric), new_=TRUE)
```

---

centers *Calculate weighted centroids*

Description

Calculate the weighted centroids of clouds of multivariate points.

Usage

```
centers(ordination.results, road.map)
```

Arguments

`ordination.results`
: Data frame of ordination results, e.g. the $x element from a prcomp object, or the $points element from a metaMDS object in vegan.

`road.map` Identical to the input for the 'a' argument in the dbFD function of the FD package, and to the picante.cdm argument used elsewhere in this package. Thus, this is a matrix containing the abundance of each 'species' in the ordination results. Rows are "sites" and columns are "species". Rather than abundances, the values can simply be presence/absences. Moreover, sites could be species and species could be individuals. See details.

Details

The definition of FDis provided by Laliberte and Legendre (2010) and implemented in the dbFD function of the FD package is geared towards calculating the functional diversity of a community, given a set of species with an array of traits. Another useful way in which FDis might be implemented is as a measure of a species' niche breadth. In this case, ordination.results would be measures (e.g. foraging observations) of multiple individuals of multiple species', and road.map would be a matrix describing which observations belong to which species. The abundances in the matrix in this case would describe how much weight to assign to each individual observation.

Value

Matrix specifying the position of the centroids along each axis.

References

Miller, E. T. 2016. Random thoughts.

Laliberte, E. & P. Legendre. 2010. A distance-based framework for measuring functional diversity from multiple traits. Ecology 91:299-305.

Examples

```
#example of how to calculate the weighted centroids of a series of plots based on the
#traits of a set of species. begin by simulating a phylogeny with a birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#create a log-normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community data matrix, with species as columns and sites as rows
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#simulate two traits, combine into a matrix, then ordinate with a PCA
traits <- evolveTraits(tree)[[2]]
ord <- prcomp(traits)

#the weighted centroids ("average traits") of the species in each plot
exCenter <- centers(ordination.results=ord$x, road.map=cdm)

#visual demonstration of how the function works. plot all of the species in the first
#three sites in the CDM according to their ordinated trait values, color the points
#according to the site (red=site1, purple=site2, blue=site3), and scale the size of the
#point to the abundance of that species in that site. centroids are plotted in the
#same color but given a thick black border.
```

```
plot(ord$x, type="n")
for(i in 1:3)
{
#find the species and their abundances that occur in the relevant plot
focal <- as.matrix(cdm)[i,][as.matrix(cdm)[i,]!=0]
pointColors <- c(rgb(1, 0, 0, 0.6), rgb(0.7, 0, 1, 0.6), rgb(0, 0, 1, 0.6))
for(j in 1:length(focal))
{
location <- ord$x[names(focal)[j],]
points(x=location[1], y=location[2], col=pointColors[i], pch=20, cex=focal[j]/5)
}
points(x=exCenter[i,1], y=exCenter[i,2], pch=21, col="black", lwd=2, cex=2,
bg=pointColors[i])
}
```

---

checkBetaMetrics *Confirm that the metric functions are in suitable format*

Description

Utility function. Creates a list of functions, either those defined in defineMetrics or a named list of metric functions.

Usage

```
checkBetaMetrics(x, new_ = FALSE)
```

Arguments

x	Optional. If not provided, defines the metrics as those in defineBetaMetrics. Else either a character vector or a named list of functions, depending on whether new_ is set to TRUE or FALSE. See calcBetaMetrics.
new_	Whether or not new metrics are being defined on the fly. Default is FALSE. Set to TRUE if a new metric is being used.

Details

A few quick checks to confirm the metric functions are input in suitable format.

Value

A list of functions.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
checkBetaMetrics(names(defineBetaMetrics()))
```

---

checkCDM *Confirm that a CDM will run*

Description

Check that a CDM will work with the dispersalNull model.

Usage

```
checkCDM(picante.cdm)
```

Arguments

picante.cdm	A picante-style community data matrix with sites as rows, and species as columns

Details

It is possible that a CDM is unsuitable for the dispersalNull model. Specifically, if any single grid cell contains more species than are contained in the sum of the remaining grid cells, the model will get stuck in an indefinite loop.

Value

either "pass" or "fail"

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#this CDM should pass the check
checkCDM(cdm)

#this CDM should not pass the check
test <- matrix(nrow=3, ncol=10)
test[1,] <- 1:10
test[2,] <- c(1,1,0,0,0,0,0,0,0,0)
```

```
test[3,] <- c(1,1,0,0,0,0,0,0,0,0)
test <- as.data.frame(test)
names(test)<-paste("s",1:10, sep="")
row.names(test) <- c("cell1","cell2","cell3")
checkCDM(test)
```

---

checkMetrics *Confirm that the metric functions are in suitable format*

Description

Utility function. Creates a list of functions, either those defined in defineMetrics or a named list of metric functions.

Usage

```
checkMetrics(x, new_ = FALSE)
```

Arguments

x	Optional. If not provided, defines the metrics as those in defineMetrics. Else either a character vector or a named list of functions, depending on whether new_ is set to TRUE or FALSE. See calcMetrics.
new_	Whether or not new metrics are being defined on the fly. Default is FALSE. Set to TRUE if a new metric is being used.

Details

A few quick checks to confirm the metric functions are input in suitable format.

Value

A list of functions.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
checkMetrics(names(defineMetrics()))
```

checkNulls *Confirm that the null model functions are in suitable format*

Description

Utility function. Creates a list of null models, either those defined in defineNulls or a named list of null model functions.

Usage

```
checkNulls(x, new_ = FALSE)
```

Arguments

x	Optional. If not provided, defines the nulls as those in defineNulls. Else either a character vector or a named list of functions, depending on whether new_ is set to TRUE or FALSE. See runNulls.
new_	Whether or not new nulls are being defined on the fly. Default is FALSE. Set to TRUE if a new null is being used.

Details

A few quick checks to confirm the null model functions are input in suitable format.

Value

A list of functions.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
checkNulls(names(defineNulls()))
```

checkSimulations *Confirm that the spatial simulation functions are in suitable format*

Description

Utility function. Creates a list of spatial simulations, either those defined in defineSimulations or a named list of simulation functions.

Usage

```
checkSimulations(x, new_ = FALSE)
```

Arguments

x	Optional. If not provided, defines the simulations as those in defineSimulations. Else either a character vector or a named list of functions, depending on whether new_ is set to TRUE or FALSE. See runSimulations.
new_	Whether or not new simulations are being defined on the fly. Default is FALSE. Set to TRUE if a new simulation is being used.

Details

A few quick checks to confirm the spatial simulations are in suitable format.

Value

A list of functions.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
checkSimulations(names(defineSimulations()))
```

`competitionArena` *Simulate competitive exclusion over multiple generations*

Description

Given a simulations.input object, creates an initial random arena, then removes some closely related individuals in the arena, settles individuals based on abundances from a regional species pool, and repeats across the desired number of generations.

Usage

```
competitionArena(simulations.input)
```

Arguments

`simulations.input`
: A prepared simulations.input object from prepSimulations

Details

This function combines the killSome and settleSome functions into a loop that runs for the desired number of generations. If the number of individuals in the arena exceeds 50,000, then the killSomeBig function is invoked. This runs much slower than the default version of the function, but will not use all memory and crash at large numbers of individuals.

Value

A list of 5 elements: the average relatedness in the geographic neighborhood of consideration (appended to any previous values that were fed into the function), the number of individuals killed, the original input regional abundance vector, the new spatial arena, and the dimensions of that arena. On the last iteration, it returns the arena BEFORE settling new individuals randomly.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=2)

test <- competitionArena(prepped)
```

defineBetaMetrics *Output all beta metrics as a list of named functions*

Description

Creates a list of named functions, each of which accept a metrics.input object

Usage

```
defineBetaMetrics()
```

Details

All of the beta metrics we calculated for our manuscript are included in this function. To add additional functions, they can either be defined on the fly or, to permanently include a new metric in all downstream simulations, it can be included here. The function needs to be included with a name, and it must accept a metrics.input as input. If the function needs additional elements not included in that input, then the prepData function must also be revised.

Value

A list of named functions

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
defineBetaMetrics()
```

defineMetrics *Output all metrics as a list of named functions*

Description

Creates a list of named functions, each of which accept a metrics.input object

Usage

```
defineMetrics()
```

Details

All of the functions we calculated for our manuscript are included in this function. To add additional functions, they can either be defined on the fly or, to permanently include a new metric in all downstream simulations, it can be included here. The function needs to be included with a name, and it must accept a metrics.input as input. If the function needs additional elements not included in that input, then the prepData function must also be revised.

Value

A list of named functions

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
defineMetrics()
```

---

defineNulls *Output all null models as a list of named functions*

Description

Creates a list of named functions, each of which accept a nulls.input object

Usage

```
defineNulls()
```

Details

All of the null models we calculated for our manuscript are included in this function. To add additional null model functions, they can either be defined on the fly or to permanently include a new model in all downstream simulations, it can be included here. The function needs to be included with a name, and it must accept a nulls.input object. If the function needs additional elements not included in that input, then the prepNulls function must also be revised.

Value

A list of named functions

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
defineNulls()
```

`defineSimulations` *Output all spatial simulations as a list of named functions*

Description

Creates a list of named functions, each of which accept a simulations.input object

Usage

```
defineSimulations()
```

Details

All of the spatial simulations we calculated for our manuscript are included in this function. To add additional spatial simulations, it can either be defined on the fly or to permanently include a new simulation in all downstream simulations, it can be included here. The function needs to be included with a name, and it must accept a simulations.input object. If the function needs additional elements not included in that input, then the prepSimulations function must also be revised.

Value

A list of named functions

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
defineSimulations()
```

`dispersalNull` *Randomize community data matrix with dispersal null model*

Description

Null model that maintains species richness and settles species in simulated plots with a probability proportional to their abundance in and distance from the plot in question.

Usage

```
dispersalNull(picante.cdm, tree, distances.among, abundance.matters = TRUE,
  abundance.assigned = "directly")
```

Arguments

`picante.cdm`	Picante-style community data matrix with communities/plots/plots/etc as rows and species as columns
`tree`	Phylo object
`distances.among`	A symmetric distance matrix, summarizing the distances among all plots from the cdm.
`abundance.matters`	Default is TRUE. If FALSE, species are sampled from neighboring grid cells with equal probability.
`abundance.assigned`	Default is "directly". If set to "explore", rather than assigning species abundances equal to those in neighboring cells, a normal distribution of standard deviation 1 is created around that abundance, rounded to whole numbers, negative numbers are converted to 1, and then abundance is assigned from this distribution. Note, importantly, that if the cdm is a relative abundance matrix, as opposed to an absolute abundance matrix, if "explore" is chosen, odd results are likely (don't do it). If set to "overall", which may be preferable when abundance.matters is set to FALSE, species' are assigned abundances by drawing from the vector of non-zero abundances from the original cdm.

Details

This function can run quite slowly, as it employs a while loop to discard selected species if they are already contained in the plot being simulated. The function contains an internal check to ensure that it doesn't get stuck in an indefinite while loop. Specifically, it checks that an observed plot does not contain more species than the sum of unique species in the remaining plots. The argument distances.among is flexible, and can relate to e.g., straight-line distances, great-circle distances, and ecological distances. If the argument abundance.matters is set to FALSE, then species' abundances in neighboring grid cells does not influence their probability of settling (only their proximity influences the probability of settling). If this is the case, it is recommended that the argument abundance.assigned be set to "overall".

Value

A matrix with the same dimensions as the input cdm.

References

Miller, E. T. 2016. A new dispersal-informed null model for community ecology shows strong performance. biorxiv.

Examples

```
#set up a matrix to simulate lat/long
coordDF <- matrix(ncol=2, nrow=100)

coordDF[,1] <- runif(n=100, min=40, max=50)
coordDF[,2] <- runif(n=100, min=-130, max=-120)

#convert to data frame, give column names. also give row names such as if the cells had
#names (as they should or there'd be no way to track them)
coordDF <- as.data.frame(coordDF)

row.names(coordDF) <- paste("cell", 1:100, sep="")

names(coordDF) <- c("latitude","longitude")

#calculate the distances among all of these points. in the real program you're going to
#want to calculate great arc distance or whatever it's called
distances <- dist(coordDF, diag=TRUE, upper=TRUE)

#turn it into a symmetric distance matrix
distances <- as.matrix(distances)

#simulate a regional phylogeny of 100 species
tree <- geiger::sim.bdtree(b=1, d=0, stop="taxa", n=100)

#simulate a community data matrix of 100 cells by 100 species. do it 4 times so that
#you can use your simulateComm function and have it span a reasonable range of richness
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm1 <- simulateComm(tree, richness.vector=10:34, abundances=sim.abundances)
cdm2 <- simulateComm(tree, richness.vector=10:34, abundances=sim.abundances)
cdm3 <- simulateComm(tree, richness.vector=10:34, abundances=sim.abundances)
cdm4 <- simulateComm(tree, richness.vector=10:34, abundances=sim.abundances)

#bind these into a list and use dplyr rbind_all to bind together. recast as data frame

cdmList <- list(cdm1, cdm2, cdm3, cdm4)

cdm <- dplyr::rbind_all(cdmList)

cdm <- as.data.frame(cdm)

#fix as necessary manually here (i.e. make sure dimensions are 100 x 100), seems to
```

```
#usually work. then give cell names

row.names(cdm) <- paste("cell", 1:100, sep="")

#fill NAs with 0s.

cdm[is.na(cdm)] <- 0

#not run
#newCDM <- dispersalNull(cdm, tree, distances)
```

---

distMRCA *Calculate plot-level distances to most recent common ancestors*

Description

Given a picante-style community data matrix (sites are rows, species are columns), and a phylogeny, calculate the distances of sets of taxa to their MRCA.

Usage

```
distMRCA(samp, tree, pairwise)
```

Arguments

samp	A picante-style community data matrix with sites as rows, and species as columns.
tree	An ape-style phylogeny.
pairwise	Whether to use the MRCA of all taxa in the sample, or the MRCA of each pairwise comparison in the sample. See details.

Details

Experimental metrics! This function calculates two simple but potentially useful measures. The first, accessed by setting pairwise to FALSE, is the mean branch length between a set of taxa and their most recent common ancestor (MRCA). I have not seen this used in the literature before, but it seems likely I'm wrong. This metric was not tested in our recent Ecography review, but given certain data structures, it seems potentially useful. In other cases, the MRCA will often simply be the root of the tree, and the metric will perhaps be of less use. Large values of the version of distMRCA correspond to taxa with a distant MRCA, while small values correspond to taxa with a more recent MRCA. Given an ultrametric tree, the mean distance between a set of taxa and a single ancestor is of course equal to the distance between one of those taxa and the ancestor. However, in case an ultrametric tree is passed to the function, I do define it as the mean distance between all present taxa and their MRCA. It will throw a warning if a non-ultrametric tree is passed along.

The second measure calculated by this function is accessed by setting pairwise to TRUE. Here, per plot, the metric finds the distance of the MRCA of each pairwise taxon comparison from the root. The value returned per plot is then the mean of these distances. DANGER. Because this second option calculates all pairwise comparisons, the time it takes to run grows exponentially with the

size of the commuity data matrix. For instance, on my personal computer, pairwise distMRCA was calculated in 0.2 seconds for a CDM with 16 plots containing between 10 and 25 species each. However, for a CDM with 100 plots containing between 25 and 55 species, it took 42s. In contrast to the first flavor of this metric, large values of this metric correspond to plots where the taxa present are more recently derived, while small values correspond to plots where the taxa are less recently derived (average common ancestor closer to the root). To make these measures more comparable, it may be better subtract the final values from the total tree height (with caveat about ultrametric tree above). It would also be easy to derive an abundance weighted version of this function. UPDATE. It appears that this second form is yet another (slower) way of deriving the calculation of MPD/PSV.

Value

A vector of distMRCA values.

References

Miller, E. T. 2016. Random thoughts.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

results <- distMRCA(cdm, tree, pairwise=FALSE)
```

---

errorChecker *Wrapper for summarizing randomizations and testing significance of observed metrics*

Description

Given a data frame of observed metrics and a list of randomizations based on different null models, returns a list of data frames summarizing the significance of observed metrics both at the single plot and the entire arena level.

Usage

```
errorChecker(observed, reduced.randomizations, concat.by)
```

Arguments

`observed`	Data frame of observed metric scores, such as from observedMetrics()
`reduced.randomizations`	List of random, reduced results, such as those from reduceRandomizations()
`concat.by`	Whether to concatenate the randomizations by richness, plot or both

Details

This function wraps a number of smaller functions into a useful type I and II error checker. It takes a reduced list of randomizations such as those reduced from metricsNnulls with reduceRandomizations, summarizes the mean, SD, and CI of each metric plus null model either at the richness or plot level, then compares the observed metric scores to those summarized metrics. It return a list with two elements. The first is a list of data frames, where each corresponds to the standardized effect scores of the observed metrics for a given null model. The second is a list of data frames, where each corresponds to whether a given plot deviates beyond CI. For the latter, 0 corresponds to within CI, 1 corresponds to less than the CI, and 2 corresponds to greater than the CI.

Value

A list of lists of data frames.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate a log normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community of varying richness
cdm <- simulateComm(tree, richness.vector=10:13, abundances=sim.abundances)

#run the metrics and nulls combo function
rawResults <- metricsNnulls(tree=tree, picante.cdm=cdm, randomizations=2, cores="seq",
nulls=c("richness","frequency"), metrics=c("richness","NAW_MPD"))

#summarize the results
results <- reduceRandomizations(rawResults)

#calculate the observed metrics from the input CDM
observed <- observedMetrics(tree, cdm, metrics=c("richness", "NAW_MPD"))

test <- errorChecker(observed, results, "richness")
```

---

evolveTraits *Evolve two traits up a tree*

Description

Given a phylogeny, will generate associated trait data for two traits following a Brownian motion evolution model.

Usage

```
evolveTraits(tree, sigma = 0.1)
```

Arguments

tree	A phylogeny.
sigma	The Brownian motion rate parameter.

Details

Evolves two traits independently up phylogeny with Brownian motion evolution process. Sigma is currently set to 0.1 as default.

Value

A list where the first object is a phylogeny with the desired number of species and the second object is a matrix of trait values for those species.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

results <- evolveTraits(tree)
```

---

expectations *Generate expectations for null+metric combinations*

Description

Will generate mean, SD and CI expectations for the desired metric + null combinations.

Usage

```
expectations(picante.cdm, tree, optional.dists = NULL,
  regional.abundance = NULL, distances.among = NULL, randomizations, cores,
  metrics, nulls, concat.by, output.raw = FALSE)
```

Arguments

picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
tree	Phylo object
optional.dists	A symmetric distance matrix can be directly supplied. This option is experimental. Behavior depends on metric being used. If the metric in question relies on the phylogenetic distance matrix from a call to cophenetic(tree), then this optional distance matrix will be inserted instead.
regional.abundance	A character vector in the form "s1, s1, s1, s2, s2, s3, etc". Optional, will be generated from the input CDM if not provided.
distances.among	A symmetric distance matrix, summarizing the distances among all plots from the cdm. Optional, only used by some null models.
randomizations	The number of times the input CDM should be randomized and the metrics calculated across it.
cores	This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize.
metrics	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).
nulls	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).
concat.by	Whether to concatenate the randomizations by richness, plot or both
output.raw	Default is FALSE. Set to TRUE if raw randomized values are preferred (as opposed to summarized mean, SD, CI, etc).

Details

Given a list of desired metrics (which should always include richness) and null models, will generate the expected mean, standard deviation and confidence intervals based on the number of specified randomizations. This function is flexible in that new metrics and nulls can be added and tested with it. By setting output.raw to TRUE, the function can also output raw randomized values as opposed to the summarized values.

Value

A list of data frames, where each data frame corresponds to a given null model, and contains the mean, SD, and CI for each metric for that null model.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate a log normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community of varying richness
cdm <- simulateComm(tree, richness.vector=10:13, abundances=sim.abundances)

#below not run for timing issues on CRAN
test <- expectations(picante.cdm=cdm, tree=tree, optional.dists=NULL,
regional.abundance=NULL, distances.among=NULL, randomizations=3, cores="seq",
nulls="richness", metrics=c("richness", "NAW_MPD"),
concat.by="both", output.raw=FALSE)

#an example of how to explore behavior of a new metric in the metricTester framework
#this "metric" simply calculates the richness of each plot in the CDM
exampleMetric <- function(metrics.input)
{
output <- apply(metrics.input$picante.cdm, 1, lengthNonZeros)
output
}

#test2 <- expectations(picante.cdm=cdm, tree=tree, optional.dists=NULL,
#regional.abundance=NULL, distances.among=NULL, randomizations=3, cores=1,
#nulls="frequency",
#metrics=list("richness"=metricTester:::my_richness, "exampleMetric"=exampleMetric),
#concat.by="both", output.raw=FALSE)
```

FDis *Calculate functional dispersion (FDis)*

Description

Calculate the functional dispersion of clouds of multivariate points.

Usage

```
FDis(ordination.results, road.map)
```

Arguments

`ordination.results`
: Matrix of ordination results, e.g. the $x element from a prcomp object, or the $points element from a metaMDS object in vegan. Could also be raw trait values, but cannot currently handle categorical variables.

`road.map`
: Identical to the input for the 'a' argument in the dbFD function of the FD package, and to the picante.cdm argument used elsewhere in this package. Thus, this is a matrix or data frame containing the abundance of each 'species' in the ordination results. Rows are "sites" and columns are "species". Rather than abundances, the values can simply be presence/absences. Moreover, sites could be species and species could be individuals. See details.

Details

The definition of FDis provided by Laliberte and Legendre (2010) and implemented in the dbFD function of the FD package is geared towards calculating the functional diversity of a community, given a set of species with an array of traits. Another useful way in which FDis might be implemented is as a measure of a species' niche breadth. In this case, ordination.results would be measures (e.g. foraging observations) of multiple individuals of multiple species', and road.map would be a matrix describing which observations belong to which species. The abundances in the matrix in this case would describe how much weight to assign to each individual observation. Regardless of the scale of calculation (either across species within a community or across individuals within a species), this function determines the weighted centroid of each cloud of points and then determines the weighted mean absolute deviation from each centroid.

Value

Named numeric with FDis values.

References

Miller, E. T. 2016. Random thoughts, though please cite metricTester via our 2016 Ecography paper.

Laliberte, E. & P. Legendre. 2010. A distance-based framework for measuring functional diversity from multiple traits. Ecology 91:299-305.

Examples

```
#example of how to calculate the FDis of a series of plots based on the trait values
#of a set of species. begin by simulating a phylogeny with a birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#create a log-normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community data matrix, with species as columns and sites as rows
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#simulate two traits and combine into a matrix. because species are sometimes absent
```

```
#from the cdm, also exclude any species from the trait data frame that are not in the
#cdm (to avoid errors), then ordinate with a PCA
traits <- evolveTraits(tree)[[2]]
traits <- traits[row.names(traits) %in% colnames(cdm),]
ord <- prcomp(traits)

#the FDis of the species in each plot
FDis(ordination.results=ord$x, road.map=cdm)

#now an example of how to calculate the FDis of a series of species based on the trait
#values of a set of individuals. begin by simulating trait data for a series of
#individuals. to illustrate the point, simulate varying numbers of individuals per
#species, and where there are varying degrees of variance in traits per species.
traits2 <- data.frame(trait1=c(rnorm(n=30, sd=1), rnorm(n=60, sd=2), rnorm(n=120,sd=4)),
trait2=c(rnorm(n=30, sd=1), rnorm(n=60, sd=2), rnorm(n=120, sd=4)))

#ordinate the traits. could readily use another ordination here, e.g. nmds with gower
ord2 <- prcomp(traits2)

#create a road.map where species are rows and individual observations are columns. the
#first 30 observations belong to sp1, the following 60 to sp2, and the following 120
#to sp3.
cdm2 <- matrix(nrow=3, ncol=dim(traits2)[1], 0)
colnames(cdm2) <- row.names(traits2)
row.names(cdm2) <- c("sp1", "sp2", "sp3")
cdm2[1,1:30] <- 1
cdm2[2,31:90] <- 1
cdm2[3,91:210] <- 1

#the FDis of each species (i.e. niche breadth)
FDis(ordination.results=ord2$x, road.map=cdm2)
```

`filteringArena` *Simulate a community assembled according to habitat filtering*

Description

Given a simulations.input object, will create an arena settled according to habitat filtering rules (and parameters defined by prepSimulations).

Usage

```
filteringArena(simulations.input)
```

Arguments

`simulations.input`
: A prepared simulations.input object from prepSimulations

Details

This is the habitat filtering simulation that was used in our paper (reference below). In short, species have phylogenetically conserved spatial preferences, and individuals of those species are settled near that preferred location with a controllable degree of variation. Species' spatial preferences are smoothed to a uniform distribution. Thus, individuals are fairly evenly distributed throughout the simulated arena.

Value

A list of 3 elements: the original input regional abundance vector, the new spatial arena, and the dimensions of that arena.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=2)

test <- filteringArena(prepped)
```

---

killSome *Remove most closely related individuals*

Description

Given a phylogenetic tree, a spatial arena of individuals with species identities, and arguments for the desired distance and percent removed, removes some of the most closely related individuals in the arena.

Usage

```
killSome(tree, arena.output, max.distance, proportion.killed)
```

Arguments

`tree`	Phylo object
`arena.output`	A spatial arena with three columns: individuals (the species ID), X (the x axis location of that individual), and Y (the y axis location). The arena.output actually needs a number of other elements in order for later functions to work properly, so any modifications to the code should take note of this.

`max.distance` The geographic distance within which geographically neighboring indivduals should be considered to influence the individual in question.

`proportion.killed`
The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half).

Details

This function identifies individuals in the most genetically clustered geographic neighborhoods, continues on to identify the most closely related individual to a focal individual, and randomly chooses whether to remove that individual or the focal individual. It expects a list with a number of additional elements beyond the arena (currently, the mean genetic relatedness of geographic neighborhoods, a vector of regional abundance [where each element is a species name, repeated as many times as is present in pool], and the dimensions of the arena).

Value

A list of 5 elements: the average relatedness in the geographic neighbordhood of consideration (appended to any previous values that were fed into the function), the number of individuals killed, the original input regional abundance vector, the new spatial arena, and the dimensions of that arena.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=5)

#use the competition simulation
positions <- competitionArena(prepped)

#in normal use, these parameters will be carried down from the simulations.input object
new.arena <- killSome(tree, arena.output=positions, max.distance=50,
proportion.killed=0.2)

#look at how number of individuals in arena changes
dim(positions$arena)
dim(new.arena$arena)
```

`killSomeBig` *Remove most closely related individuals for large arenas*

Description

Given a phylogenetic tree, a large spatial arena of individuals with species identities, and arguments for the desired distance and percent removed, removes some of the most closely related individuals in the arena.

Usage

```
killSomeBig(tree, arena.output, max.distance, proportion.killed)
```

Arguments

`tree`	Phylo object
`arena.output`	A spatial arena with three columns: individuals (the species ID), X (the x axis location of that individual), and Y (the y axis location). The arena.output actually needs a number of other elements in order for later functions to work properly, so any modifications to the code should take note of this.
`max.distance`	The geographic distance within which geographically neighboring indivduals should be considered to influence the individual in question.
`proportion.killed`	The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half).

Details

This function identifies individuals in the most genetically clustered geographic neighborhoods, continues on to identify the most closely related individual to a focal individual, and randomly chooses whether to remove that individual or the focal individual. It expects a list with a number of additional elements beyond the arena (currently, the mean genetic relatedness of geographic neighborhoods, a vector of regional abundance [where each element is a species name, repeated as many times as is present in pool], and the dimensions of the arena). This function is invoked if the number of individuals in the arena exceeds 50,000. It was originally intended only for arenas with large numbers of individuals, but it ends up being faster than the original version of the function for all but the smallest arenas. That said, if the local area within the arena that the function considers does not contain any individuals, then it will fail. To overcome this, either a larger max.distance would need to be set, or more individuals should be settled in the arena, or the original version of the function should be invoked by modifying the code in competitionArena(). Note, however, that if there are more than e.g. 50,000 individuals total in the arena, the original version of the function should never be used, or all memory in the computer will be used and it will crash.

Value

A list of 5 elements: the average relatedness in the geographic neighbordhood of consideration (appended to any previous values that were fed into the function), the number of individuals killed, the original input regional abundance vector, the new spatial arena, and the dimensions of that arena.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=1,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=5)

#use the random simulation to generate a starting arena
random <- randomArena(prepped)

#in normal use, these parameters will be carried down from the simulations.input object
new.arena <- killSomeBig(tree, arena.output=random, max.distance=50,
proportion.killed=0.2)

#look at how number of individuals in arena changes
dim(random$arena)
dim(new.arena$arena)
```

---

lengthNonZeros *Calculate the species richness of a vector from a CDM*

Description

Given a vector of abundances or presence/absences from a community data matrix, will calculate the species richness of that vector.

Usage

```
lengthNonZeros(input.vector)
```

Arguments

input.vector	A vector from a community data matrix of abundances.

Details

An internal function to calculate richness of a cdm.

Value

A named vector of species richness.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#note that with this example, each community in the cdm will be labeled by its richness
apply(cdm, 1, lengthNonZeros)
```

`linker` *Run spatial simulations, null and metric calculations to test metric + null performance*

Description

This function wraps a number of wrapper functions into one big metric + null tester function. Only a single test is performed, with results saved into memory.

Usage

```
linker(no.taxa, arena.length, mean.log.individuals, length.parameter,
  sd.parameter, max.distance, proportion.killed, competition.iterations,
  no.plots, plot.length, concat.by, randomizations, cores, simulations, nulls,
  metrics)
```

Arguments

`no.taxa`	The desired number of species in the input phylogeny
`arena.length`	A numeric, specifying the length of a single side of the arena
`mean.log.individuals`	Mean log of abundance vector from which species abundances will be drawn
`length.parameter`	Length of vector from which species' locations are drawn. Large values of this parameter dramatically decrease the speed of the function but result in nicer looking communities
`sd.parameter`	Standard deviation of vector from which species' locations are drawn
`max.distance`	The geographic distance within which neighboring indivduals should be considered to influence the individual in question

`proportion.killed`
: The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half)

`competition.iterations`
: Number of generations over which to run competition simulations

`no.plots`
: Number of plots to place

`plot.length`
: Length of one side of desired plot

`concat.by`
: Whether to concatenate the randomizations by richness, plot or both

`randomizations`
: The number of randomized CDMs, per null, to generate. These are used to compare the significance of the observed metric scores.

`cores`
: The number of cores to be used for parallel processing.

`simulations`
: Optional. If not provided, defines the simulations as all of those in defineSimulations. If only a subset of those simulations is desired, then simulations should take the form of a character vector corresponding to named functions from defineSimulations. The available simulations can be determined by running names(defineSimulations()). Otherwise, if the user would like to define a new simulation on the fly, the argument simulations can take the form of a named list of new functions (simulations).

`nulls`
: Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).

`metrics`
: Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function wraps a number of other wrapper functions into one big metric + null performance tester function. Only a single test is performed, with results saved into memory. To perform multiple complete tests, use the multiLinker function, which saves results to file.

Value

A list of lists of data frames. The first level of the output has one element for each simulation. The second level has one element for each null model. Each of these elements is a list of two data frames, one that summarizes the plot-level significance and another and arena-level significance.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#below not run for timing issues on CRAN
#system.time(test <- linker(no.taxa=50, arena.length=300, mean.log.individuals=2,
#length.parameter=5000, sd.parameter=50, max.distance=30, proportion.killed=0.2,
#competition.iterations=3, no.plots=15, plot.length=30, concat.by="richness",
#randomizations=3, cores="seq",
#nulls=c("richness", "frequency")))
```

`makeCDM` *Wrapper for creating a CDM from a spatial simulation result*

Description

Given the results of a single spatial simulation, and a desired number of plots and the length of one side of each plot, will place the plots down and output a CDM. Importantly, also carries along the regional abundance vector from the spatial simulation results if one was included.

Usage

```
makeCDM(single.simulation, no.plots, plot.length)
```

Arguments

`single.simulation`	The results of a single spatial simulation, e.g. a call to randomArena
`no.plots`	The desired number of plots in the final CDM
`plot.length`	The length of one side of each plot

Details

Just a simple wrapper function to quickly turn spatial simulations into CDMs for subsequent analysis.

Value

A list with the regional abundance from the single simulation result, if it included such a result, or the results of a call to abundanceVector() if not. The list also includes the CDM based on the parameters (number and size of plots) provided.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

competition <- competitionArena(prepped)

test <- makeCDM(competition, 15, 30)
```

`metricPerformance` *Summarize metric performance of a series of summarized simulation results*

Description

Summarizes metric performance after reading in and testing simulation results with either sesIndiv or plotOverall.

Usage

```
metricPerformance(summarized.results, nulls, concat.by = "both")
```

Arguments

`summarized.results`	The results of a call to sesIndiv() or plotOverall(). If from plotOverall, the results must include the standard three spatial simulations (random, filtering, competition), but must contain no other spatial simulations.
`nulls`	By default, this function summarizes metric performance over all tested null models. Alternatively, user can supply a vector of null model names to summarize the results over.
`concat.by`	Default is "both". Meaning that if both plot-level and richness-level summary statistics are available, then performance will be average over both types of results. Alternatively (and perhaps preferably), either "plot" or "richness" can be provided and, assuming that concatenation option was specified in the multiLinker runs, performance results will be summarized just over that specified option.

Details

If an overall picture of metric performance is desired, this function can provide it. It can also be used to summarize metric performance over a specific subset of simulations, null models, and concatenation options. If provided with the results of a call to plotOverall, the options are more limited. Currently, if provided with such a result, the assumption is that there are three spatial simulations,

"random", "filtering", and "competition". It then assumes that any clustered or overdispersed plots for the random simulation, or any overdispersed or clustered for the filtering or competition simulations, respectively, count as typeI errors. It assumes that any plots that are not clustered or overdispersed for the filtering or competition simulations, respectively, count as typeII errors.

Value

A data frame of summarized results

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- sesIndiv(results)
#examp <- metricPerformance(summ)
```

---

metricsNnulls *Parallelized function that calculates metrics on randomized matrices*

Description

This function sends out jobs to as many cores as are specified. Each randomizes the input CDM according to all defined null models, then calculates each observed metric on each randomized matrix.

Usage

```
metricsNnulls(tree, picante.cdm, optional.dists = NULL,
  regional.abundance = NULL, distances.among = NULL, randomizations = 2,
  cores = "seq", nulls, metrics)
```

Arguments

`tree`	Phylo object
`picante.cdm`	A picante-style community data matrix with sites as rows, and species as columns
`optional.dists`	A symmetric distance matrix can be directly supplied. This option is experimental. Behavior depends on metric being used. If the metric in question relies on the phylogenetic distance matrix from a call to cophenetic(tree), then this optional distance matrix will be inserted instead.
`regional.abundance`	A character vector in the form "s1, s1, s1, s2, s2, s3, etc". Optional, will be generated from the input CDM if not provided.

`distances.among`
: A symmetric distance matrix, summarizing the distances among all plots from the cdm. Optional, only used by some null models.

`randomizations`
: The number of times the input CDM should be randomized and the metrics calculated across it.

`cores`
: This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially.

`nulls`
: Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).

`metrics`
: Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function sends out jobs to as many cores as are specified. Each randomizes the input CDM according to all defined null models, then calculates each observed metric on each randomized matrix.

Value

A list of lists of vectors. The first level has as many elements as there are randomizations. The second level has one list for each null model. Each element of this second level is a named vector corresponding to the calculated metric at each plot.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

rawResults <- metricsNnulls(tree, cdm, randomizations=3,
nulls=c("richness", "frequency"))
```

`modifiedMPD` *Calculate different versions of abundance-weighted MPD*

Description

Given a picante-style community data matrix (sites are rows, species are columns), a phylogenetic distance matrix, and a desired method of abundance-weighting, will calculate MPD.

Usage

```
modifiedMPD(samp, dis, abundance.weighted = FALSE)
```

Arguments

`samp`	A picante-style community data matrix with sites as rows, and species as columns
`dis`	Phylogenetic distance matrix
`abundance.weighted`	One of either "FALSE", "interspecific", "intraspecific", or "complete"

Details

See accompanying publication for details. Non-abundance-weighted and interspecific and intraspecific methods are equivalent to those previously described by Clarke & Warwick.

Value

A vector of MPD values, calculated according to the abundance-weighted method specified.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

dists <- ape::cophenetic.phylo(tree)

results <- modifiedMPD(cdm, dists, abundance.weighted = "interspecific")
```

MRD *Calculate mean root distance*

Description

Given a picante-style community data matrix (sites are rows, species are columns), and a phylogeny, calculate the mean root distance of the set of taxa in each site.

Usage

```
MRD(samp, tree, abundance.weighted)
```

Arguments

samp	A picante-style community data matrix with sites as rows, and species as columns.
tree	An ape-style phylogeny.
abundance.weighted	Whether to weight the calculation by the abundance of a given species in a given plot.

Details

Mean root distance (MRD) as originally formulated by Kerr & Currie (1999) defined MRD as the mean number of nodes between a set of taxa and the root. It is not clear to me whether the number of nodes includes the tip itself. In other words, should a species connected directly to the root of the tree be considered to be separated by zero or one nodes from the root? I have chosen to define it as one, but am open to changing it. This definition emphasizes that "one" speciation event has occurred along that branch since the origin of the clade (all caveats about extinction and phylogenetic sampling aside). The abundance-weighted form of this calculation takes a row-wise (plot-wise) weighted-mean where the values are a species' node-distance to the root, and the weights are a species' abundance in the input community data matrix. I ran a quick test to see whether abundance-weighted MRD might be equal to IAC of Cadotte et al. (2010), and it does not seem to be. Therefore its utility is unknown (as is that of non-abundance weighted MRD).

Value

A vector of MRD values.

References

Kerr, J. T. and D. J. Currie. 1999. The relative importance of evolutionary and environmental controls on broad-scale patterns of species richness in North America. Ecoscience 6:329-337.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

results <- MRD(cdm, tree, abundance.weighted=FALSE)
```

`multiCDM` *Wrapper for deriving CDMs from the results of multiple spatial simulations*

Description

Given the results of a call to runSimulations(), this function places plots down randomly (though identically across simulations).

Usage

```
multiCDM(simulations.result, no.plots, plot.length)
```

Arguments

`simulations.result`	List of data frames of three columns: "individuals", "X", and "Y"
`no.plots`	Number of plots to place
`plot.length`	Length of one side of desired plot

Details

Both the size and number of plots are determined by the user. A conservative check (perhaps overly so) is in place to ensure the function doesn't get stuck looking for solutions for how to randomly place non-overlapping plots. Either decreasing the number or size of plots is necessary if this throws an error.

Value

A list of data frames.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=1000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=5)

#run the spatial simulations
arenas <- runSimulations(prepped)

#derive CDMs. plots are placed in the same places across all spatial simulations.
#density of individuals per arena is low enough in this example that sometimes all
#plots contain < 2 species, and are cut, causing an error. this not run so as not to
#throw errors on CRAN
#cdms <- multiCDM(arenas, no.plots=10, plot.length=20)
```

`multiLinker` *Run multiple simulations and calculations to test metric + null performance*

Description

This function runs multiple iterations of the linker function, saving results to file.

Usage

```
multiLinker(no.taxa, arena.length, mean.log.individuals, length.parameter,
  sd.parameter, max.distance, proportion.killed, competition.iterations,
  no.plots, plot.length, concat.by, randomizations, cores, iterations, prefix,
  simulations, nulls, metrics)
```

Arguments

`no.taxa`	The desired number of species in the input phylogeny
`arena.length`	A numeric, specifying the length of a single side of the arena
`mean.log.individuals`	Mean log of abundance vector from which species abundances will be drawn
`length.parameter`	Length of vector from which species' locations are drawn. Large values of this parameter dramatically decrease the speed of the function but result in nicer looking communities
`sd.parameter`	Standard deviation of vector from which species' locations are drawn
`max.distance`	The geographic distance within which neighboring indivduals should be considered to influence the individual in question

`proportion.killed`
: The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half)

`competition.iterations`
: Number of generations over which to run competition simulations

`no.plots`
: Number of plots to place

`plot.length`
: Length of one side of desired plot

`concat.by`
: Whether to concatenate the randomizations by richness, plot or both

`randomizations`
: The number of randomized CDMs, per null, to generate. These are used to compare the significance of the observed metric scores.

`cores`
: The number of cores to be used for parallel processing.

`iterations`
: The number of complete tests to be run. For instance, 1 iteration would be considered a complete cycle of running all spatial simulations, randomly placing plots in the arenas, sampling the contents, creating a community data matrix, calculating observed metric scores, then comparing these to the specified number of randomizations of the original CDMs.

`prefix`
: Optional character vector to affix to the output RData file names, e.g. "test1".

`simulations`
: Optional. If not provided, defines the simulations as all of those in defineSimulations. If only a subset of those simulations is desired, then simulations should take the form of a character vector corresponding to named functions from defineSimulations. The available simulations can be determined by running names(defineSimulations()). Otherwise, if the user would like to define a new simulation on the fly, the argument simulations can take the form of a named list of new functions (simulations).

`nulls`
: Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls).

`metrics`
: Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

This function wraps a number of other wrapper functions into one big metric + null performance tester function. Unlike the basic linker function, multiple tests can be run, with results saved as RDS files.

Value

A list of lists of data frames. The first level of the output has one element for each simulation. The second level has one element for each null model. Each of these elements is a list of two data frames, one that summarizes the plot-level significance and another and arena-level significance.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#system.time(multiLinker(no.taxa=50, arena.length=300, mean.log.individuals=3.2,
#length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.3,
#competition.iterations=2, no.plots=20, plot.length=30, concat.by="richness",
#randomizations=3, cores="seq", iterations=2, prefix="test",
#nulls=c("richness", "frequency")))
```

---

nullPerformance *Summarize null model performance of a series of summarized simulation results*

Description

Summarizes null performance after reading in and testing simulation results with either sesIndiv or plotOverall.

Usage

```
nullPerformance(summarized.results, metrics, concat.by = "both")
```

Arguments

summarized.results	The results of a call to sesIndiv() or plotOverall(). If from plotOverall, the results must include the standard three spatial simulations (random, filtering, competition), but must contain no other spatial simulations.
metrics	By default, this function summarizes null performance over all tested metrics. Alternatively, user can supply a vector of metric names to summarize the results over.
concat.by	Default is "both". Meaning that if both plot-level and richness-level summary statistics are available, then performance will be average over both types of results. Alternatively (and perhaps preferably), either "plot" or "richness" can be provided and, assuming that concatenation option was specified in the multiLinker runs, performance results will be summarized just over that specified option.

Details

If an overall picture of null performance is desired, this function can provide it. It can also be used to summarize null performance over a specific subset of simulations, metrics, and concatenation options. If provided with the results of a call to plotOverall, the options are more limited. Currently, if provided with such a result, the assumption is that there are three spatial simulations, "random", "filtering", and "competition". It then assumes that any clustered or overdispersed plots for the random simulation, or any overdispersed or clustered for the filtering or competition simulations, respectively, count as typeI errors. It assumes that any plots that are not clustered or overdispersed for the filtering or competition simulations, respectively, count as typeII errors.

Value

A data frame of summarized results

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- sesIndiv(results)
#examp <- nullPerformance(summ)
```

---

observedBetaMetrics *Wrapper for prepping and calculating observed beta metrics*

Description

Given a cdm and phylogeny, this function preps the data and calculates beta metrics of thee user's choice

Usage

```
observedBetaMetrics(tree, picante.cdm, metrics)
```

Arguments

tree	Phylo object
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
metrics	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineBetaMetrics()). If the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

A simple wrapper function to quickly prep data and calculate observed metrics.

Value

A data frame with the species richness, total abundance from the CDM, and calculated phylogenetic community structure beta metrics for all input plots from the CDM.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

positions <- randomArena(prepped)

tempCDM <- makeCDM(positions, 15, 30)

results <- observedBetaMetrics(tree=tree, picante.cdm=tempCDM$picante.cdm)

#example of how to pass specific metrics to be calculated. not run

#results <- observedBetaMetrics(tree=tree, picante.cdm=tempCDM$picante.cdm,
#metrics=c("richness", "Ist"))
```

---

observedMetrics *Wrapper for prepping and calculating observed metrics*

Description

Given a cdm and phylogeny, this function preps the data and calculates metrics of the user's choice

Usage

```
observedMetrics(tree, picante.cdm, metrics)
```

Arguments

tree	Phylo object
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
metrics	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()). Otherwise, if the user would like to define a new metric on the fly, the argument can take the form of a named list of new functions (metrics).

Details

A simple wrapper function to quickly prep data and calculate observed metrics.

Value

A data frame with the species richness and calculated phylogenetic community structure metrics for all input plots from the CDM.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

positions <- randomArena(prepped)

tempCDM <- makeCDM(positions, 15, 30)

results <- observedMetrics(tree=tree, picante.cdm=tempCDM$picante.cdm)

#example of how to pass specific metrics to be calculated (always use at least
#richness). not run

#results <- observedMetrics(tree=tree, picante.cdm=tempCDM$picante.cdm,
#metrics=c("richness","PSV"))
```

phyloField *Calculate a species' phylogenetic field*

Description

Calculate the phylogenetic relatedness of a species to those it occurs with.

Usage

```
phyloField(tree, picante.cdm, metric)
```

Arguments

tree	Phylo object. Will be pruned to species actually in the cdm.
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
metric	Phylogenetic metric of choice (see details)

Details

This function is being deprecated. This and the rest of the first generation of field functions are being replaced by a two-step process akin to the calcMetrics set of functions. The user first preps the data with a prep function, then runs the desired metrics and nulls over the prepped object. This allows sets of metrics to be calculated over the same randomized matrix, rather than having to repeatedly generate the same random matrix for each metric. Currently this is only programmed to use either non-abundance-weighted mean pairwise or interspecific abundance-weighted mean pairwise phylogenetic distance. Importantly, we are in the process of generalizing the phylo & trait field functions, so they operate more like the calcMetrics functions. In other words, the user will prep their data first, then choose which metrics to calculate and the function will detect whether to calculate phylo or trait fields based on the inputs. Take note of this, as code using the current forms of these functions is liable to break when these updates are made.

Value

Named vector of species' phylogenetic fields.

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate log-normal abundances
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1
```

```
#simulate a community data matrix with these inputs
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#example phylogenetic field calculations
exampleField <- phyloField(tree=tree, picante.cdm=cdm, metric="naw.mpd")
```

`plotContents` *Identify individuals contained within a plot*

Description

Given a spatially explicit data frame of individual locations in a simulated arena, and the bounds of a series of plots, identifies the contents of each plot.

Usage

```
plotContents(arena, plotPlacer.results)
```

Arguments

`arena`	Data frame of three columns: "individuals", "X", and "Y"
`plotPlacer.results`	The results of a call to plotPlacer. A list with two elementers. The first is a matrix of X Y coordinates of plots. The second is a symmetrical distance matrix summarizing the distances among these plots.

Details

Takes a data frame like that returned from filteringArena(), and a matrix like that returned from plotPlacer(), and returns the resulting community data matrix such as might be generated by someone surveying a forest plot. Plots with < 2 species are excluded from the CDM.

Value

A matrix with species as rows and plots as columns.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

temp <- evolveTraits(tree)

phydistmatrix <- ape::cophenetic.phylo(temp[[1]])

#define a color for each species
cols <- plotrix::color.scale(x=1:nrow(phydistmatrix),
cs1=c(0.2,0.4,0.8), cs2=c(0,0.5,0.8), cs3=c(1,0.5,0))

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

singleArena <- filteringArena(prepped)

#plot the arena. don't close the window
plot(singleArena$arena$X, singleArena$arena$Y, pch=20, cex=1, xlim=c(0,300),
ylim=c(0,300), col=cols[singleArena$arena$individuals])

boundResults <- plotPlacer(no.plots=10, arena.length=300, plot.length=50)

plotPlotter(boundResults$plot.bounds)

#return a CDM in picante format
cdm <- plotContents(singleArena$arena, boundResults)
```

`plotOverall` *Overall per simulation-null-metric plot test*

Description

This function provides one of many ways of summarizing and considering simulation results.

Usage

```
plotOverall(simulation.list)
```

Arguments

`simulation.list`
: A summarized results list such as one output from reduceResults(). See examples.

Details

This function provides one way of summarizing and considering simulation results. It takes as input a vector of 0s, 1s and 2s (corresponding to within, less, and greater than the 95% CIs, respectively) for all plots from a given simulation-null-metric combination, and determines how many plots overall deviated beyond expectations. A number of utility functions used for that goal are also defined but not exported in this function.

Value

A data frame summarizing the total number of plots tested and how many of these deviated above (significantly overdispersed) or below (significantly clustered) the 95% CI for each simulation, null, metric combination.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- reduceResults(results)
#test <- plotOverall(summ$plot)
```

---

plotPlacer *Randomly place plots in arena*

Description

Given a desired number of plots, the arena size, and the plot size, will attempt to place plots down in a non-overlapping fashion

Usage

```
plotPlacer(no.plots, arena.length, plot.length)
```

Arguments

`no.plots`	Number of plots to place
`arena.length`	Length of one side of arena
`plot.length`	Length of one side of desired plot.

Details

Places plots down in non-overlapping fashion according to parameters supplied. Because this would run indefinitely if unacceptable parameters were supplied, a conservative check is implemented to "ensure" the function does not get stuck. If unacceptable parameters are supplied, will return an arena and a smaller total sampling area will need to be defined.

Value

A matrix with the X & Y coordinates of the four corners of each plot placed

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
boundResults <- plotPlacer(no.plots=10, arena.length=300,
plot.length=50)
```

---

plotPlotter *Plot simulated plots in arena*

Description

Given a matrix of plot bounds, plots the plots in an already plotted, simulated arena.

Usage

```
plotPlotter(plot.bounds)
```

Arguments

plot.bounds	Matrix of plot bounds

Details

Plots plots as defined by the supplied matrix, e.g. a call to plotPlacer. An active plot with the simulated arena needs to already be open, see example.

Value

Plotted plots

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

temp <- evolveTraits(tree)

phydistmatrix <- ape::cophenetic.phylo(temp[[1]])

#define a color for each species
cols <- plotrix::color.scale(x=1:nrow(phydistmatrix),
cs1=c(0.2,0.4,0.8), cs2=c(0,0.5,0.8), cs3=c(1,0.5,0))

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

positions <- filteringArena(prepped)

#plot the arena. don't close the window
plot(positions$arena$X, positions$arena$Y, pch=20, cex=1, xlim=c(0,300), ylim=c(0,300),
col=cols[positions$arena$individuals])

bounds <- plotPlacer(no.plots=10, arena.length=300,
plot.length=50)$plot.bounds

plotPlotter(bounds)
```

---

plotTest *Calculate if single, observed metrics deviate beyond expectations*

Description

Given a table of results, where means, SDs, and CIs are bound to the observed scores at the corresponding richness or plot, this function calculates whether each observed score is significantly less or ore than expected at that plot or richness.

Usage

```
plotTest(results.table, concat.by)
```

Arguments

`results.table`	Data frame of observed metrics with expected mean, SD and CI bound in. See example
`concat.by`	Whether to concatenate results by richness, plot or both. If richness, observed scores are compared to all randomized scores where the plot had the corresponding richness. If plot, observed scores (e.g. those from plot 1) are compared to all randomized plot 1 scores. If both, both are run and each is saved as a separate data frame in a single list.

Details

Given a table of results, where means, SDs, and CIs are bound to the observed scores at the corresponding richness or plot, this function returns 0, 1, or 2, corresponding to not significant, significantly clustered, and significantly overdispersed. Previously the metrics being passed to the function needed to be explicitly specified, but the function now attempts to determine the names of the metrics via the results.table input.

Value

A data frame of 0s, 1s, and 2s.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate a log normal abundance distribution
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community of varying richness
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#below not run for example timing issues on CRAN

#run the metrics and nulls combo function
rawResults <- metricsNnulls(tree=tree, picante.cdm=cdm, randomizations=2, cores="seq",
nulls=c("richness","frequency"), metrics=c("richness","NAW_MPD"))

#reduce the randomizations to a more manageable format
reduced <- reduceRandomizations(rawResults)

#calculate the observed metrics from the input CDM
observed <- observedMetrics(tree, cdm, metrics=c("richness","NAW_MPD"))

#summarize the means, SD and CI of the randomizations
summarized <- lapply(reduced, summaries, concat.by="richness")

#merge the observations and the summarized randomizations to facilitate significance
#testing
merged <- lapply(summarized, merge, observed)

#calculate the standardized scores of each observed metric as compared to the richness
#null model randomization.
plotTest(merged$richness, "richness")

#do the same as above but across all null models
```

```
#temp <- lapply(1:length(merged), function(x) plotTest(merged[[x]], "richness"))
```

---

prepData *Prep data to test phylogenetic community structure metrics*

Description

Given a phylo object, and a picante-style community data matrix (sites are rows, species are columns), prepare data for analysis.

Usage

```
prepData(tree, picante.cdm, optional.dists = NULL)
```

Arguments

tree	Phylo object.
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
optional.dists	A symmetric distance matrix can be directly supplied. This option is experimental. Behavior depends on metric being used. If the metric in question relies on the phylogenetic distance matrix from a call to cophenetic(tree), then this optional distance matrix will be inserted instead.

Details

Returns a named list with three elements: the original phylogenetic tree phylogenetic distances among species, and the original picante-style CDM.

Value

An object of class metrics.input

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

prepped <- prepData(tree, cdm)
```

prepFieldData *Prep data to to calculate phylogenetic fields*

Description

Given a phylo object, and a picante-style community data matrix (sites are rows, species are columns), prepare data for calculation of phylogenetic or trait fields.

Usage

```
prepFieldData(tree, dists, picante.cdm)
```

Arguments

tree	Phylo object.
dists	A symmetric distance matrix can be directly supplied. By doing this, the user is implying that trait fields should be calculated.
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns

Details

Returns a named list with four elements: the original phylogenetic tree phylogenetic distances among species, the original picante-style CDM, and an argument, specifying whether the phylogenetic or trait field should be calculated. This is determined by the inputs. If a tree is supplied, the phylogenetic fields will be calculated. If a distance matrix is supplied, the trait fields will be calculated. Importantly, note that some metrics require a tree for calculation. These metrics include PSV, PSC, PD, and QE. If a trait distance matrix is supplied, and these metrics are called, the distances will be automatically coerced into a dendrogram via the hclust function and conversion to an ape phylogeny.

Value

An object of class field.input

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

prepped <- prepFieldData(tree=tree, picante.cdm=cdm)
```

prepNulls *Prep data for null randomizations*

Description

Given a phylo object, a picante-style community data matrix (sites are rows, species are columns), and an optional vector of regional abundance, prepare data for randomizations.

Usage

```
prepNulls(tree, picante.cdm, regional.abundance = NULL,
  distances.among = NULL)
```

Arguments

tree	Phylo object
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns
regional.abundance	A character vector in the form "s1, s1, s1, s2, s2, s3, etc". Optional, will be generated from the input CDM if not provided.
distances.among	An optional symmetric distance matrix describing the distances among plots/etc, for use with null models like the dispersal null.

Details

Returns a named list with four elements: the original phylogenetic tree, the original picante-style CDM, a spacodi-style CDM, and vector of regional abundance.

Value

A list of class nulls.input

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

prepped <- prepNulls(tree, cdm)
```

`prepSimulations` *Prep data for spatial simulations*

Description

Given the required parameters for defined spatial simulations, will prepare an object of class simulations.input for actual simulation.

Usage

```
prepSimulations(tree, arena.length, mean.log.individuals, length.parameter,
  sd.parameter, max.distance, proportion.killed, competition.iterations)
```

Arguments

`tree`	Phylo object
`arena.length`	A numeric, specifying the length of a single side of the arena
`mean.log.individuals`	Mean log of abundance vector from which species abundances will be drawn
`length.parameter`	Length of vector from which species' locations are drawn. Large values of this parameter dramatically decrease the speed of the function but result in nicer looking communities
`sd.parameter`	Standard deviation of vector from which species' locations are drawn
`max.distance`	The geographic distance within which neighboring indivduals should be considered to influence the individual in question
`proportion.killed`	The percent of individuals in the total arena that should be considered (as a proportion, e.g. 0.5 = half)
`competition.iterations`	Number of generations over which to run competition simulations

Details

This function preps the input for any of the spatial simulations as defined in defineSimulations. If additional parameters are ever required for those simulations, they would have to be added as additional arguments here.

Value

A prepared simulations.input object

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)
```

---

pscCorr *Calculate corrected PSC*

Description

Given a phylo object and a picante-style community data matrix (sites are rows, species are columns), calculated corrected phylogenetic species clustering

Usage

```
pscCorr(samp, tree)
```

Arguments

samp	A picante-style community data matrix with sites as rows, and species as columns
tree	Phylo object

Details

Returns the inverse of psc as defined in picante

Value

A data frame of correctly calculated PSC values, with associated species richness and name of all communities in input cdm

References

Helmus, M.R., T.J. Bland, C.K. Williams, & A.R. Ives. 2007. Phylogenetic measures of biodiversity. The American Naturalist. 169:E69-E83.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

results <- pscCorr(samp=cdm, tree=tree)
```

randomArena *Generate a random spatial arena*

Description

Given a simulations.input object, will create a randomly settled arena according to the parameters in the simulations.input object.

Usage

```
randomArena(simulations.input)
```

Arguments

simulations.input	A prepared simulations.input object from prepSimulations

Details

This function uses the log-normal regional abundance distribution to randomly assign abundances to species in the tree. It then draws from this regional abundance distribution to settle individuals at random in the landscape.

Value

A list of 4 elements: the mean of the genetic distance matrix of the input phylogeny, the regional abundance vector (where each element is a species name, repeated as many times as is present in pool), the spatial arena, and the dimensions of that arena.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=2)

test <- randomArena(prepped)
```

`readIn` *Read in the results of multiple metric/null/simulation tests*

Description

This function is used to read all .RDS files from the working directory into a single list. This list is then summarized with additional functions.

Usage

```
readIn(working.directory)
```

Arguments

`working.directory`
: Optional character string specifying the working directory. If missing, the current working directory will be used.

Details

This function reads all .RDS files from the working directory (or another specified directory) into a single list.

Value

A list of simulation results. Each element in the list consists of a single output from multiLinker().

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
```

`reduceRandomizations` *Reduce randomized results to a manageable list of dataframes*

Description

The metricsNnulls function creates lists of lists of dataframes. This function will combine the dataframes from each null model into a single data frame. The output is a more manageable list of dataframes.

Usage

```
reduceRandomizations(randomizations.list)
```

Arguments

`randomizations.list`
: The results of a call to metricsNnulls()

Details

Given a list of lists of dataframes, such as those that come from a call to metricsNnulls, where the first level of the list relates to a given randomization, and each second level is a data frame containing the calculated metrics after randomization according to a given null model, reduces the results to a simpler list of data frames, where each data frame contains all the results from a given null model from the input randomizations.list.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#below not run for timing issues on CRAN
rawResults <- metricsNnulls(tree, cdm, metrics=c("richness", "NAW_MPD"),
nulls=c("richness", "frequency"))

results <- reduceRandomizations(rawResults)
```

reduceResults *Reduce results from multiLinker into a manageable format*

Description

Calling multiLinker creates .RDS files, one per iteration. This function will combine these results into a more manageable format.

Usage

```
reduceResults(results.list)
```

Arguments

results.list	The results of a call to readIn()

Details

Given a list of results readIn() from multiLinker, this function will reduce those results into a manageable format like that expected for calls to plotOverall and sesOverall.

Value

A list of data frames.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- reduceResults(results)
```

regionalNull *Randomize community data matrix with regional null model*

Description

Null model that simulates community assembly where species probabilities of occurrence are proportional to their regional abundance.

Usage

```
regionalNull(picante.cdm, tree, regional.abundance)
```

Arguments

picante.cdm	Picante-style community data matrix with communities/plots/plots/etc as rows and species as columns
tree	Ape-style phylogeny
regional.abundance	Vector of species names, where each species' name is repeated the number of times necessary to accomodate its abundance in the regional species pool

Details

Although nowhere near as fast as, e.g. randomizeMatrix, this function still runs fairly quickly (no for or while loops). It works by drawing the total number of individuals observed in the input plot from the regional abundance vector. Thus, while a randomized plot will not necessarily have the same number of species as the observed plot, over many iterations it will likely be sampled. We can then concatenate the results by richness at the end, which will only compare observed values to random plots of the same richness. As an example, an observed plot might have two individuals of speciesA and two of speciesB. If the regional abundance vector is c("spA","spA","spA","spA","spB","spB","spB","spC"), and we draw four individuals, it would be possible to draw 1, 2, or 3 species, but in general, two species would be seen in the randomized plots.

Value

A matrix with all species in the input tree in phylogenetic order, and the same number of randomized plots as used in the input community data matrix

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

positions <- competitionArena(prepped)

boundResults <- plotPlacer(no.plots=15, arena.length=300, plot.length=30)

#return a CDM in picante format
cdmTemp <- plotContents(positions$arena, boundResults)

test <- regionalNull(cdmTemp$picante.cdm, tree,
regional.abundance=abundanceVector(cdmTemp$picante.cdm))
```

`relativeCDM` *Convert absolute abundance matrix to relative abundance*

Description

Simple utility function to convert an absolute abundance matrix to a relative abundance matrix.

Usage

```
relativeCDM(picante.cdm, tree = NULL)
```

Arguments

`picante.cdm`	Picante-style community data matrix with communities/plots/plots/etc as rows and species as columns
`tree`	Optional phylo object

Details

This function converts species' absolute abundances in a given community (a row in the input CDM) into relative abundances by dividing observed abundances by the maximum abundance in that row. If a tree is provided, the function confirms that the CDM is indeed in the correct format, otherwise it assumes it is formatted correctly and proceeds accordingly.

Value

A relative abundance matrix otherwise identical to the input CDM.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)
```

runNulls *Randomize input CDM according to defined null models*

Description

Given a prepared nulls.input object, will randomize a community data matrix according to specified null models, and return a list of randomized CDMs.

Usage

```
runNulls(nulls.input, nulls, new_ = FALSE)
```

Arguments

nulls.input	Prepped nulls.input object
nulls	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()). Otherwise, if the user would like to define a new null on the fly, the argument nulls can take the form of a named list of new functions (nulls). If the latter, new_ must be set to TRUE.
new_	Whether or not new nulls are being defined on the fly. Default is FALSE. Set to TRUE if a new null is being used.

Details

Determine which nulls will be calculated by running names(defineNulls()). If only a subset of these is desired, supply metrics with a character vector of the named, available metrics.

Value

A list of matrices. Each matrix is a product of a randomization of the input CDM and one of the specified null models.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)
```

```
prepped <- prepNulls(tree, cdm)

results <- runNulls(prepped)
```

`runSimulations` *Run defined spatial simulations*

Description

Given a prepared simulations.input object, will run all specified spatial simulations, and return a list of randomized CDMs.

Usage

```
runSimulations(simulations.input, simulations, new_ = FALSE)
```

Arguments

`simulations.input`	Prepped simulations.input object
`simulations`	Optional. If not provided, defines the simulations as all of those in defineSimulations. If only a subset of those simulations is desired, then simulations should take the form of a character vector corresponding to named functions from defineSimulations. The available simulations can be determined by running names(defineSimulations()). Otherwise, if the user would like to define a new simulation on the fly, the argument simulations can take the form of a named list of new functions (simulations). In this case, new_ must be set to TRUE.
`new_`	Whether or not new simulations are being defined on the fly. Default is FALSE. Set to TRUE if a new metric is being used.

Details

We ran three spatial simulations in our Ecography paper: neutral, habitat filtering, and competitive exclusion community assembly.

Value

A list of lists of simulation results, where each of the first-order elements in the list relates to a unique simulation as defined in defineSimulations.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=3)

results <- runSimulations(prepped)
```

---

sesField *Calculate a species' standardized trait field*

Description

Calculate the null-model standardized effect size of a species' trait field.

Usage

```
sesField(field.input, metrics, nulls, randomizations, regional.abundance,
  distances.among, cores = "seq")
```

Arguments

field.input	Prepped field.input object.
metrics	Optional. If not provided, defines the metrics as all of those in defineMetrics. If only a subset of those metrics is desired, then metrics should take the form of a character vector corresponding to named functions from defineMetrics. The available metrics can be determined by running names(defineMetrics()).
nulls	Optional. If not provided, defines the nulls as all of those in defineNulls. If only a subset of those is desired, then nulls should take the form of a character vector corresponding to named functions from defineNulls. The available nulls can be determined by running names(defineNulls()).
randomizations	The number of times the input CDM should be randomized and the metrics calculated across it.
regional.abundance	A character vector in the form "s1, s1, s1, s2, s2, s3, etc". Optional, will be generated from the input CDM if not provided.
distances.among	A symmetric distance matrix, summarizing the distances among all quadrats from the cdm. For use with the dispersal null.
cores	This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially.

Details

The trait distance matrix should be symmetrical and "complete". See example. Currently only non-abundance-weighted mean pairwise and interspecific abundance-weighted mean pairwise phylogenetic distances are implemented. The only null models that are currently implemented are the richness and dispersal nulls. The function could be improved by tapping into any of the metrics and nulls defined in defineMetrics and defineNulls.

Value

Data frame of standardized effect sizes of species' trait fields. Table includes the observed trait field, the mean and standard deviation of the species' trait field after randomization with the chosen null model, and the resulting species-specific standarized effect size.

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#in this example, occasionally some species are not in the CDM, so prune the tree
#accordingly so as not to throw any errors
tree <- drop.tip(tree, setdiff(tree$tip.label, colnames(cdm)))

prepped <- prepFieldData(tree=tree, picante.cdm=cdm)

results <- sesField(prepped, randomizations=3,
metrics="NAW_MPD", nulls="richness")
```

---

sesIndiv *Summary statistics of SES results*

Description

Summarizes individual iteration performance of each sim/null/metric combination across all iterations.

Usage

```
sesIndiv(raw.results, direction)
```

Arguments

raw.results	A list of lists of lists of dataframes; the results of a call to readIn.
direction	Character vector that needs to be provided if spatial simulations beyond the standard "random", "filtering", and "competition" simulations are run. The character vector must be the same length as the number of spatial simulations that were run, and can take the possible values of "two.sided" (for a two-tailed test when the SES scores are expected to be centered around 0), "less" (for when the observed SES scores are expected to be less than 0), and "greater" (for when the observed SES scores are expected to be greater than 0). For instance, habitat filtering would be set to "less". The relevant simulation to which these directional tests will be applied can be determined by calling names(raw.results[[1]]). Note that currently, even if a direction vector is supplied, if the standard simulations were run then the supplied direction vector will be replaced with the standard expectations.

Details

This function takes a raw list of results from multiple iterations from multiLinker, and runs a non-exported function, sesSingle, over each element (iteration) in that list. This sesSingle function runs a Wilcoxon signed rank test over each iteration to determine whether the plots from a spatial simulation/null/metric differ from expectations. Assuming there are three spatial simulations named random, filtering, and competition, this function will use two.sided, lesser and greater Wilcoxon tests, respectively. If additional (or a limited set of) spatial simulations are included, requiring modified expectations, these can be passed along with the "direction" argument. It then summarizes the results of those single run tests as the number of sim/null/metrics that deviated beyond expectations and the number that were within expectations. A single run from a given unique metric + null approach is considered as throwing a type I error only if p is less than or equal to 0.05 for the random spatial simulation. It would be possible to also assess whether such unique combinations throw the opposite signal than expected for habitat filtering and competitive exclusion. A unique combination iteration is considered to throw a type II error if the p value from either the filtering or the exclusion simulation is greater than 0.05.

Value

A data frame summarizing the total number of runs per spatial simulation, null, metric, concat.by combination, as well as the type I and II error rates of each such unique combination.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- sesIndiv(results)
```

`sesOverall` *Overall per simulation-null-metric SES test*

Description

This function provides one of many ways of summarizing and considering simulation results.

Usage

```
sesOverall(simulation.list, test, direction)
```

Arguments

`simulation.list`	A summarized results list such as one output from reduceResults(). See examples.
`test`	Either "ttest" or "wilcotest", depending on whether the user wants to run a two-sided t-test or a Wilcoxon signed rank test.
`direction`	Character vector that needs to be provided if spatial simulations beyond the standard "random", "filtering", and "competition" simulations are run. The character vector must be the same length as the number of spatial simulations that were run, and can take the possible values of "two.sided" (for a two-tailed test when the SES scores are expected to be centered around 0), "less" (for when the observed SES scores are expected to be less than 0), and "greater" (for when the observed SES scores are expected to be greater than 0). For instance, habitat filtering would be set to "less". The relevant simulation to which these directional tests will be applied can be determined by calling names(simulation.list).

Details

This function provides one way of summarizing and considering simulation results. It takes as input a vector of all standardized effect sizes for all plots from a given simulation-null-metric combination, and calculates the mean of the vector and whether it differs significantly from a mean of zero. It does this either with a simple two-sided t-test, or with a Wilcoxon signed rank test. If the latter, and if there are three different spatial simulations with names random, filtering and competition, the test is two-sided, less and greater, respectively. If additional spatial simulations are included, requiring modified expectations, these can be passed along with the "direction" argument.

Value

A data frame summarizing the mean, overall standardized effect sizes and the significance of those devations from expectations for each simulation, null, metric combination. This test works across all iterations, and looks for overall shifts in SES from expectations (see details for for expectations).

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#not run
#results <- readIn()
#summ <- reduceResults(results)
#examp <- sesOverall(summ$ses, test="wilcotest")
```

`sesPhyloField` *Calculate a species' standardized trait field*

Description

Calculate the null-model standardized effect size of a species' trait field.

Usage

```
sesPhyloField(tree, picante.cdm, metric, null, randomizations,
  distances.among = NULL, abundance.matters = TRUE,
  abundance.assigned = "directly", cores = "seq")
```

Arguments

`tree`	Phylo object.
`picante.cdm`	A picante-style community data matrix with sites as rows, and species as columns.
`metric`	Phylogenetic metric of choice (see details).
`null`	Null model of choice (see details).
`randomizations`	The number of times the input CDM should be randomized and the metrics calculated across it.
`distances.among`	A symmetric distance matrix, summarizing the distances among all plots from the cdm. For use with the dispersal null.
`abundance.matters`	Default is TRUE. If FALSE, species are sampled from neighboring grid cells with equal probability. For use with the dispersal null.
`abundance.assigned`	For use with the dispersal null. See details there.
`cores`	This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially.

Details

This function is being deprecated. This and the rest of the first generation of field functions are being replaced by a two-step process akin to the calcMetrics set of functions. The user first preps the data with a prep function, then runs the desired metrics and nulls over the prepped object. This allows sets of metrics to be calculated over the same randomized matrix, rather than having to repeatedly generate the same random matrix for each metric. The trait distance matrix should be symmetrical and "complete". See example. Currently only non-abundance-weighted mean pairwise and inter-specific abundance-weighted mean pairwise phylogenetic distances are implemented. The only null models that are currently implemented are the richness and dispersal nulls. The function could be improved by tapping into any of the metrics and nulls defined in defineMetrics and defineNulls.

Value

Data frame of standardized effect sizes of species' phylogenetic fields. Table includes the observed phylogenetic field, the mean and standard deviation of the species' trait field after randomization with the chosen null model, and the resulting species-specific standarized effect size.

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate log-normal abundances
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community data matrix with these inputs
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#example trait field calculations
exampleField <- sesPhyloField(tree=tree, picante.cdm=cdm,
metric="naw.mpd", null="richness", randomizations=10)
```

---

sesTraitField *Calculate a species' standardized trait field*

Description

Calculate the null-model standardized effect size of a species' trait field.

Usage

```
sesTraitField(trait.distance, tree, picante.cdm, metric, null, randomizations,
  distances.among = NULL, abundance.matters = TRUE,
  abundance.assigned = "directly", cores = "seq")
```

Arguments

trait.distance	Symmetrical matrix summarizing pairwise trait distances.
tree	Phylo object.
picante.cdm	A picante-style community data matrix with sites as rows, and species as columns.
metric	Phylogenetic metric of choice (see details).
null	Null model of choice (see details).
randomizations	The number of times the input CDM should be randomized and the metrics calculated across it.
distances.among	A symmetric distance matrix, summarizing the distances among all plots from the cdm. For use with the dispersal null.
abundance.matters	Default is TRUE. If FALSE, species are sampled from neighboring grid cells with equal probability. For use with the dispersal null.
abundance.assigned	For use with the dispersal null. See details there.
cores	This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially.

Details

This function is being deprecated. This and the rest of the first generation of field functions are being replaced by a two-step process akin to the calcMetrics set of functions. The user first preps the data with a prep function, then runs the desired metrics and nulls over the prepped object. This allows sets of metrics to be calculated over the same randomized matrix, rather than having to repeatedly generate the same random matrix for each metric. The trait distance matrix should be symmetrical and "complete". See example. Currently only non-abundance-weighted mean pairwise and interspecific abundance-weighted mean pairwise phylogenetic distances are implemented. The only null models that are currently implemented are the richness and dispersal nulls. The function could be improved by tapping into any of the metrics and nulls defined in defineMetrics and defineNulls.

Value

Data frame of standardized effect sizes of species' trait fields. Table includes the observed trait field, the mean and standard deviation of the species' trait field after randomization with the chosen null model, and the resulting species-specific standarized effect size.

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate trait evolution up the tree. Make 2-d trait space and find distances between
#species in that space
traits <- evolveTraits(tree)

#calculate the distances betweeen species
dists <- as.matrix(dist(traits[[2]], diag=TRUE, upper=TRUE))

#simulate log-normal abundances
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community data matrix with these inputs
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#example trait field calculations
exampleField <- sesTraitField(trait.distance=dists, tree=tree, picante.cdm=cdm,
metric="naw.mpd", null="richness", randomizations=10)
```

`settleSome` *Randomly settle individuals in a spatial arena*

Description

Given output from the killSome function, randomly settles individuals in the arena.

Usage

```
settleSome(killSome.output)
```

Arguments

`killSome.output`
: Output from the killSome function

Details

This function uses the number killed element of the killSome output to randomly draw from the regional abundance vector, then settles the individuals at random in the arena.

Value

A list of 4 elements: the average relatedness in the geographic neighbordhood of consideration (passed directly from the killSome output, not re-calculated here), the regional abundance vector, the new spatial arena, and the dimensions of that arena.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#prep the data for the simulation
prepped <- prepSimulations(tree, arena.length=300, mean.log.individuals=2,
length.parameter=5000, sd.parameter=50, max.distance=20, proportion.killed=0.2,
competition.iterations=5)

#use the competition simulation
positions <- competitionArena(prepped)

#in normal use, these parameters will be carried down from the simulations.input object
new.arena <- killSome(tree, arena.output=positions, max.distance=50,
proportion.killed=0.2)

#now settle some indiviudals
newer.arena <- settleSome(new.arena)

#look at how number of individuals in arena changes
dim(new.arena$arena)
dim(newer.arena$arena)
```

---

simulateComm *Generate a simulated community data matrix*

Description

Given a phylo object, a vector of desired species richnesses, and a vector of potential species abundances, will generate a community data matrix with these characteristics.

Usage

```
simulateComm(tree, richness.vector, abundances)
```

Arguments

tree	Phylo object
richness.vector	Vector of desired species richness, one for each desired plot
abundances	A vector of potential abundances, e.g. a log-normal distribution

Details

There is currently no implementation to control the frequency with which a given species is selected. As of metricTester 1.2.2, this function no longer can occasionally return a CDM missing species that are in the input phylogeny.

Value

A community data matrix (as a data frame) with species as columns and sites as rows.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)
```

---

summaries — *Return summary statistics from a data frame of randomized metric values*

Description

Summarizes observed metric scores. Returns the mean, standard deviation and 95% confidence intervals of each plot or observed richness.

Usage

```
summaries(null.output, concat.by = "richness")
```

Arguments

`null.output`	Data frame of randomized metric values such as an element from a call to reduceRandomizations()
`concat.by`	Whether to concatenate the randomizations by richness, plot or both

Details

Given a data frame of metric values, summarizes either by plot or richness. Outputs the mean, standard deviation and 95% confidence intervals of each plot or observed richness. If provided with concat.by="both", outputs a list of two data frames, one for by richness and one for by plot. Otherwise, outputs a data frame.

Value

Either a list of or a data frame of summarized metric scores, see details.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#run the metrics and nulls combo function
rawResults <- metricsNnulls(tree=tree, picante.cdm=cdm, randomizations=2, cores="seq",
nulls=c("richness","frequency"), metrics=c("richness","NAW_MPD"))

#summarize the results
results <- reduceRandomizations(rawResults)

test <- summaries(results$frequency, concat.by="richness")
```

---

summCorrs *Summarize correlations among metrics over a result from a varyX function*

Description

Takes the results of one of the varyX functions, and calculates the correlations among metrics, returning either the raw or summarized correlations.

Usage

```
summCorrs(vary.results, exclude, return.raw = FALSE,
  cor.method = "spearman")
```

Arguments

vary.results	Results from any of the varyX (e.g., varyAbundance) functions.
exclude	Results columns to exclude from correlation. For instance, with alpha metrics, one would want to, at the minimum, exclude the plot name column.
return.raw	Default is FALSE. Whether to return the raw correlation coefficients between the metrics for each element from vary.results, or whether to summarize the correlations by their mean per parameter set from vary.results.

`cor.method` Default is "spearman", but takes any of the other options from the base cor function.

Details

Not a well tested function.

Value

If return.raw is set to FALSE, returns a list of matrices, one for each set of parameters in vary.results, summarizing the mean correlation coefficients between each pairwise metric correlation. If return.raw is set to TRUE, returns a list of lists, one for each set of parameters. Each of these lists is the length of the number of iterations in vary.results. Each element in the list is a matrix of pairwise metric correlations.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#below not run for timing issues on CRAN
#system.time(vSize <- varyX(alpha=TRUE, tree.size=c(40, 50), richness=20:30, delta=1,
#abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1, iterations=2, cores="seq"))

#test <- summCorrs(vSize, exclude=c("plot", "richness"))
```

---

synthComm *Create synthetic community niche space*

Description

Create niche spaces with similar characteristics to input.

Usage

```
synthComm(ordination.space, road.map, cov.matrix = NULL)
```

Arguments

`ordination.space`
Total niche space, e.g. the $x element of a PCA object. This should be a matrix. If it is not, it will be converted to one. This means categorical variables will not be handled well.

road.map	Identical to the input for the 'a' argument in the dbFD function of the FD package, and to the picante.cdm argument used elsewhere in this package. Thus, this is a matrix containing the abundance of each 'species' in the ordination results. Rows are "sites" and columns are "species". Rather than abundances, the values can simply be presence/absences. Moreover, sites could be species and species could be individuals. See details and examples.
cov.matrix	Optional covariance matrix, e.g. from a larger total trait space.

Details

This function is totally untested and may not do what I think it does. Use with care! In theory, it takes the points that describe the total trait space of a community (or multiple communities, depending upon the analysis) and a community road map, optionally a cov.matrix, and will return a new total trait space with the same number of species (or sites) and observations per species (or community) as the input. It works by first simulating a multivariate normal distribution with centroids like those from ordination.space. It bases the positions of these centroids on the covariance matrix of the input ordination.space unless a different covariance matrix (e.g., from a larger trait space) is provided. Points (either akin to multiple individuals or species, depending on the level of analysis) are then distributed around those points with multivariate normal distributions like those in the input.

Value

Dataframe with the coordinates and species identities of the new niche space.

References

Miller, E. T. 2016. Random thoughts, though please cite metricTester via our 2016 Ecography paper.

Examples

```
#generate six species' niche spaces of 20 observations each in 3 dimensions. first
#define the centroids of their distributions
centroids <- matrix(nrow=6, ncol=3, rep(seq(from = -2, to = 2, length.out=6), 3))

#brute force the points into a list and reduce back into a large simulated
#ordination.space object

output <- list()

for(i in 1:6)
{
temp1 <- rnorm(n=20, mean=centroids[i,1], sd=1/i)
temp2 <- rnorm(n=20, mean=centroids[i,2], sd=1/i)
temp3 <- rnorm(n=20, mean=centroids[i,3], sd=1/i)
output[[i]] <- data.frame(temp1, temp2, temp3)
}

totalNiche <- Reduce(rbind, output)
```

```
#add a species then color column to totalNiche then scramble to simulate real data
totalNiche$species <- sort(rep(paste("species", 1:6, sep=""),20))

toMerge <- data.frame(species=paste("species", 1:6, sep=""), color=1:6)

totalNiche <- merge(totalNiche, toMerge)

totalNiche <- totalNiche[sample(row.names(totalNiche)),]

#plot the points to give some sense of what it looks like (not run, but works)
#plot(totalNiche[,3]~totalNiche[,2], col=totalNiche$color, pch=20)

#create a road map identifying which points belong to which species
roadMap <- matrix(nrow=6, ncol=120, 0)

row.names(roadMap) <- paste("species", 1:6, sep="")
colnames(roadMap) <- 1:120
roadMap[1,][row.names(totalNiche)[totalNiche$species=="species1"]] <- 1
roadMap[2,][row.names(totalNiche)[totalNiche$species=="species2"]] <- 1
roadMap[3,][row.names(totalNiche)[totalNiche$species=="species3"]] <- 1
roadMap[4,][row.names(totalNiche)[totalNiche$species=="species4"]] <- 1
roadMap[5,][row.names(totalNiche)[totalNiche$species=="species5"]] <- 1
roadMap[6,][row.names(totalNiche)[totalNiche$species=="species6"]] <- 1

roadMap <- as.data.frame(roadMap)

#now run the synthComm null model. exclude 1st and 5th columns since these are species
#names and color, which are not normal inputs
temp <- synthComm(totalNiche[,c(-1,-5)], roadMap)

#plot the points to give some sense of what it looks like (not run, but works)
#plot(temp[,3]~temp[,2], col=temp$species, pch=20)
```

`traitField` *Calculate a species' trait field*

Description

Calculate the similarity in trait space of a species to those it occurs with.

Usage

```
traitField(trait.distance, picante.cdm, metric)
```

Arguments

`trait.distance`	Symmetrical matrix summarizing pairwise trait distances. If it contains.
`picante.cdm`	A picante-style community data matrix with sites as rows, and species as columns.
`metric`	Phylogenetic metric of choice (see details).

Details

This function is being deprecated. This and the rest of the first generation of field functions are being replaced by a two-step process akin to the calcMetrics set of functions. The user first preps the data with a prep function, then runs the desired metrics and nulls over the prepped object. This allows sets of metrics to be calculated over the same randomized matrix, rather than having to repeatedly generate the same random matrix for each metric. The trait distance matrix should be symmetrical and "complete". See example. Currently this is only programmed to use either non-abundance-weighted mean pairwise or interspecific abundance-weighted mean pairwise phylogenetic distance. Importantly, we are in the process of generalizing the phylo & trait field functions, so they operate more like the calcMetrics functions. In other words, the user will prep their data first, then choose which metrics to calculate and the function will detect whether to calculate phylo or trait fields based on the inputs. Take note of this, as code using the current forms of these functions is liable to break when these updates are made.

Value

Named vector of species' trait fields.

References

Miller, Wagner, Harmon & Ricklefs. In review. Radiating despite a lack of character: closely related, morphologically similar, co-occurring honeyeaters have diverged ecologically.

Examples

```
#simulate tree with birth-death process
tree <- geiger::sim.bdtree(b=0.1, d=0, stop="taxa", n=50)

#simulate trait evolution up the tree. Make 2-d trait space and find distances between
#species in that space
traits <- evolveTraits(tree)

#calculate the distances betweeen species
dists <- as.matrix(dist(traits[[2]], diag=TRUE, upper=TRUE))

#simulate log-normal abundances
sim.abundances <- round(rlnorm(5000, meanlog=2, sdlog=1)) + 1

#simulate a community data matrix with these inputs
cdm <- simulateComm(tree, richness.vector=10:25, abundances=sim.abundances)

#example trait field calculations
exampleField <- traitField(trait.distance=dists, picante.cdm=cdm, metric="naw.mpd")
```

---

varLandscape *Create landscapes with varying degrees of heterogeneity*

Description

This function will simulate 3-dimensional landscapes of varying complexity.

Usage

```
varLandscape(cells, seeds = 1, exponent = 1, cutoff = 0)
```

Arguments

cells	The number of cells to divide each side of the arena into. Larger values provide smoother looking surfaces, but values larger than 100 can require too much RAM to run.
seeds	The number of "peaks" or trait "optima" that will be chosen in the landscape. Default is 1.
exponent	The exponent to which the distances will be raised. Default is 1. Values larger than 1 have the effect of making distance decay slowly at first, then drop off more quickly at the end, while values smaller than 1 have the effect of dropping off quickly and then decreasing slowly.
cutoff	Values below which distances from the focal cell will be converted to zero. This operates after the exponent is applied to the distance matrix, and after the distances specific to a given focal cell have been scaled to min 0 max 1. The default cutoff is zero, meaning that all but the most distanct cells are still influenced by the new optimum of the focal cell. Increasing this number towards 1 has the effect of minimizing the distance over which the focal cell influences neighboring cells.

Details

This function forms the guts of a new habitat filtering spatial simulation. The output from the function is a square matrix with values corresponding, in my mind, to optimum trait values for a location in 2d space. Alternatively, this might be useful for simulations of elevational gradients. A good sequence to show how landscapes can be varied might be (all with cells = 100 and exponent = 1) to change seeds from 1 to 2 to 10 while holding cutoff at 0. Then change cutoff from 0.01 to 0.1 to 0.9 while holding seeds at 10.

Value

A square matrix of dimensions cells x cells.

References

Miller, E. T. 2016. A new dispersal-informed null model for community ecology shows strong performance. biorxiv.

Examples

```
plotrix::color2D.matplot(varLandscape(10, seeds=1, exponent=1, cutoff=0),
cs1=c(0.2,0.4,0.8), cs2=c(0,0.5,0.8), cs3=c(1,0.5,0), border=NA)
```

varyX *Calculate alpha or beta metrics across a set of parameters*

Description

Takes a specified set of parameters, and holds those constant while varying one of the parameters to see variance in community structure metrics.

Usage

```
varyX(alpha = TRUE, tree.size, richness, delta, abundances, beta.iterations,
  iterations, cores = "seq")
```

Arguments

alpha	Whether to calculate alpha or beta phylogenetic community structure metrics. Default is TRUE. Set to FALSE for beta metrics.
tree.size	Either a single number, or if the aim is to vary tree size while holding other parameters constant, then a list of numbers of species desired in each total tree. If provided as a vector of numbers, will be coerced to a list.
richness	Either the number of species to be placed in each plot, or if the aim is to vary richness while holding other parameters constant, then a list of vectors of number of species to be placed in each plot. See examples.
delta	Either a value for the delta transformation (Pagel 1999), or if the aim is to vary tree shape while holding other parameters constant, then a list of numbers of delta transformations. If provided as a vector of numbers, will be coerced to a list. Values greater than 1 push the branching events towards the root, while values less than 1 push the branching events closer to the tips. See details for particularly low delta values.
abundances	Either a vector of abundances, or if the aim is to vary the abundance distribution function, then a list of vectors of abundances. See examples.
beta.iterations	Because the type of beta-level phylogenetic community structure metrics used here return a single value per community data matrix, it is not possible to look for inter-metric correlations with only a single matrix and tree. To deal with this, the same tree can be used with different community data matrices. This argument specifies the number of matrices to be used per tree. Not needed if alpha=TRUE.
iterations	How many times to simulate the given set of parameters. For instance, with a single tree size, richness.vector, delta, and two sets of abundances, and 10 iterations, the result will be a list with 10 elements. Each of those 10 elements will be a list of two elements, each of which will be the calculated metrics for a given set of parameters (one for each abundance vector).

cores — This function can run in parallel. In order to do so, the user must specify the desired number of cores to utilize. The default is "seq", which runs the calculations sequentially. The iteration aspect of the function is parallelized, so for efficiency purposes, it is best to run this over numerous iterations rather than repeating the same parameter numerous times (e.g., rather than setting deltas to rep(1, 10), set delta to 1 and iterations to 10).

Details

If given a small value, e.g. 0.1, the delta parameter (tree shape) can occasionally result in oddly formatted trees that would cause errors. To deal with this, there is an internal check that will recreate a new tree and re-scale it with the desired delta. This has not been tested at delta < 0.1, and is currently programmed with a while loop. Care should be taken not to get R stuck in an indefinite loop at delta values even lower than 0.1

Value

A list of lists of data frames.

References

Miller, E. T., D. R. Farine, and C. H. Trisos. 2016. Phylogenetic community structure metrics and null models: a review with new methods and software. Ecography DOI: 10.1111/ecog.02070

Examples

```
#example of how to vary tree size
#not run
#system.time(temp <- varyX(alpha=TRUE, tree.size=c(59, 100),
#richness=40:59, delta=1,
#abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1, iterations=2))

#example of how to vary richness
#not run
#system.time(temp <- varyX(alpha=TRUE, tree.size=59,
#richness=list(30:39, 40:49), delta=1,
#abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1, iterations=2))

#example of how to vary tree shape
#not run
#system.time(temp <- varyX(alpha=TRUE, tree.size=59,
#richness=40:59, delta=c(0.1,10),
#abundances=round(rlnorm(5000, meanlog=2, sdlog=1)) + 1, iterations=2))

#example of how to vary abundance
#not run
#inputAbunds <- list(rep(2,5000), (round(rnorm(5000, 50, 15)) + 1),
#(round(rlnorm(5000, meanlog=2, sdlog=1)) + 1))

#system.time(temp <- varyX(alpha=TRUE, tree.size=59,
#richness=40:59, delta=1, abundances=inputAbunds, iterations=2))
```

Index