Package 'ecospace'

October 13, 2022

Type Package

Title Simulating Community Assembly and Ecological Diversification Using Ecospace Frameworks

Version 1.4.2

Date 2020-06-12

Depends R (>= 4.0.0)

Imports FD (>= 1.0-12)

Suggests vegan, knitr, rmarkdown, data.table

Maintainer Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

Description Implements stochastic simulations of community assembly (ecological diversification) using customizable ecospace frameworks (functional trait spaces). Provides a wrapper to calculate common ecological disparity and functional ecology statistical dynamics as a function of species richness. Functions are written so they will work in a parallel-computing environment.

License CC0

URL https://github.com/pnovack-gottshall/ecospace, http://www.ben.edu/faculty/pnovack-gottshall/index.html

LazyData TRUE

BugReports https://github.com/pnovack-gottshall/ecospace/issues

RoxygenNote 7.1.0

Encoding UTF-8

VignetteBuilder knitr

NeedsCompilation no

Author Phil Novack-Gottshall [aut, cre]

Repository CRAN

Date/Publication 2020-06-13 17:50:03 UTC

R topics documented:

`ecospace-package` *ecospace: Simulating Community Assembly and Ecological Diversification Using Ecospace Frameworks*

Description

ecospace is an R package that implements stochastic simulations of community assembly (ecological diversification) using customizable ecospace frameworks (functional trait spaces). Simulations model the 'neutral', 'redundancy', 'partitioning', and 'expansion' models of Bush and Novack-Gottshall (2012) and Novack-Gottshall (2016a,b). It provides a wrapper to calculate common ecological disparity and functional ecology statistical dynamics as a function of species richness. Functions are written so they will work in a parallel-computing environment.

Details

The package also contains a sample data set, functional traits for Late Ordovician (Type Cincinnatian) fossil species from the Kope and Waynesville formations, from Novack-Gottshall (2016b).

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bush, A. and P.M. Novack-Gottshall. 2012. Modelling the ecological-functional diversification of marine Metazoa on geological time scales. *Biology Letters* 8: 151-155.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

See Also

The 'calc_metrics' function relies extensively on the functional diversity package FD-package, and hence lists this package as a depends, so it is loaded simultaneously.

Examples

```
# Get package version, citation, updates, and vignette
packageVersion("ecospace")
citation("ecospace")
news(package = "ecospace")
vignette("ecospace")

# Create an ecospace framework (functional-trait space) with 15 characters
#    (functional traits) of mixed types
nchar <- 15
ecospace <- create_ecospace(nchar = nchar, char.state = rep(3, nchar),
  char.type = rep(c("factor", "ord.fac", "ord.num"), nchar / 3))

# Use to assemble a stochastic "neutral" sample of 20 species (from
#    initial seeding by 5 species)
x <- neutral(Sseed = 5, Smax = 20, ecospace = ecospace)
head(x, 10)

# Calculate ecological disparity (functional diversity) dynamics as a
#    function of species richness
# Statistic 'V' [total variance] not calculated because there are factors
#    in the sample
metrics <- calc_metrics(samples = x, Smax = 20, Model = "Neutral", Param = "NA")
metrics

# Plot statistical dynamics as function of species richness
op <- par()
par(mfrow = c(2,4), mar = c(4, 4, 1, .3))
attach(metrics)
plot(S, H, type = "l", cex = .5)
plot(S, D, type = "l", cex = .5)
plot(S, M, type = "l", cex = .5)
plot(S, V, type = "l", cex = .5)
plot(S, FRic, type = "l", cex = .5)
plot(S, FEve, type = "l", cex = .5)
plot(S, FDiv, type = "l", cex = .5)
plot(S, FDis, type = "l", cex = .5)

par(op)
```

`calc_metrics` *Calculate Ecological Disparity (Functional Diversity) Dynamics as Function of Sample Size.*

Description

Wrapper to FD::dbFD that calculates common ecological disparity and functional diversity statistics. When used with species-wise simulations of community assembly or ecological diversification (and default increm = 'TRUE'), calculates statistical dynamics incrementally as a function of species richness. Avoids file-sharing errors so that can be used in 'embarrasingly parallel' implementations in a high-performance computing environment.

Usage

```
calc_metrics(
  nreps = 1,
  samples = NA,
  Smax = NA,
  Model = "",
  Param = "",
  m = 3,
  corr = "lingoes",
  method = "Euclidean",
  increm = TRUE,
  ...
)
```

Arguments

nreps	Sample number to calculate statistics for. Default is the first sample nreps = 1, but statistics can be calculated for other samples (i.e., second sample if nreps = 2), or multiple samples if assigned a vector (sequence) of integers and function is applied within lapply or related function.
samples	Data frame (if nreps = 1) or list of data frames (if nreps = seq() or nreps! = 1), with each data frame a species-by-trait matrix with species as rows and traits as columns. Traits can be binary, numeric, ordered numeric, factor, or ordered factor types. Each sample is converted to a distance metric (see method below) before calculating statistics.
Smax	Maximum number of samples rows (species) to include in calculations, incremented starting with first row. Default (NA) is to increment to the maximum number of samples rows (calculated separately for each data frame sample, if a list of data frames). If Smax is greater than the size of a sample, then calculation stops after calculating the sample statistics and issues a warning.
Model	Optional character string or numeric value naming simulation model. A warning issues if left blank.
Param	Optional numeric value or character string naming strength parameter used in simulation. A warning issues if left blank.
m	The number of PCoA axes to keep as 'traits' for calculating FRic and FDiv in FD::dbFD. Default m = 3 is justified below, but any integer value greater than 1 is possible. See 'details' for more information.
corr	Character string specifying the correction method to use in FD::dbFD when the species-by-species distance matrix cannot be represented in a Euclidean space.

Default corr = 'lingoes' is justified below, but see FD::dbFD for other possible values.

method — Distance measure to use when calculating functional distances between species. Default is method = 'Euclidean' using stats::dist. method = 'Gower' or any other value uses Gower distance (using FD::gowdis). Presence of factor or ordered factor character types forces use of Gower distance, triggering a warning to notify user when changed internally.

increm — Default increm = 'TRUE' calculates statistics incrementally as a function of species richness. increm = 'FALSE' only calculates a single set of statistics for the entire sample.

... — Additional parameters for controlling FD::dbFD. Common uses include setting calc.FRic = FALSE or calc.FDiv = FALSE to exclude calculation of FRic and FDiv. Note that the arguments m, corr, and method above have different defaults than used in FD::dbFD, and w.abun = FALSE and messages = FALSE are also internally changed to different defaults. These and others can be controlled here.

Details

The primary goal of this function is to describe the statistical dynamics of common ecological disparity (functional diversity) metrics as a function of species richness (sample size). Statistics are calculated incrementally within samples, first for the first row (species), second for the first and second rows, ..., ending with the entire sample (by default, or terminating with Smax total species). The function assumes that supplied samples are ecologically or evolutionary cohesive assemblages in which there is a logical order to the rows (such that the sixth row is the sixth species added to the assemblage) and that such incremental calculations are sensible. See Novack-Gottshall (2016a,b) for additional context. Samples must have species as rows and traits as columns (of many allowed character types), and have class(data.frame) or a list of such data frames, with each data frame a separate sample.

Statistics calculated include four widely used in ecological disparity studies (adapted from studies of morphological disparity) and four used in functional diversity studies. See Foote (1993), Ciampaglio et al. (2001), and Wills (2001) for definitions and details on morphological disparity measures and Novack-Gottshall (2007; 2016a,b) for implementation as measures of ecological disparity. See Mason et al. (2005), Anderson et al. (2006), Villeger et al. (2008), Laliberte and Legendre (2010), Mouchet et al. (2010), Mouillot et al. (2013) for definitions and details on functional diversity statistics. For computation details of functional diversity metrics, see Laliberte and Shipley (2014) package FD, and especially FD::dbFD, which this function wraps around to calculate the functional diversity statistics.

Statistic (S) is species (taxonomic) richness, or sample size.

When increm = 'FALSE', the function calculates statistics for the entire sample(s) instead of doing so incrementally. In this case, the implementation is essentially the same as FD::dbFD with default arguments (e.g., m, corr) that reduce common calculation errors, plus inclusion of common morphological disparity statistics.

Statistics that measure diversity (unique number of life habits / trait combinations within ecospace / functional-trait space):

H Life habit richness, the number of functionally unique trait combinations.

Statistics that measure disparity (or dispersion of species within ecospace / functional-trait space) (Note these statistics are sensitive to outliers and sample size):

M Maximum pairwise distance between species in functional-trait space, measured using the distance `method` specified above.

FRic Functional richness, the minimal convex-hull volume in multidimensional principal coordinates analysis (PCoA) trait-space ordination.

FDiv Functional divergence, the mean distance of species from the PCoA trait-space centroid.

Statistics that measure internal structure (i.e., clumping or inhomogeneities within the trait-space):

D Mean pairwise distance between species in functional-trait space, measured using the distance `method` specified above.

V Total variance, the sum of variances for each functional trait across species; when using factor or ordered factor character types, this statistic cannot be measured and is left blank, with a warning.

FDis Functional dispersion, the total deviance of species from the circle with radius equal to mean distance from PCoA trait-space centroid.

Statistics that measure spacing among species within the trait-space:

FEve Functional evenness, the evenness of minimum-spanning-tree lengths between species in PCoA trait-space.

The default number of PCoA axes used in calculating of FRic and FDiv equals `m = 3`. Because their calculation requires more species than traits (here the `m = 3` PCoA axes), the four functional diversity statistics are only calculated when a calculated sample contains a minimum of `m` species (S) or unique life habits (H). `qual.FRic` is appended to the output to record the proportion ('quality') of PCoA space retained by this loss of dimensionality. Although including more PCoA axes allows greater statistical power (Villeger et al. 2011, Maire et al. 2015), the use of `m = 3` here is computationally manageable, ecologically meaningful, and allows standardized measurement of statistical dynamics across the wide range of sample sizes typically involved in simulations of ecological/evolutionary assemblages, especially when functionally redundant data occur. Other integers greater than 1 can also be specified. See the help file for `FD::dbFD` for additional information.

Lingoes correction `corr = 'lingoes'`, as recommended by Legendre and Anderson (1999), is called when the species-by-species distance matrix cannot be represented in a Euclidean space. See the help file for `FD::dbFD` for additional information.

Note that the ecological disparity statistics are calculated on the raw (unstandardized) distance matrix. The functional diversity statistics are calculated on standardized data using standardizations in `FD::dbFD`. If all traits are numeric, they by default are standardized to mean 0 and unit variance. If not all traits are numeric, Gower's (1971) standardization by the range is automatically used.

Value

Returns a data frame (if `nreps` is a single integer or `samples` is a single data frame) or a list of data frames. Each returned data frame has `Smax` rows corresponding to incremental species richness (sample size) and 12 columns, corresponding to:

`Model` (optional) `Model` name

Param	(optional) strength parameter
S	Species richness (sample size)
H	Number of functionally unique life habits
D	Mean pairwise distance
M	Maximum pairwise distance
V	Total variance
FRic	Functional richness
FEve	Functional evenness
FDiv	Functional divergence
FDis	Functional dispersion
qual.FRic	proportion ('quality') of total PCoA trait-space used when calculating FRic and FDiv

Note

A bug exists within `FD::gowdis` where nearest-neighbor distances can not be calculated when certain characters (especially numeric characters with values other than 0 and 1) share identical traits across species. The nature of the bug is under investigation, but the current implementation is reliable under most uses. If you run into problems because of this bug, a work-around is to manually change the function to call `cluster::daisy` using `metric = "gower"` instead.

If calculating statistics for more than several hundred samples, it is recommended to use a parallel-computing environment. The function has been written to allow usage (using `lapply` or some other list-apply function) in 'embarrassingly parallel' implementations in such HPC environments. Most importantly, overwriting errors during calculation of convex hull volume in FRic are avoided by creating CPU-specific temporarily stored vertices files.

See Novack-Gottshall (2016b) for recommendations for using random forest classification trees to conduct multi-model inference.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Anderson, M. J., K. E. Ellingsen, and B. H. McArdle. 2006. Multivariate dispersion as a measure of beta diversity. *Ecology Letters* 9(6):683-693.

Ciampaglio, C. N., M. Kemp, and D. W. McShea. 2001. Detecting changes in morphospace occupation patterns in the fossil record: characterization and analysis of measures of disparity. *Paleobiology* 27(4):695-715.

Foote, M. 1993. Discordance and concordance between morphological and taxonomic diversity. *Paleobiology* 19:185-204.

Gower, J. C. 1971. A general coefficient of similarity and some of its properties. *Biometrics* 27:857-871.

Laliberte, E., and P. Legendre. 2010. A distance-based framework for measuring functional diversity from multiple traits. *Ecology* 91(1):299-305.

Legendre, P., and M. J. Anderson. 1999. Distance-based redundancy analysis: testing multispecies responses in multifactorial ecological experiments. *Ecological Monographs* 69(1):1-24.

Maire, E., G. Grenouillet, S. Brosse, and S. Villeger. 2015. How many dimensions are needed to accurately assess functional diversity? A pragmatic approach for assessing the quality of functional spaces. *Global Ecology and Biogeography* 24(6):728-740.

Mason, N. W. H., D. Mouillot, W. G. Lee, and J. B. Wilson. 2005. Functional richness, functional evenness and functional divergence: the primary components of functional diversity. *Oikos* 111(1):112-118.

Mouchet, M. A., S. Villeger, N. W. H. Mason, and D. Mouillot. 2010. Functional diversity measures: an overview of their redundancy and their ability to discriminate community assembly rules. *Functional Ecology* 24(4):867-876.

Mouillot, D., N. A. J. Graham, S. Villeger, N. W. H. Mason, and D. R. Bellwood. 2013. A functional approach reveals community responses to disturbances. *Trends in Ecology and Evolution* 28(3):167-177.

Novack-Gottshall, P.M. 2007. Using a theoretical ecospace to quantify the ecological diversity of Paleozoic and modern marine biotas. *Paleobiology* 33: 274-295.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

Villeger, S., N. W. H. Mason, and D. Mouillot. 2008. New multidimensional functional diversity indices for a multifaceted framework in functional ecology. *Ecology* 89(8):2290-2301.

Villeger, S., P. M. Novack-Gottshall, and D. Mouillot. 2011. The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. *Ecology Letters* 14(6):561-568.

Wills, M. A. 2001. Morphological disparity: a primer. Pp. 55-143. *In* J. M. Adrain, G. D. Edgecombe, and B. S. Lieberman, eds. *Fossils, phylogeny, and form: an analytical approach.* Kluwer Academic/Plenum Publishers, New York.

Laliberte, E., and B. Shipley. 2014. *FD: Measuring functional diversity from multiple traits, and other tools for functional ecology*, Version 1.0-12.

See Also

`FD::dbFD` for details on the core function wrapped here for calculating functional diversity statistics. `neutral`, `redundancy`, `partitioning`, `expansion` for building samples using simulations. `rbind_listdf` for efficient way to combine lists of data frames for subsequent analyses.

Examples

```
# Build ecospace framework and a random 50-species sample using neutral rule:
ecospace <- create_ecospace(nchar = 15, char.state = rep(3, 15), char.type = rep("numeric", 15))
sample <- neutral(Sseed = 5, Smax = 50, ecospace = ecospace)
# Using Smax = 10 here for fast example
metrics <- calc_metrics(samples = sample, Smax = 10, Model = "Neutral", Param = "NA")
metrics
```

```
# Plot statistical dynamics as function of species richness
op <- par()
par(mfrow = c(2,4), mar = c(4, 4, 1, .3))
attach(metrics)
plot(S, H, type = "l", cex = .5)
plot(S, D, type = "l", cex = .5)
plot(S, M, type = "l", cex = .5)
plot(S, V, type = "l", cex = .5)
plot(S, FRic, type = "l", cex = .5)
plot(S, FEve, type = "l", cex = .5)
plot(S, FDiv, type = "l", cex = .5)
plot(S, FDis, type = "l", cex = .5)

par(op)

# Argument 'increm' switches between incremental and entire-sample calculation
metrics2 <- calc_metrics(samples = sample, Smax = 10, Model = "Neutral",
                         Param = "NA", increm = FALSE)
metrics2
identical(tail(metrics, 1), metrics2) # These are identical

# ... can further control 'FD::dbFD', here turning off calculation of FRic and FDiv
metrics3 <- calc_metrics(samples = sample, Smax = 10, Model = "Neutral",
                         Param = "NA", calc.FRic = FALSE, calc.FDiv = FALSE)
metrics3
rbind(metrics[10, ], metrics3[10, ])

## Not run:
# Can take a few minutes to run to completion
# Calculate for 5 samples
nreps <- 1:5
samples <- lapply(X = nreps, FUN = neutral, Sseed = 5, Smax = 50, ecospace)
metrics <- lapply(X = nreps, FUN = calc_metrics, samples = samples,
                  Model = "Neutral", Param = "NA")
alarm()
str(metrics)

## End(Not run)
```

`create_ecospace` *Create Ecospace Framework.*

Description

Create ecospace frameworks (functional trait spaces) of specified structure.

Usage

```
create_ecospace(
  nchar,
```

```
  char.state,
  char.type,
  char.names = NA,
  state.names = NA,
  constraint = Inf,
  weight.file = NA
)
```

Arguments

nchar	Number of life habit characters (functional traits).
char.state	Numeric vector of number of character states in each character.
char.type	Character string listing type for each character. See 'Details' for explanation. Allowed types include: • numeric for numeric and binary characters, • ord.num for ordered numeric characters, • ord.fac for ordered factor characters, or • factor for factor characters.
char.names	Optional character string listing character names.
state.names	Optional character string listing character state names.
constraint	Positive integer specifying the maximum number of "multiple presences" to allow if using multistate binary/numeric character types. The default Inf allows all possible permutations (except for "all absences"). See 'Details' for additional explanation.
weight.file	Data frame (species X trait matrix) or a vector (of mode numeric, integer, or array) of relative weights for ecospace character-state probabilities. Default action omits such probabilities and creates equal weighting among states. If a data frame is supplied, the first three columns must be (1) class [or similar taxonomic identifier], (2) genus, and (3) species names (or three dummy columns that will be ignored by algorithm).

Details

This function specifies the data structure for a theoretical ecospace framework used in Monte Carlo simulations of ecological diversification. An ecospace framework (functional trait space) is a multidimensional data structure describing how organisms interact with their environments, often summarized by a list of the individual life habit characters (functional traits) inhabited by organisms. Commonly used characters describe diet and foraging habit, mobility, microhabitat, among others, with the individual diets, modes of locomotions, and microhabitats as possible character states. When any combination of character states is allowed, the framework serves as a theoretical ecospace; actually occurring life-habit combinations circumscribe the realized ecospace.

Arguments nchar, char.state, char.type specify the number and types of characters and their states. Character names and state names are optional, and assigned using numeric names (i.e., character 1, character 2, etc.) if not provided. The function returns an error if the number of states and names is different than numbers specified in provided arguments.

Allowed character types include the following:

- `numeric` for numeric and binary characters, whether present/absent or multistate. See below for examples and more discussion on these implementations.
- `ord.num` for ordered numeric values, whether discrete or continuous. Examples include body size, metabolic rate, or temperature tolerance. States are pulled as sorted unique levels from `weight.file`, if provided.
- `ord.fac` for ordered factor characters (factors with a specified order). An example is mobility: habitual > intermittent > facultative > passive > sedentary. (If wish to specify relative distances between these ordered factors, it is best to use an ordered numeric character type instead).
- `factor` for discrete, unordered factors (e.g., diet can have states of autotrophic, carnivorous, herbivorous, or microbivorous).

Binary characters can be treated individually (e.g., states of present = 1/absent = 0) or can be treated as multiple binary character states. For example, the character 'reproduction' could be treated as including two states [sexual, asexual] with exclusively sexual habits coded as [0,1], exclusively asexual as [1,0], and hermaphrodites as [1,1]. The `constraint` argument allows additional control of such combinations. Setting `constraint = 2` only allows a maximum of "two-presence" combinations (e.g., [1,0,0], [0,1,0], [0,0,1], [1,1,0], [1,0,1], and [0,1,1]) as state combinations, but excludes [1,1,1]; setting `constraint = 1` only allows the first three of these combinations; the default behavior (`Inf`) allows all of these combinations. In all cases, the nonsensical "all-absence" state combination [0,0,0] is disallowed.

Character states can be weighted using the optional `weight.file`. This is useful so that random draws of life habits (functional-trait combinations) from the ecospace framework are biased in specified ways. If not provided, the default action assigns equal weighting among states. If a vector of mode array, integer, or numeric is provided, character states (or character-state combinations, if multistate binary) are assigned the specified relative weight. The function returns an error if the supplied vector has a length different that the number of states allowed.

If a data frame is supplied for the weight file (such as a species-by-trait matrix, with species as rows and traits as columns, describing a regional species pool), weights are calculated according to the observed relative frequency of states in this pool. If such a data frame is supplied, the first three columns must be (1) class [or similar taxonomic identifier], (2) genus, and (3) species names, although these can be left blank. In all cases, character state probabilities are calculated separately within each character (including only those allowed by the specified `constraint`).

Value

Returns a list of class `ecospace` describing the structure of the theoretical ecospace framework needed for running simulations. The list has a length equal to `nchar + 1`, with one list component for each character, plus a final list component recording constraints used in producing allowable character states.

Each character component has the following list components:

`char` character name.

`type` character type.

`char.space` data frame listing each allowable state combination in each row, the calculated proportional weight (`pro`), frequency (`n`) of observed species with such state combination in species pool (`weight.file`, if supplied).

`allowed.combos` allowed character state combinations allowed by `constraint` and `weight.file`, if supplied.

The last component lists the following components:

`constraint` `constraint` argument used.

`wts` vector of character-state weights used.

`pool` species by trait matrix used in assigning character-state weights, if supplied. Note that this matrix may differ from that supplied as `weight.file` when, for example, the supplied file includes character-state combinations not allowed by `constraint`]. It also excludes taxonomic identifiers (class, genus, species).

Note

If you have trouble matching the characters with `char.state` and `char.type`, see `data.frame` in first example for easy way to trouble-shoot. If you have trouble supplying correct length of `char.name`, `state.name` and `weight.file`, consider producing an ecospace framework with defaults first, then using these to supply custom names and weights.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bambach, R. K. 1983. Ecospace utilization and guilds in marine communities through the Phanerozoic. Pp. 719-746. *In* M. J. S. Tevesz, and P. L. McCall, eds. *Biotic Interactions in Recent and Fossil Benthic Communities*. Plenum, New York.

Bambach, R. K. 1985. Classes and adaptive variety: the ecology of diversification in marine faunas through the Phanerozoic. Pp. 191-253. *In* J. W. Valentine, ed. *Phanerozoic Diversity Patterns: Profiles in Macroevolution*. Princeton University Press, Princeton, NJ.

Bambach, R. K., A. M. Bush, and D. H. Erwin. 2007. Autecology and the filling of ecospace: key metazoan radiations. *Palaeontology* 50(1):1-22.

Bush, A. M. and R. K. Bambach. 2011. Paleoecologic megatrends in marine Metazoa. *Annual Review of Earth and Planetary Sciences* 39:241-269.

Bush, A. M., R. K. Bambach, and G. M. Daley. 2007. Changes in theoretical ecospace utilization in marine fossil assemblages between the mid-Paleozoic and late Cenozoic. *Paleobiology* 33(1):76-97.

Bush, A. M., R. K. Bambach, and D. H. Erwin. 2011. Ecospace utilization during the Ediacaran radiation and the Cambrian eco-explosion. Pp. 111-134. *In* M. Laflamme, J. D. Schiffbauer, and S. Q. Dornbos, eds. *Quantifying the Evolution of Early Life: Numerical Approaches to the Evaluation of Fossils and Ancient Ecosystems*. Springer, New York.

Novack-Gottshall, P.M. 2007. Using a theoretical ecospace to quantify the ecological diversity of Paleozoic and modern marine biotas. *Paleobiology* 33: 274-295.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

Examples

```
# Create random ecospace framework with all character types
set.seed(88)
nchar <- 10
char.state <- rpois(nchar, 1) + 2
char.type <- replace(char.state, char.state <= 3, "numeric")
char.type <- replace(char.type, char.state == 4, "ord.num")
char.type <- replace(char.type, char.state == 5, "ord.fac")
char.type <- replace(char.type, char.state > 5, "factor")
# Good practice to confirm everything matches expectations:
data.frame(char = seq(nchar), char.state, char.type)
ecospace <- create_ecospace(nchar, char.state, char.type, constraint = Inf)
ecospace

# How many life habits in this ecospace are theoretically possible?
seq <- seq(nchar)
prod(sapply(seq, function(seq) length(ecospace[[seq]]$allowed.combos)))
# ~12 million

# Observe effect of constraint for binary characters
create_ecospace(1, 4, "numeric", constraint = Inf)[[1]]$char.space
create_ecospace(1, 4, "numeric", constraint = 2)[[1]]$char.space
create_ecospace(1, 4, "numeric", constraint = 1)[[1]]$char.space
try(create_ecospace(1, 4, "numeric", constraint = 1.5)[[1]]$char.space) # ERROR!
try(create_ecospace(1, 4, "numeric", constraint = 0)[[1]]$char.space) # ERROR!

# Using custom-weighting for traits (singletons weighted twice as frequent
#   as other state combinations)
weight.file <- c(rep(2, 3), rep(1, 3), 2, 2, 1, rep(1, 4), rep(2, 3), rep(1, 3),
rep(1, 14), 2, 2, 1, rep(1, 4), rep(2, 3), rep(1, 3), rep(1, 5))
create_ecospace(nchar, char.state, char.type, constraint = 2,
  weight.file = weight.file)

# Bambach's (1983, 1985) classic ecospace framework
# 3 characters, all factors with variable states
nchar <- 3
char.state <- c(3, 4, 4)
char.type <- c("ord.fac", "factor", "factor")
char.names <- c("Tier", "Diet", "Activity")
state.names <- c("Pelag", "Epif", "Inf", "SuspFeed", "Herb", "Carn", "DepFeed",
  "Mobile/ShallowActive", "AttachLow/ShallowPassive", "AttachHigh/DeepActive",
  "Recline/DeepPassive")
ecospace <- create_ecospace(nchar, char.state, char.type, char.names, state.names)
ecospace
seq <- seq(nchar)
prod(sapply(seq, function(seq) length(ecospace[[seq]]$allowed.combos)))
# 48 possible life habits

# Bush and Bambach's (Bambach et al. 2007, bush et al. 2007) updated ecospace
#   framework, with Bush et al. (2011) and Bush and Bambach (2011) addition of
#   osmotrophy as a possible diet category
#   3 characters, all factors with variable states
```

```
nchar <- 3
char.state <- c(6, 6, 7)
char.type <- c("ord.fac", "ord.fac", "factor")
char.names <- c("Tier", "Motility", "Diet")
state.names <- c("Pelag", "Erect", "Surfic", "Semi-inf", "ShallowInf", "DeepInf",
  "FastMotile", "SlowMotile ", "UnattachFacMot", "AttachFacMot", "UnattachNonmot",
  "AttachNonmot", "SuspFeed", "SurfDepFeed", "Mining", "Grazing", "Predation",
  "Absorpt/Osmotr", "Other")
ecospace <- create_ecospace(nchar, char.state, char.type, char.names, state.names)
ecospace
seq <- seq(nchar)
prod(sapply(seq, function(seq) length(ecospace[[seq]]$allowed.combos)))
# 252 possible life habits

# Novack-Gottshall (2007) ecospace framework, updated in Novack-Gottshall (2016b)
#   Fossil species pool from Late Ordovician (Type Cincinnatian) Kope and
#   Waynesville Formations, with functional-trait characters coded according
#   to Novack-Gottshall (2007, 2016b)
data(KWTraits)
head(KWTraits)
nchar <- 18
char.state <- c(2, 7, 3, 3, 2, 2, 5, 5, 2, 5, 2, 2, 5, 2, 5, 5, 3, 3)
char.type <- c("numeric", "ord.num", "numeric", "numeric", "numeric", "numeric",
  "ord.num", "ord.num", "numeric", "ord.num", "numeric", "numeric", "ord.num",
  "numeric", "ord.num", "numeric", "numeric", "numeric")
char.names <- c("Reproduction", "Size", "Substrate composition", "Substrate
  consistency", "Supported", "Attached", "Mobility", "Absolute tier", "Absolute
  microhabitat", "Relative tier", "Relative microhabitat", "Absolute food
  microhabitat", "Absolute food tier", "Relative food microhabitat", "Relative
  food tier", "Feeding habit", "Diet", "Food condition")
state.names <- c("SEXL", "ASEX", "BVOL", "BIOT", "LITH", "FLUD", "HARD", "SOFT",
  "INSB", "SPRT", "SSUP", "ATTD", "FRLV", "MOBL", "ABST", "AABS", "IABS", "RLST",
  "AREL", "IREL", "FAAB", "FIAB", "FAST", "FARL", "FIRL", "FRST", "AMBT", "FILT",
  "ATTF", "MASS", "RAPT", "AUTO", "MICR", "CARN", "INCP", "PART", "BULK")
ecospace <- create_ecospace(nchar, char.state, char.type, char.names, state.names,
  constraint = 2, weight.file = KWTraits)
ecospace
seq <- seq(nchar)
prod(sapply(seq, function(seq) length(ecospace[[seq]]$allowed.combos)))
# ~57 billion life habits

ecospace <- create_ecospace(nchar, char.state, char.type, char.names, state.names,
  constraint = Inf)
ecospace
seq <- seq(nchar)
prod(sapply(seq, function(seq) length(ecospace[[seq]]$allowed.combos)))
# ~3.6 trillion life habits
```

`expansion` *Use Expansion Rule to Simulate Ecological Diversification of a Biota.*

Description

Implement Monte Carlo simulation of a biota undergoing ecological diversification using the expansion rule.

Usage

```
expansion(nreps = 1, Sseed, Smax, ecospace, method = "Euclidean", strength = 1)
```

Arguments

nreps	Vector of integers (such as a sequence) specifying sample number produced. Only used when function is applied within lapply or related function. Default nreps=1 or any other integer produces a single sample.
Sseed	Integer giving number of species (or other taxa) to use at start of simulation.
Smax	Maximum number of species (or other taxa) to include in simulation.
ecospace	An ecospace framework (functional trait space) of class ecospace.
method	Distance measure to use when calculating functional distances between species. Default is Euclidean using stats::dist. Gower or any other value uses Gower distance (using FD::gowdis). Presence of factor or ordered factor character types forces use of Gower distance.
strength	Strength parameter controlling probability that expansion rule is followed during simulation. Values must range between strength = 1 (default, rules always implemented) and strength = 0 (rules never implemented).

Details

Simulations are implemented as Monte Carlo processes in which species are added iteratively to assemblages, with all added species having their character states specified by the model rules, here the 'expansion' rule. Simulations begin with the seeding of Sseed number of species, chosen at random (with replacement) from either the species pool (if provided in the weight.file when building the ecospace framework using create_ecospace) or following the neutral-rule algorithm (if a pool is not provided). Once seeded, the simulations proceed iteratively (character-by-character, species-by-species) by following the appropriate algorithm, as explained below, until terminated at Smax.

Expansion rule algorithm: Measure distances between all pairs of species, using Euclidean or Gower distance method specified by method argument. Identify species pair that is maximally distant. If multiple pairs are equally maximally distant, one pair is chosen at random. The newly added species has traits that are equal to or more extreme than those in this species pair, with probability of following the expansion rule determined by the strength parameter. Default strength = 1 always implements the rule, whereas strength = 0 never implements it (essentially making the simulation follow the neutral rule.)

Each newly assigned character is compared with the ecospace framework (ecospace) to confirm that it is an allowed state combination before proceeding to the next character. If the newly built character is disallowed from the ecospace framework (i.e., because it has "dual absences" [0,0], has been excluded based on the species pool [weight.file in create_ecospace], or is not allowed by the ecospace constraint parameter), then the character-selection algorithm is repeated until

an allowable character is selected. When simulations proceed to very large sample sizes (>100), this confirmatory process can require long computational times, and produce "new" species that are functionally identical to pre-existing species. This can occur, for example, when no life habits, or perhaps only one, exist that forms an allowable novelty between the selected neighbors.

Expansion rules tend to produce ecospaces that progressively expand into more novel regions. Additional details on the expansion simulation are provided in Novack-Gottshall (2016a,b), including sensitivity to ecospace framework (functional trait space) structure, recommendations for model selection, and basis in ecological and evolutionary theory.

Value

Returns a data frame with `Smax` rows (representing species) and as many columns as specified by number of characters/states (functional traits) in the ecospace framework. Columns will have the same data type (numeric, factor, ordered numeric, or ordered factor) as specified in the ecospace framework.

Note

A bug exists within `FD::gowdis` where nearest-neighbor distances can not be calculated when certain characters (especially numeric characters with values other than 0 and 1) share identical traits across species. The nature of the bug is under investigation, but the current implementation is reliable under most uses. If you run into problems because of this bug, a work-around is to manually change the function to call `cluster::daisy` using `metric = "gower"` instead.

The function has been written to allow usage (using `lapply` or some other list-apply function) in 'embarrassingly parallel' implementations in a high-performance computing environment.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bush, A. and P.M. Novack-Gottshall. 2012. Modelling the ecological-functional diversification of marine Metazoa on geological time scales. *Biology Letters* 8: 151-155.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

See Also

`create_ecospace`, `neutral`, `redundancy`, `partitioning`

Examples

```
# Create an ecospace framework with 15 3-state factor characters
# Can also accept following character types: "numeric", "ord.num", "ord.fac"
nchar <- 15
ecospace <- create_ecospace(nchar = nchar, char.state = rep(3, nchar),
```

```
  char.type = rep("factor", nchar))

# Single (default) sample produced by expansion function (with strength = 1):
Sseed <- 5
Smax <- 40
x <- expansion(Sseed = Sseed, Smax = Smax, ecospace = ecospace)
head(x, 10)

# Plot results, showing order of assembly
# (Seed species in red, next 5 in black, remainder in gray)
# Notice that new life habits progressively expand outward into previously
#   unoccupied portions of ecospace
seq <- seq(nchar)
types <- sapply(seq, function(seq) ecospace[[seq]]$type)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Expansion model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Change strength parameter so rules followed 95% of time:
x <- expansion(Sseed = Sseed, Smax = Smax, ecospace = ecospace, strength = 0.95)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Expansion model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Create 4 samples using multiple nreps and lapply (can be slow)
nreps <- 1:4
samples <- lapply(X = nreps, FUN = expansion, Sseed = 5, Smax = 40, ecospace)
str(samples)
```

`KWTraits` *Species-by-Trait Matrix for Late Ordovician Marine Fossils.*

Description

Sample data set of life habit codings (functional traits) for fossil taxa from the Late Ordovician (Type Cincinnatian) Kope and Waynesville Formations from Ohio, Indiana, and Kentucky (U.S.A.). The faunal list was compiled from the Paleobiology Database.

Usage

```
KWTraits
```

Format

A data frame with 237 rows (taxa) and 40 columns (3 taxonomic identifiers and 37 functional traits):

Class Taxonomic class(character)

Genus Taxonomic genus (character)

sp. Taxonomic species (character)

SEXL Sexual reproduction (binary)

ASEX Asexual reproduction (binary)

BVOL Skeletal body volume of typical adult (ordered numeric with 7 bins). Estimated using methods of Novack-Gottshall (2008).

- 1.000: >= 100 cm^3
- 0.833: 100-10 cm^3
- 0.667: 10-1 cm^3
- 0.500: 1-0.1 cm^3
- 0.333: 0.1-0.01 cm^3
- 0.167: 0.01-0.001 cm^3
- 0: < 0.001 cm^3

BIOT Biotic substrate composition (binary)

LITH Lithic substrate composition (binary)

FLUD Fluidic medium (binary)

HARD Hard substrate consistency (binary)

SOFT Soft substrate consistency (binary)

INSB Insubstantial medium consistency (binary)

SPRT Supported on other object (binary)

SSUP Self-supported (binary)

ATTD Attached to substrate (binary)

FRLV Free-living (binary)

MOBL Mobility (ordered numeric with 5 bins):

- 1: habitually mobile
- 0.75: intermittently mobile
- 0.50: facultatively mobile
- 0.25: passively mobile (i.e., planktonic drifting)
- 0: sedentary (immobile)

ABST Primary microhabitat stratification: absolute distance from seafloor (ordered numeric with 5 bins):

- 1: >= 100 cm
- 0.75: 100-10 cm:
- 0.50: 10-1 cm
- 0.25: 1-0.1 cm
- 0: <0.1 cm

AABS Primary microhabitat is above seafloor (i.e., epifaunal)

IABS Primary microhabitat is within seafloor (i.e., infaunal)

RLST Immediately surrounding microhabitat stratification: relative distance from substrate (ordered numeric with 5 bins):

- 1: >= 100 cm
- 0.75: 100-10 cm:
- 0.50: 10-1 cm
- 0.25: 1-0.1 cm
- 0: <0.1 cm

AREL Lives above immediate substrate

IREL Lives within immediate substrate

FAAB Food is above seafloor

FIAB Food is within seafloor

FAST Primary feeding microhabitat stratification: absolute distance of food from seafloor (ordered numeric with 5 bins):

- 1: >= 100 cm
- 0.75: 100-10 cm:
- 0.50: 10-1 cm
- 0.25: 1-0.1 cm
- 0: <0.1 cm

FARL Food is above immediate substrate

FIRL Food is within immediate substrate

FRST Immediately surrounding feeding microhabitat stratification: relative distance of food from substrate (ordered numeric with 5 bins):

- 1: >= 100 cm
- 0.75: 100-10 cm:
- 0.50: 10-1 cm
- 0.25: 1-0.1 cm
- 0: <0.1 cm

AMBT Ambient foraging habit

FILT Filter-feeding foraging habit

ATTF Attachment-feeding foraging habit

MASS Mass-feeding foraging habit

RAPT Raptorial foraging habit

AUTO Autotrophic diet

MICR Microbivorous (bacteria, protists, algae) diet

CARN Carnivorous diet

INCP Food has incorporeal physical condition

PART Food consumed as particulate matter

BULK Food consumed as bulk matter

Details

Binary traits are coded with 0 = absent and 1 = present. Five ordered numeric traits (body volume, mobility, distance from seafloor [stratification]) were rescaled to range from 0 to 1 with discrete bins at equally spaced intermediate values.

See Novack-Gottshall (2007: especially online Supplementary Appendix A at for reprint) for definition each functional trait, justifications, explanations, and examples. Novack-Gottshall (2007: Supplementary Appendix B; 2016: Supplementary Appendix A) provides examples of how traits were coded using inferences derived from functional morphology, body size, ichnology, *in situ* preservation, biotic associations recording direct interactions, and interpretation of geographic and depositional environment patterns.

Indeterminate taxa (e.g., trepostome bryozoan indet. or *Platystrophia* sp.) that occurred within individual samples within these formations were excluded from the aggregate species pool unless their occurrence was the sole member of that taxon. Such indeterminate taxa and genera lacking a species identification were coded for a particular state only when all other members of that taxon within the Kope-Waynesville species pool unanimously shared that common state; otherwise, the state was listed as NA (missing).

Source

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

References

Novack-Gottshall, P.M. 2007. Using a theoretical ecospace to quantify the ecological diversity of Paleozoic and modern marine biotas. *Paleobiology* 33: 274-295.

Novack-Gottshall, P.M. 2008. Using simple body-size metrics to estimate fossil body volume: empirical validation using diverse Paleozoic invertebrates. *PALAIOS* 23(3):163-173.

Novack-Gottshall, P.M. 2016. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

Villeger, S., P. M. Novack-Gottshall, and D. Mouillot. 2011. The multidimensionality of the niche reveals functional diversity changes in benthic marine biotas across geological time. *Ecology Letters* 14(6):561-568.

`neutral` *Use Neutral Rule to Simulate Ecological Diversification of a Biota.*

Description

Implement Monte Carlo simulation of a biota undergoing ecological diversification using the neutral rule. Can be used as a simple permutation test (draw species at random with replacement from provided species pool) if set `Sseed` equal to `Smax`.

Usage

```
neutral(nreps = 1, Sseed, Smax, ecospace)
```

Arguments

nreps	Vector of integers (such as a sequence) specifying sample number produced. Only used when function is applied within lapply or related function. Default nreps = 1 or any other integer produces a single sample.
Sseed	Integer giving number of species (or other taxa) to use at start of simulation.
Smax	Maximum number of species (or other taxa) to include in simulation.
ecospace	An ecospace framework (functional trait space) of class ecospace.

Details

Simulations are implemented as Monte Carlo processes in which species are added iteratively to assemblages, with all added species having their character states specified by the model rules, here the 'neutral' rule. Simulations begin with the seeding of Sseed number of species, chosen at random (with replacement) from either the species pool (if provided in the weight.file when building the ecospace framework using create_ecospace) or following the neutral-rule algorithm (if a pool is not provided). Once seeded, the simulations proceed iteratively (character-by-character, species-by-species) by following the appropriate algorithm, as explained below, until terminated at Smax.

Neutral rule algorithm: Choose remaining species (or seed species, if no pool) as random multinomial draws from theoretical ecospace framework (using whatever constraints and structure was provided by the ecospace framework in create_ecospace). Thus, if relative weighting was provided to character states (functional traits), simulated species will mimic these weights, on average. If state combinations were constrained (by setting constraint in create_ecospace), then unallowed state combinations will not be allowed in simulated species.

Note that this simulation is not a simple permutation test of a species pool (if provided). The life habit of each new species is built character-by-character from the realm of theoretically possible states allowed by the ecospace framework. Simulated species can occupy combinations of character states that did not occur in the species pool (if provided). This is an important feature of the simulations, allowing the entire theoretical ecospace to be explored by the neutral model. However, the simulation can be used as a simple permutation test (draw species at random with replacement from provided species pool) if set Sseed equal to Smax and a species pool is supplied when building the ecospace framework.

This rule has also been termed the diffusional, null, and passive model (Bush and Novack-Gottshall 2012). Additional details on the neutral simulation are provided in Novack-Gottshall (2016a,b), including sensitivity to ecospace framework (functional trait space) structure, recommendations for model selection, and basis in ecological and evolutionary theory.

Value

Returns a data frame with Smax rows (representing species) and as many columns as specified by number of characters/states (functional traits) in the ecospace framework. Columns will have the same data type (numeric, factor, ordered numeric, or ordered factor) as specified in the ecospace framework.

Note

The function has been written to allow usage (using lapply or some other list-apply function) in 'embarrassingly parallel' implementations in a high-performance computing environment.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bush, A. and P.M. Novack-Gottshall. 2012. Modelling the ecological-functional diversification of marine Metazoa on geological time scales. *Biology Letters* 8: 151-155.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

See Also

create_ecospace, redundancy, partitioning, expansion

Examples

```
# Create an ecospace framework with 15 3-state factor characters
# Can also accept following character types: "numeric", "ord.num", "ord.fac"
nchar <- 15
ecospace <- create_ecospace(nchar = nchar, char.state = rep(3, nchar),
  char.type = rep("factor", nchar))

# Single (default) sample produced by neutral function:
Sseed <- 5
Smax <- 50
x <- neutral(Sseed = Sseed, Smax = Smax, ecospace = ecospace)
head(x, 10)

# Plot results, showing order of assembly
# (Seed species in red, next 5 in black, remainder in gray)
# Notice the neutral model fills the entire ecospace with life habits
seq <- seq(nchar)
types <- sapply(seq, function(seq) ecospace[[seq]]$type)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Neutral model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Create 5 samples using multiple nreps and lapply
nreps <- 1:5
samples <- lapply(X = nreps, FUN = neutral, Sseed = 5, Smax = 50, ecospace)
str(samples)

# Implement as simple permutation test by setting Sseed = Smax and providing species pool)
nchar <- 18
char.state <- c(2, 7, 3, 3, 2, 2, 5, 5, 2, 5, 2, 2, 5, 2, 5, 5, 3, 3)
char.type <- c("numeric", "ord.num", "numeric", "numeric", "numeric", "numeric",
```

```
  "ord.num", "ord.num", "numeric", "ord.num", "numeric", "numeric", "ord.num",
  "numeric", "ord.num", "numeric", "numeric", "numeric")
data(KWTraits)
ecospace <- create_ecospace(nchar, char.state, char.type, constraint = 2,
  weight.file = KWTraits)

x <- neutral(Sseed = 100, Smax = 100, ecospace = ecospace)
mean(dist(x))

# Note ecological disparity (functional diversity) is less when perform permutation
x <- neutral(Sseed = 5, Smax = 100, ecospace = ecospace)
mean(dist(x))

# Simulated character states (functional traits) proportionally mimic those in species pool
x <- neutral(Sseed = 5, Smax = 234, ecospace = ecospace)
table(x[,1:2])
table(KWTraits$SEXL, KWTraits$ASEX)
```

`partitioning` *Use Partitioning Rule to Simulate Ecological Diversification of a Biota.*

Description

Implement Monte Carlo simulation of a biota undergoing ecological diversification using the partitioning rule.

Usage

```
partitioning(
  nreps = 1,
  Sseed,
  Smax,
  ecospace,
  method = "Euclidean",
  rule = "strict",
  strength = 1
)
```

Arguments

`nreps`	Vector of integers (such as a sequence) specifying sample number produced. Only used when function is applied within `lapply` or related function. Default `nreps=1` or any other integer produces a single sample.
`Sseed`	Integer giving number of species (or other taxa) to use at start of simulation.
`Smax`	Maximum number of species (or other taxa) to include in simulation.
`ecospace`	An ecospace framework (functional trait space) of class `ecospace`.

method	Distance measure to use when calculating functional distances between species. Default is Euclidean using stats::dist. Gower or any other value uses Gower distance (using FD::gowdis). Presence of factor or ordered factor character types forces use of Gower distance.
rule	The partitioning implementation to use in the simulation. Default strict implements the 'minimum distant neighbor' rule; relaxed implements the "maximum nearest neighbor" rule. See 'Details' below for further explanation.
strength	Strength parameter controlling probability that partitioning rule is followed during simulation. Values must range between strength = 1 (default, rules always implemented) and strength = 0 (rules never implemented).

Details

Simulations are implemented as Monte Carlo processes in which species are added iteratively to assemblages, with all added species having their character states specified by the model rules, here the 'partitioning' rule. Simulations begin with the seeding of Sseed number of species, chosen at random (with replacement) from either the species pool (if provided in the weight.file when building the ecospace framework using create_ecospace) or following the neutral-rule algorithm (if a pool is not provided). Once seeded, the simulations proceed iteratively (character-by-character, species-by-species) by following the appropriate algorithm, as explained below, until terminated at Smax.

Partitioning rule algorithm: Measure distances between all pairs of species, using Euclidean or Gower distance method specified by method argument. Use either of the following rules to identify the position of each additional species.

strict **(minimum distant neighbor) rule** Identify the maximum distances between all pairs of species (the most-distant neighbors); the space to be partitioned is the minimum of these distances. This implementation progressively fills in the largest parts of the ecospace that are least occupied between neighboring species, and eventually partitions the ecospace in straight-line gradients between seed species.

relaxed **(maximum nearest neighbor) rule** Identify nearest-neighbor distances between all pairs of species; the space to be partitioned is the maximum of these distances. This implementation places new species in the most unoccupied portion of the ecospace that is within the cluster of pre-existing species, often the centroid.

In both rules, each new species is created as a resampled combination of the character states of the identified neighbors. If multiple pairs meet the specific criteria, one of these pairs is chosen at random. Ordered, multistate character partitioning (such as ordered factors or order numeric character types) can include any state equal to or between the observed states of existing species. The probability of following the partitioning rule is determined by the strength parameter. Default strength = 1 always implements the rule, whereas strength = 0 never implements it (essentially making the simulation follow the neutral rule.)

Each newly assigned character is compared with the ecospace framework (ecospace) to confirm that it is an allowed state combination before proceeding to the next character. If the newly built character is disallowed from the ecospace framework (i.e., because it has "dual absences" [0,0], has been excluded based on the species pool [weight.file in create_ecospace], or is not allowed by the ecospace constraint parameter), then the character-selection algorithm is repeated until an allowable character is selected. When simulations proceed to very large sample sizes (>100),

this confirmatory process can require long computational times, and produce "new" intermediate species that are functionally identical to pre-existing species. This can occur, for example, when no life habits, or perhaps only one, exist that forms an allowable intermediate between the selected neighbors.

Partitioning rules tend to produce ecospaces displaying linear gradients between seed species (in the `strict` implementation) or concentration of life habits near the functional centroid (in the `relaxed` implementation). Additional details on the partitioning simulation are provided in Novack-Gottshall (2016a,b), including sensitivity to ecospace framework (functional trait space) structure, recommendations for model selection, and basis in ecological and evolutionary theory.

Value

Returns a data frame with `Smax` rows (representing species) and as many columns as specified by number of characters/states (functional traits) in the ecospace framework. Columns will have the same data type (numeric, factor, ordered numeric, or ordered factor) as specified in the ecospace framework.

Note

A bug exists within `FD::gowdis` where nearest-neighbor distances can not be calculated when certain characters (especially numeric characters with values other than 0 and 1) share identical traits across species. The nature of the bug is under investigation, but the current implementation is reliable under most uses. If you run into problems because of this bug, a work-around is to manually change the function to call `cluster::daisy` using `metric = "gower"` instead.

The function has been written to allow usage (using `lapply` or some other list-apply function) in 'embarrassingly parallel' implementations in a high-performance computing environment.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bush, A. and P.M. Novack-Gottshall. 2012. Modelling the ecological-functional diversification of marine Metazoa on geological time scales. *Biology Letters* 8: 151-155.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

See Also

`create_ecospace`, `neutral`, `redundancy`, `expansion`

Examples

```
# Create an ecospace framework with 15 3-state factor characters
# Can also accept following character types: "numeric", "ord.num", "ord.fac"
nchar <- 15
```

```
ecospace <- create_ecospace(nchar = nchar, char.state = rep(3, nchar),
  char.type = rep("factor", nchar))

# Single (default) sample produced by partitioning function (with strength = 1 and
#   "strict" partitioning rules):
Sseed <- 5
Smax <- 40
x <- partitioning(Sseed = Sseed, Smax = Smax, ecospace = ecospace, rule = "strict")
head(x, 10)

# Plot results, showing order of assembly
# (Seed species in red, next 5 in black, remainder in gray)
# Notice the 'strict' partitioning model produces an ecospace with life-habit gradients
#   between seed species
seq <- seq(nchar)
types <- sapply(seq, function(seq) ecospace[[seq]]$type)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Partitioning model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Same, but following "relaxed" partitioning rules:
# Notice the 'relaxed' partitioning model only fills in the ecospace between seed species
x <- partitioning(Sseed = Sseed, Smax = Smax, ecospace = ecospace, rule = "relaxed")
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Partitioning model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Change strength parameter so rules followed 95% of time:
x <- partitioning(Sseed = Sseed, Smax = Smax, ecospace = ecospace, strength = 0.95, rule = "strict")
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Partitioning model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Create 5 samples using multiple nreps and lapply (can be slow)
nreps <- 1:5
samples <- lapply(X = nreps, FUN = partitioning, Sseed = 5, Smax = 40, ecospace)
str(samples)
```

---

prep_data *Internal Ecospace Functions.*

Description

Internal functions not intended to be called directly by users.

Usage

```
prep_data(ecospace, Smax)
```

Arguments

ecospace	An ecospace framework (functional trait space) of class ecospace.
Smax	Maximum number of species (or other taxa) to include in simulation.

Details

Pre-allocate the basic data frame structure so classes (and factor levels, if used) are properly set. Bit clunky but efficient solution: adding first placeholder column allows cbind to work properly with data frame later, but requires deleting that placeholder at end.

Value

Returns an empty data frame with pre-defined number of species and functional traits of proper class type.

See Also

create_ecospace for how to create an ecospace framework (functional trait space).

---

rbind_listdf *Combine Large Lists of Data Frames.*

Description

Quickly combine large lists of dataframes into a single data frame by combining them first into smaller data frames. This is only time- and memory-efficient when dealing with large (>100) lists of data frames.

Usage

```
rbind_listdf(lists = NULL, seq.by = 100)
```

Arguments

lists	List in which each component is a data frame. Each data frame must have the same number and types of columns, although the data frames can have different numbers of rows.
seq.by	Length of small data frames to work with at a time. Default is 100, which timing tests confirm is generally most efficient.

Details

Rather than combine all list data frames into a single data frame, this function builds smaller subsets of data frames, and then combines them together at the end. This process is more time- and memory-efficient. Timing tests confirm that seq.by = 100 is the optimal break-point size. See examples for confirmation. Function can break large lists into up to 456,976 data frame subparts, giving a warning if requires more subparts. Only time- and memory-efficient when dealing with large (>100) lists of data frames.

Value

Single data frame with number of rows equal to the sum of all data frames in the list, and columns the same as those of individual list data frames.

Note

When called within `lapply`, the simulation functions `neutral`, `redundancy`, `partitioning`, `expansion`, and `calc_metrics`, which calculates common ecological disparity (functional diversity) statistics on simulated biotas, produce lists of data frames. This function is useful for combining these separate lists into a single data frame for subsequent analyses. This is especially useful when using the functions within a high-performance computing environment when submitted as 'embarrassingly parallel' implementations. `rbindlist` is a more efficient version of this function.

See Also

`neutral`, `redundancy`, `partitioning`, `expansion`, `calc_metrics`, and `rbindlist`

Examples

```
nl <- 500     # List length
lists <- vector("list", length = nl)
for(i in 1:nl) lists[[i]] <- list(x = rnorm(100), y = rnorm(100))
str(lists)
object.size(lists)
all <- rbind_listdf(lists)
str(all)
object.size(all)       # Also smaller object size

# Build blank ecospace framework to use in simulations
ecospace <- create_ecospace(nchar = 15, char.state = rep(3, 15),
                            char.type = rep("numeric", 15))

# Build 5 samples for neutral model:
nreps <- 1:5
Smax <- 10
n.samples <- lapply(X = nreps, FUN = neutral, Sseed = 3, Smax = Smax, ecospace)

# Calculate functional diversity metrics for simulated samples
n.metrics <- lapply(X = nreps, FUN = calc_metrics, samples = n.samples,
                    Model = "neutral", Param = "NA")
alarm()
str(n.metrics)
```

```
# rbind lists together into a single dataframe
all <- rbind_listdf(n.metrics)

# Calculate mean dynamics
means <- n.metrics[[1]]
for(n in 1:Smax) means[n,4:11] <- apply(all[which(all$S == means$S[n]),4:11],
                                        2, mean, na.rm = TRUE)
means

# Plot statistics as function of species richness, overlaying mean dynamics
op <- par()
par(mfrow = c(2,4), mar = c(4, 4, 1, .3))
attach(all)

plot(S, H, type = "p", cex = .75, col = "gray")
lines(means$S, means$H, type = "l", lwd = 2)
plot(S, D, type = "p", cex = .75, col = "gray")
lines(means$S, means$D, type = "l", lwd = 2)
plot(S, M, type = "p", cex = .75, col = "gray")
lines(means$S, means$M, type = "l", lwd = 2)
plot(S, V, type = "p", cex = .75, col = "gray")
lines(means$S, means$V, type = "l", lwd = 2)
plot(S, FRic, type = "p", cex = .75, col = "gray")
lines(means$S, means$FRic, type = "l", lwd = 2)
plot(S, FEve, type = "p", cex = .75, col = "gray")
lines(means$S, means$FEve, type = "l", lwd = 2)
plot(S, FDiv, type = "p", cex = .75, col = "gray")
lines(means$S, means$FDiv, type = "l", lwd = 2)
plot(S, FDis, type = "p", cex = .75, col = "gray")
lines(means$S, means$FDis, type = "l", lwd = 2)

par(op)

## Not run:
# Note each of following can take a few seconds to run
# Compare timings:
t0 <- Sys.time()
all <- rbind_listdf(lists)
(Sys.time() - t0)

t0 <- Sys.time()
all <- rbind_listdf(lists, seq.by = 20)
(Sys.time() - t0)

t0 <- Sys.time()
all <- rbind_listdf(lists, seq.by = 500)
(Sys.time() - t0)

# Compare to non-function version
all2 <- data.frame()
t0 <- Sys.time()
for(i in 1:nl) all2 <- rbind(all2, lists[[i]])
```

```
(Sys.time() - t0)

## End(Not run)

# Compare to data.table's 'rbindlist' version
library(data.table)
t0 <- Sys.time()
all <- rbindlist(lists)
(Sys.time() - t0)
```

---

redundancy *Use Redundancy Rule to Simulate Ecological Diversification of a Biota.*

Description

Implement Monte Carlo simulation of a biota undergoing ecological diversification using the redundancy rule.

Usage

```
redundancy(nreps = 1, Sseed, Smax, ecospace, strength = 1)
```

Arguments

nreps	Vector of integers (such as a sequence) specifying sample number produced. Only used when function is applied within `lapply` or related function. Default `nreps = 1` or any other integer produces a single sample.
Sseed	Integer giving number of species (or other taxa) to use at start of simulation.
Smax	Maximum number of species (or other taxa) to include in simulation.
ecospace	An ecospace framework (functional trait space) of class `ecospace`.
strength	Strength parameter controlling probability that redundancy rule is followed during simulation. Values must range between `strength = 1` (default, rules always implemented) and `strength = 0` (rules never implemented).

Details

Simulations are implemented as Monte Carlo processes in which species are added iteratively to assemblages, with all added species having their character states specified by the model rules, here the 'redundancy' rule. Simulations begin with the seeding of `Sseed` number of species, chosen at random (with replacement) from either the species pool (if provided in the `weight.file` when building the ecospace framework using `create_ecospace`) or following the neutral-rule algorithm (if a pool is not provided). Once seeded, the simulations proceed iteratively (character-by-character, species-by-species) by following the appropriate algorithm, as explained below, until terminated at `Smax`.

Redundancy rule algorithm: Pick one existing species at random and create a new species using that species' characters as a template. A character is modified (using a random multinomial draw from the ecospace framework) according to the `strength` parameter. Default `strength = 1` always implements the redundancy rule, whereas `strength = 0` never implements it (essentially making the simulation follow the `neutral` rule.) Because new character states can be any allowed by the ecospace framework, there is the possibility of obtaining redundancy greater than that specified by a strength parameter less than 1 (if, for example, the new randomly chosen character states are identical to those of the template species).

Redundancy rules tend to produce ecospaces with discrete clusters of functionally similar species. Additional details on the redundancy simulation are provided in Novack-Gottshall (2016a,b), including sensitivity to ecospace framework (functional trait space) structure, recommendations for model selection, and basis in ecological and evolutionary theory.

Value

Returns a data frame with `Smax` rows (representing species) and as many columns as specified by number of characters/states (functional traits) in the ecospace framework. Columns will have the same data type (numeric, factor, ordered numeric, or ordered factor) as specified in the ecospace framework.

Note

The function has been written to allow usage (using `lapply` or some other list-apply function) in 'embarrassingly parallel' implementations in a high-performance computing environment.

Author(s)

Phil Novack-Gottshall <pnovack-gottshall@ben.edu>

References

Bush, A. and P.M. Novack-Gottshall. 2012. Modelling the ecological-functional diversification of marine Metazoa on geological time scales. *Biology Letters* 8: 151-155.

Novack-Gottshall, P.M. 2016a. General models of ecological diversification. I. Conceptual synthesis. *Paleobiology* 42: 185-208.

Novack-Gottshall, P.M. 2016b. General models of ecological diversification. II. Simulations and empirical applications. *Paleobiology* 42: 209-239.

See Also

`create_ecospace`, `neutral`, `partitioning`, `expansion`

Examples

```
# Create an ecospace framework with 15 3-state factor characters
# Can also accept following character types: "numeric", "ord.num", "ord.fac"
nchar <- 15
ecospace <- create_ecospace(nchar = nchar, char.state = rep(3, nchar),
  char.type = rep("factor", nchar))
```

```
# Single (default) sample produced by redundancy function (with strength = 1):
Sseed <- 5
Smax <- 50
x <- redundancy(Sseed = Sseed, Smax = Smax, ecospace = ecospace)
head(x, 10)

# Plot results, showing order of assembly
# (Seed species in red, next 5 in black, remainder in gray)
# Notice the redundancy model produces an ecospace with discrete clusters of life habits
seq <- seq(nchar)
types <- sapply(seq, function(seq) ecospace[[seq]]$type)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Redundancy model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Change strength parameter so new species are 95% identical:
x <- redundancy(Sseed = Sseed, Smax = Smax, ecospace = ecospace, strength = 0.95)
if(any(types == "ord.fac" | types == "factor")) pc <- prcomp(FD::gowdis(x)) else
  pc <- prcomp(x)
plot(pc$x, type = "n", main = paste("Redundancy model,\n", Smax, "species"))
text(pc$x[,1], pc$x[,2], labels = seq(Smax), col = c(rep("red", Sseed), rep("black", 5),
  rep("slategray", (Smax - Sseed - 5))), pch = c(rep(19, Sseed), rep(21, (Smax - Sseed))),
  cex = .8)

# Create 5 samples using multiple nreps and lapply (can be slow)
nreps <- 1:5
samples <- lapply(X = nreps, FUN = redundancy, Sseed = 5, Smax = 50, ecospace)
str(samples)
```

`sample2` *Internal Ecospace Functions.*

Description

Internal functions not intended to be called directly by users.

Usage

```
sample2(x, ...)
```

Arguments

`x`	Either a vector of one or more elements from which to choose, or a positive integer.
`...`	arguments for particular methods.

Details

Corrects behavior when object sampled has unit length. This is the same function proposed in help file of `sample` as `resample`.

See Also

`sample`

`unique2` *Internal Ecospace Functions.*

Description

Internal functions not intended to be called directly by users.

Usage

```
unique2(x, length = 1, ...)
```

Arguments

`x`	a vector or a data frame or an array or NULL.
`length`	number of times (default = 1) to repeat the dimensionally shorter state.
`...`	arguments for particular methods.

Details

Modified `unique` function that returns proper number of items for filling of matrix when called within `apply`.

Note

When dealing with ecospace frameworks with multistate binary (numeric) character types and characters weighted by supplied species pools, sometimes all species will share the same state value for one of several states. (For example, perhaps all species are capable of sexual reproduction, but there is variation in whether some are exclusively sexual and some are hermaphroditic.) When this occurs, calling `apply` when choosing possible ecospace states will 'break the dimensionality' of the character matrix and convert it into a list. This function maintains matrix dimensionality, by repeating the dimensionally shorter unique state sufficient times to maintain equal length as found in other states.

See Also

`unique`

Index