Package 'divDyn'

March 2, 2021

Type Package

Title Diversity Dynamics using Fossil Sampling Data

Version 0.8.1

Author Adam T. Kocsis, John Alroy, Carl J. Reddin, Wolfgang Kiessling

Maintainer Adam T. Kocsis <adam.t.kocsis@gmail.com>

Description Functions to describe sampling and diversity dynamics of fossil occurrence datasets (e.g. from the Paleobiology Database). The package includes methods to calculate range- and occurrence-based metrics of taxonomic richness, extinction and origination rates, along with traditional sampling measures. A powerful subsampling tool is also included that implements frequently used sampling standardization methods in a multiple bin-framework. The plotting of time series and the occurrence data can be simplified by the functions incorporated in the package, as well other calculations, such as environmental affinities and extinction selectivity testing. Details can be found in: Kocsis, A.T.; Reddin, C.J.; Alroy, J. and Kiessling, W. (2019) <doi:10.1101/423780>.

License CC BY 4.0

Date 2021-03-01

BugReports https://github.com/divDyn/r_package/issues

Encoding UTF-8

LazyData false

Depends R (>= 3.5.0)

Imports Rcpp, stats, graphics, grDevices

NeedsCompilation yes

RoxygenNote 7.1.1

LinkingTo Rcpp

VignetteBuilder knitr

Suggests knitr, rmarkdown, vegan, icosa

Repository CRAN

Date/Publication 2021-03-02 08:20:03 UTC

R topics documented:

`affinity` *Environmental affinities of taxa*

Description

This function will return the preferred environment of the taxa, given the distribution of occurrences.

Usage

```
affinity(
  x,
  tax,
  bin,
  env,
  coll = NULL,
  method = "binom",
  alpha = 1,
  reldat = NULL,
  na.rm = FALSE,
  bycoll = FALSE
)
```

Arguments

x	(data.frame) The occurrence dataset containing the taxa with unknown environmental affinities.
tax	(character) The column name of taxon names.
bin	(character) The column name of bin names.
env	(character) The environmental variable of the occurrences.
coll	(character) The column name of collection identifiers (optional). If this is provided, then then the multiple entries of a taxon within the collections will be treated as 1.
method	(character) The method used for affinity calculations. Can be either "binom" or "majority".
alpha	(numeric) The alpha value of the binomial tests. By default binomial testing is off (alpha=1) and the methods returns that environment as the preferred one, which has the highest likelihood (odds ratio).
reldat	(data.frame) Database with the same structure as x. x is typically a subset of reldat. If given, the occurrence distribution of reldat is used as the null model of sampling. Defaults to NULL, which means that x itself will be used as reldat.
na.rm	(logical) Should the NA entries in the relevant columns of x be omitted automatically?
bycoll	(logical) If set to TRUE, the number of collections (or samples, in coll) will be used rather than the number of occurrences.

Details

Sampling patterns have an overprinting effect on the frequency of taxon occurrences in different environments. The environmental affinity (Foote, 2006; Kiessling and Aberhan, 2007; Kiessling and Kocsis, 2015) expresses whether the taxa are more likely to occur in an environment, given the sampling patterns of the dataset at hand. The function returns the likely preferred environment for each taxon as a vector. NA outputs indicate that the environmental affinity is equivocal based on the selected method.

The following methods are implemented:

`'majority'`: Environmental affinity will be assigned based on the number of occurrences of the taxon in the different environments, without taking sampling of the entire dataset into account. If the taxon has more occurrences in *environment 1*, the function will return *environment 1* as the preferred habitat.

`'binom'`: The proportion of occurrences of a taxon in *environment 1* and *environment 2* will be compared to a null model, which is based on the distribution of all occurrences from the stratigraphic range of the taxon (in `x` or if provided, in `reldat`). Then a binomial test is run on with the numbers of the most likely preference (against all else). The `alpha` value indicates the significance of the binomial tests, setting `alpha` to `1` will effectively switch the testing off: if the ratio of occurrences for the taxon is different from the ratio observed in the dataset, an affinity will be assigned. This is the default method. If an environment is not sampled at all in the dataset to which the taxon's occurrences are compared to, the binomial method returns `NA` for the taxon's affinity.

References

Foote, M. (2006). Substrate affinity and diversity dynamics of Paleozoic marine animals. Paleobiology, 32(3), 345-366.

Kiessling, W., & Aberhan, M. (2007). Environmental determinants of marine benthic biodiversity dynamics through Triassic–Jurassic time. Paleobiology, 33(3), 414-434.

Kiessling, W., & Kocsis, Á. T. (2015). Biodiversity dynamics and environmental occupancy of fossil azooxanthellate and zooxanthellate scleractinian corals. Paleobiology, 41(3), 402-414.

Value

A named vector, values corresponding to affinities.

Examples

```
data(corals)
# omit values where no occurrence environment entry is present, or where unknown
  fossils<-subset(corals, stg!=95)
  fossilEnv<-subset(fossils, bath!="uk")
# calculate affinities
  aff<-affinity(fossilEnv, env="bath", tax="genus", bin="stg", alpha=1)
```

---

binstat *Sampling statistics and diversity indices in every bin*

Description

This function will return the basic sampling summaries of a dataset

Usage

```
binstat(
  x,
  tax = "genus",
  bin = "stg",
  coll = NULL,
  ref = NULL,
  noNAStart = FALSE,
  duplicates = NULL,
  xexp = NULL,
  indices = FALSE
)
```

Arguments

x	(data.frame): The occurrence dataset.
tax	(character): The column name of taxon names.
bin	(character): The column name of bin names.
coll	(character): The column name of collection numbers. (optional)
ref	(character): The column name of reference numbers. (optional)
noNAStart	(logical) Useful when the dataset does not start from bin no. 1, but positive integer bin numbers are provided. Then noNAStart=TRUE will cut the first part of the resulting table, so the first row will contain the estimates for the lowest bin number. In case of positive integer bin identifiers, and if noNAStart=FALSE, the index of the row will be the bin number.
duplicates	(logical): The function will check whether there are duplicate occurrences (multiple species/genera). When set to NULL, nothing will happen, but the function will notify you if duplicates are present. If set to TRUE, the function will not do anything with these, if set to FALSE, the duplicates will be omitted.
xexp	(numeric): Argument of the OxW subsampling type (subtrialOXW).Setting this parameter to a valid numeric value will return the maximum quota for xexp.
indices	(logical): Setting this value to TRUE will calculate all indices implemented in (indices).

Details

Secondary function of the package that calculates a number of sampling related variables and diversity estimators for each bin. In contrast to the (divDyn) function, the bins are treated independently in this function. The function also returns the maximum subsampling quota for OxW subsampling (subtrialOXW) with a given xexp value.

By setting total to FALSE (default), the following results are output:

occs: The number of occurrences in each time bin.

colls: The number of collections in each time bin.

xQuota: The maximum quota for OxW subsampling (subtrialOXW) with the given xexp value. The number of occurrences in each collection is tabulated, and is raised to the power of xexp. The xQuota value is the sum of these values across all collections in a time slice.

refs: The number of references in each time bin.

SIBs: The number of Sampled-In-Bin taxa in each time bin.

occ1: The number of taxa in each time bin, that occur in only 1 collection.

ref1: The number of taxa in each time bin, that occur in only 1 reference.

occ2: The number of taxa in each time bin, that occur in exactly 2 collections.

ref2: The number of taxa in each time bin, that occur in exactly 2 references.

u: Good's u, coverage estimator based on the number of single-collection taxa (occ1).

uPrime: Good's u, coverage estimator based on the number of single-reference taxa (ref1).

chao1occ: Chao1 extrapolation estimator, based on the the number of single-collection and two-collection taxa (occ1).

chao1ref: Chao1 extrapolation estimator, based on the the number of single-reference and two-reference taxa (occ2).

Value

A data.frame with rows corresponding to bin entries.

Examples

```
data(corals)
# slice-specific sampling
basic <- binstat(corals, tax="genus", bin="stg")

# subsampling diagnostic
 subStats <- subsample(corals, method="cr", tax="genus", FUN=binstat,
   bin="stg", q=100,noNAStart=FALSE)

# maximum quota with xexp
more <- binstat(corals, tax="genus", bin="stg", coll="collection_no", xexp=1.4)
```

---

categorize *Mapping multiple entries to categories*

Description

This basic function replaces groups of values in a vector with single values with the help of a key object.

Usage

```
categorize(x, key, incbound = "lower")
```

Arguments

x	(vector) Object containing the values to be replaced.
key	(list) A list of vectors. Each vector includes the possible elements that will be replaced in a group, the names of the vectors will be the replacement values. Also has to include an element named 'default' with a single value. (see examples)
incbound	(character) Either "lower" or "higher". Interval identifiers will be treated with different interval rules. "lower" will treat the lowest entry as included, "higher" works the opposite. The argument will be renamed to 'include.lowest' to make the interface easier to remember.

Details

Online datasets usually contain overly detailed information, as enterers intend to conserve as much data in the entry process, as possible. However, in analyses some values are treated to represent the same, less-detailed information, which is then used in further procedures. The map function allows users to do this type of multiple replacement using a specific object called a 'key'.

A key is an informal class and is essentially a list of vectors. In the case of character vectors as x, each vector element in the list corresponds to a set of entries in x. These will be replaced by the name of the vector in the list, to indicate their assumed identity.

In the case of numeric x vectors, if the list elements of the key are numeric vectors with 2 values, then this vector will be treated as an interval. The same value will be assigned to the entries that are in this interval (Example 2). If x contains values that form the boundary of an interval, than either only the one of the two boundary values can be considered to be in the interval (see the incbound argument to set which of the two). The elements of key are looped through in sequence. If values of x occur in multiple elements of key, than the last one will be used (Example 3).

Examples of this data type have been included (keys) to help process Paleobiology Database occurrences.

Value

A vector with replacements.

Examples

```
# Example 1
# x, as character
   set.seed(1000)
   toReplace <- sample(letters[1:6], 15, replace=TRUE)
# a and b should mean 'first', c and d 'second' others: NA
   key<-list(first=c("a", "b"), second=c("c", "d"), default=NA)
# do the replacement
  categorize(toReplace, key)

# Example 2 - numeric entries and mixed types
# basic vector to be grouped
  toReplace2<-1:16
```

```
# replacement rules: 5,6,7,8,9 should be "more", 11 should be "eleven" the rest: "other"
  key2<-list(default="other", more=c(5,10),eleven=11)
  categorize(toReplace2, key2)

# Example 3 - multiple occurrences of same values
# a and b should mean first, a and should mean 'second' others: NA
  key3<-list(first=c("a", "b"), second=c("a", "d"), default=NA)
# do the replacement (all "a" entries will be replaced with "second")
  categorize(toReplace, key3)
```

`cleansp` *Cleanse Species Name Vector*

Description

This function will take a vector of binomial names with various qualifiers of open nomenclatures, and removes them form the vector entries. Only the the genus and species names will remain.

Usage

```
cleansp(
  x,
  debug = FALSE,
  collapse = "_",
  subgenera = TRUE,
  misspells = TRUE,
  stems = TRUE
)
```

Arguments

`x`	(`character`): the vector containing species names with qualifiers of open taxonomy.
`debug`	(`logical`): `FALSE` will return the cleaned species name vector, `TRUE` returns a data table that allows one by one checking.
`collapse`	(`character`): This argument will be passed to the paste function's argument of the same name. The character value to be inserted between the genus and species names.
`subgenera`	(`logical`): `FALSE` omits subgenus information (in parentheses) and will construct a unique binomen based on the genus and species names alone. `TRUE` (default) will promote the subgenus names and it will create a new binomen based on the subgenus rather than the genus name.
`misspells`	`logical`: Resolution of common spelling mistakes, such as diphtongs and alternate spellings: 'ue' is replaced with 'u', 'ae' is replaced with 'e', 'll' with 'l', 'ss' with 's'and 'y' with 'i'.
`stems`	(`logical`): Setting this to `TRUE` will omit the adjective declination suffices from the species names.

Details

This version will keep subgenera, and will not assign species to the base genus. The following qualifiers will be omitted: *"n."*, *"sp."*, *"?"*, *"gen."*, *"aff."*, *"cf."*, *"ex gr."*, *"subgen."*, *"spp"* and informal species designated with letters. Entries with *"informal"* and *"indet."* in them will also be invalidated.

Value

A data.frame or character vector.

Author(s)

Adam T. Kocsis, Gwenn Antell. Adam T. Kocsis wrote the main body of the function, subroutines called by the `misspells` and `stems` are the modified work of Gwen Antell.

Examples

```
examp <- c("Genus cf. species", "Genus spp.", "Family indet.",
  "Mygenus yourspecies", "Okgenus ? questionsp",
  "Genus (cf. Subgenus) aff. species")
cleansp(examp)
```

---

corals *Fossil occurrences of scleractinian (stony) corals from the Paleobiology Database*

Description

Example dataset to illustrate the package's basic functionalities.

Usage

```
data(corals)
```

Format

A `data.frame` with 29775 observations and 38 variables:

`genus` Genus names of the occurrences. Cross referenced with a compiled table, the simplified version of this can be found in the supplementary material of Kiessling and Kocsis (2015).

`collection_no` The number of the collection of the occurrence in the PaleoDB.

`family` Family name of the occurrence.

`abund_value` Abundance value.

`abund_unit` Unit of abundance values.

`reference_no` The reference number of the occurrence in the PaleoDB.

`life_habit` The lifestyle of the occurring taxon.

`diet` The diet of the occurring taxon.

`country` Country of occurrence.

`geoplate` Plate id of the occurrence.

`lat` Present day latitude of the occurrence.

`lng` Present day longitude of the occurrence.

`paleolat` Reconstructed paleolatitude of the occurrence.

`paleolng` Reconstructed paleolongitude of the occurrence.

`period` Period of origin.

`epoch` Epoch of origin.

`subepoch` Subepoch of origin.

`stage` Geologic stage of the embedding rocks.

`early_interval` Early interval name registered in the PaleoDB dynamic time scale.

`late_interval` Late interval name registered in the PaleoDB dynamic time scale.

`max_ma` Maximum estimated age based on the PaleoDB dynamic time scale.

`min_ma` Minimum estimated age based on the PaleoDB dynamic time scale.

`stg` Bin number in the stage-level timescale `stages`.

`ten` Bin number in the PaleoDB 10 million year resolution timescale `tens`.

`env` Environment of the occurrence: reefal (`r`), non-reefal (`nr`) or unknown (`uk`), based on `keys`.

`lith` Substrate of the occurrence: carbonate (`c`), siliciclastic (`s`) or unknown (`uk`), based on `keys`.

`latgroup` Latitude of the occurrence: tropical (`t`) or non-tropical (`nt`).

`bath` Inferred depth of the occurrence: deep (`deep`), shallow (`shal`) or unknown (`uk`), based on `keys`.

`gensp` The binomen of the occurrence.

`ecology` Symbiotic status of the occurring coral: zooxanthellate (`z`) or azooxanthellate (`az`, including apozooxanthellates).

`ecologyMostZ` Symbiotic status of the occurring coral, incorporating the uncertainty of inferred symbiotic status. This variable includes assignment with the maximum likely number of zooxanthellate genera.

`ecologyMostAZ` Symbiotic status of the occurring coral, incorporating the uncertainty of inferred symbiotic status. This variable includes assignment with the maximum likely number of azooxanthellate genera.

`growth` Growth type of the coral: `colonial` or `solitary`.

`integration` Integration of corallites from the scale of 0 to 4. `solitary` corals are marked with 0s.

Details

This particular dataset was used in a study by Kiessling and Kocsis (2015). All occurrences of Scleractinia were downloaded from the Paleobiology Database (PaleoDB, `https://paleobiodb.org/`) on 23 September 2014, originally comprising 32420 occurrences. They were than cross-checked with data from Corallosphere (`http://corallosphere.org`). See the article text for details.

References

Kiessling, W., & Aberhan, M. (2007). Environmental determinants of marine benthic biodiversity dynamics through Triassic–Jurassic time. Paleobiology, 33(3), 414-434.

Source

https://paleobiodb.org/

`divDyn` *Time series from metrics of diversity dynamics*

Description

This function calculates various metrics from occurrence datasets in the form of time series.

Usage

```
divDyn(
  x,
  tax,
  bin = NULL,
  age = NULL,
  revtime = FALSE,
  breaks = NULL,
  coll = NULL,
  ref = NULL,
  om = NULL,
  noNAStart = FALSE,
  data.frame = TRUE,
  filterNA = FALSE
)
```

Arguments

`x`	(`data.frame`) Fossil occurrence table.
`tax`	(`character`) Variable name of the occurring taxa (variable type: `factor` or `character` - such as "`genus`"
`bin`	(`character`) Variable name of the discrete bin numbers of the occurrences. This variable should be `numeric`. Time flows from lower to higher values by default. Use `revtime` to reverse this order.
`age`	(`character`) Variable name of the ages of the occurrences that will be sliced with the `slice` function using the intervals provided in `breaks`. This variable should be `numeric`. Time flows from higher to lower values by default. Use `revtime` to reverse this order.
`revtime`	(`logical`) Argument for reversing the default direction of time. Setting this argument to `TRUE` will make time flow from higher to lower values when `bin` is used, and from lower to higher values, when `age` is given. CAUTION: Failing to set this argument properly can make originations become extinctions and vice versa!

breaks	(numeric) If NULL (default) the used values in the bin variable will designate independent time slices that follow each other in succession. If a vector is provided, than the numeric entries in bin will be binned similarly to the hist or cut function. The order of elements in this vector is arbitrary.
coll	(character) The variable name of the collection identifiers. (optional, only for use with the internal omit function)
ref	(character) The variable name of the reference identifiers. (optional, only for use with the internal omit function)
om	(character) The om argument of the omit() function. If set to NULL (default), then no occurrences will be omitted before the execution of the function.
noNAStart	(logical) Useful when the entries in the bin variable do not start from bin no. 1, but positive integer bin numbers are provided. Then noNAStart=TRUE will cut the first part of the resulting table, so the first row will contain the estimates for the lowest bin number. In case of positive integer bin identifiers, and if noNAStart=FALSE, the index of the row will be the bin number.
data.frame	(logical) Should the output be a data.frame or a matrix?
filterNA	(logical) The filterNA parameter of the omit function.

Details

The following variables are produced:

bin: Bin number, or the numeric identifier of the bin.

tThrough: Number of through-ranging taxa, taxa that have first occurrences before, and last occurrences after the focal bin.

tOri: Number of originating taxa, taxa that have first occurrences in the focal bin, and last occurrences after it.

tExt: Number of taxa getting extinct. These are taxa that have first occurrences before the focal bin, and last occurrences in it.

tSing: Number of stratigraphic singleton (single-interval) taxa, taxa that only occur in the focal bin.

t2d: Number of lower two timers (Alroy, 2008; 2014), taxa that are present in the *i*-1th and the ith bin (focal bin).

t2u: Number of upper two timers (Alroy, 2008; 2014), taxa that are present in the *i*th (focal) and the *i*+1th bin. (Alroy, 2008; 2014)

tGFu: Number of upper gap-fillers (Alroy, 2014), taxa that occurr in bin *i*+2 and *i*-1, but were not found in *i*+1. (Alroy, 2014)

tGFd: Number of lower gap-fillers (Alroy, 2014), taxa that occurr in bin *i*-2 and *i*+1, but were not found in *i*-1. (Alroy, 2014)

t3: Number of three timer taxa (Alroy, 2008; 2014), present in bin *i*-1, *i*, and *i*+1. (Alroy, 2008; 2014)

tPart: Part timer taxa (Alroy, 2008; 2014), present in bin *i*-1,and *i*+1, but not in bin *i*.

extProp: Proportional extinctions including single-interval taxa: *(tExt + tSing) / (tThrough + tOri + tExt + tSing)*.

`oriProp`: Proportional originations including single-interval taxa: *(tOri + tSing) / (tThrough + tOri + tExt + tSing)*.

`extPC`: Per capita extinction rates of Foote (1999). *-log(tExt/(tExt + tThrough))*. Values are not normalized with bin lengths. Similar equations were used by Alroy (1996) but without taking the logarithm.

`oriPC`: Per capita origination rates of Foote (1999). *-log(tOri/(tOri + tThrough))*. Values are not normalized with bin lengths. Similar equations were used by Alroy (1996) but without taking the logarithm.

`ext3t`: Three-timer extinction rates of Alroy (2008). *log(t2d/t3)*.

`ori3t`: Three-timer origination rates of Alroy (2008). *log(t2u/t3)*.

`extC3t`: Corrected three-timer extinction rates of Alroy (2008). *ext3t[*i*] + log(samp3t[*i*+1])*.

`oriC3t`: Corrected three-timer origination rates of Alroy (2008). *ori3t[*i*] + log(samp3t[*i*-1])*.

`divSIB`: Sampled-in-bin diversity (richness), the number of genera sampled in the focal bin.

`divCSIB`: Corrected sampled-in-bin diversity (richness). *divSIB/samp3t*totSamp3t*, where *totSamp3t* is total three-timer sampling completeness of the dataset (Alroy, 2008).

`divBC`: Boundary-crosser diversity (richness), the number of taxa with ranges crossing the boundaries of the interval. *tExt + tOri + tThrough*.

`divRT`: Range-through diversity (richness), all taxa in the interval, based on the range-through assumption. *(tSing + tOri + tExt + tThrough)*.

`sampRange`: Range-based sampling probability, without observed range end-points (Foote), *(divSIB - tExt - tOri- t-Sing)/tThrough*

`samp3t`: Three-timer sampling completeness of Alroy (2008). *t3/(t3+tPart)*

`extGF`: Gap-filler extinction rates of Alroy(2014). *log((t2d + tPart)/(t3+tPart+tGFd))*

`oriGF`: Gap-filler origination rates of Alroy(2014). *log((t2u + tPart)/(t3+tPart+tGFd))*

`E2f3`: Second-for-third extinction propotions of Alroy (2015). As these metrics are based on an algorithmic approach, for the equations please refer to the Alroy (2015, p. 634, right column and Eq. 4)). See source code (https://github.com/divDyn/r_package) for the exact implementation, found in the `Metrics` function in the diversityDynamics.R file.

`O2f3`: Second-for-third origination propotions of Alroy (2015). Please see `E2f3`.

`ext2f3`: Second-for-third extinction rates (based on Alroy, 2015). Transformed to the usual rate form with *log(1/(1-E2f3))*.

`ori2f3`: Second-for-third origination rates (based on Alroy, 2015). Transformed to the usual rate form with *log(1/(1-O2f3))*.

References:

Foote, M. (1999) Morphological Diversity In The Evolutionary Radiation Of Paleozoic and Post-Paleozoic Crinoids. Paleobiology 25, 1–115. doi:10.1017/S0094837300020236.

Alroy, J. (2008) Dynamics of origination and extinction in the marine fossil record. Proceedings of the National Academy of Science 105, 11536-11542. doi: 10.1073/pnas.0802597105

Alroy, J. (2014) Accurate and precise estimates of origination and extinction rates. Paleobiology 40, 374-397. doi: 10.1666/13036

Alroy, J. (2015) A more precise speciation and extinction rate estimator. Paleobiology 41, 633-639. doi: 10.1017/pab.2015.26

Value

A data.frame object, with every row corresponding to a time bin.

Examples

```
# import data
  data(corals)
  data(stages)

# calculate metrics of diversity dynamics
   dd <- divDyn(corals, tax="genus", bin="stg")

# plotting
  tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
    ylab="range-through diversity (genera)", ylim=c(0,230))
  lines(stages$mid, dd$divRT, lwd=2)

 # with omission of single reference taxa
   ddNoSing <- divDyn(corals, tax="genus", bin="stg", om="ref", ref="reference_no")
   lines(stages$mid, ddNoSing$divRT, lwd=2, col="red")

 # using the estimated ages (less robust) - 10 million years
   # mean ages
   corals$me_ma <- apply(corals[, c("max_ma", "min_ma")], 1, mean)
   # ages reverse the direction of time! set ages to TRUE in this case
   ddRadio10 <- divDyn(corals, tax="genus", age="me_ma",
breaks=seq(250,0,-10))
   lines(ddRadio10$me_ma, ddRadio10$divRT, lwd=2, col="green")

 # legend
   legend("topleft", legend=c("all", "no single-ref. taxa", "all, estimated ages"),
     col=c("black", "red", "green"), lwd=c(2,2,2), bg="white")
```

`fadlad` *FAD - LAD matrix from occurrence data*

Description

Function to generate range data from an occurrence dataset.

Usage

```
fadlad(
  x,
  tax,
  bin = NULL,
  age = NULL,
```

```
  revtime = FALSE,
  na.rm = TRUE,
  diffbin = TRUE
)
```

Arguments

x	(data.frame): Occurrence data.
tax	(character): Column name of taxon names.
bin	(character): Column name(s) of the discreet bin variable(s). If two column names are entered, then they will interpreted as minimum and maximum uncertainty (see examples) By default, time flows from lower to higher numbers. You can change this behavior by setting revtime=FALSE. Either a bin or age argument is mandatory.
age	(character): Column name(s) of the continuous age variable(s). If two column names are entered, then they will interpreted as the minimum and maximum age uncertainty. (see examples) By default, time flows from higher to lower numbers. You can change this behavior by setting revtime=FALSE. Either a bin or age argument is mandatory.
revtime	(logical): Should time be reversed?
na.rm	(logical): Should taxa that have no valid FADs or LADs (due to NA entries) be removed from the output?
diffbin	(logical): Difference-based duration for discreet time (only applicabble to cases when bin is provided). If set to TRUE, single-interval taxa will 0 durations. Setting this argument to FALSE will add code1 to the durations of all taxa.

Details

The function will output First and Last Appearance Dates of the taxa in the dataset. Keep in mind that incomplete sampling will influence these data and will make the ranges appear shrunken.

The following variables are produced:

row.names attribute: The names of the taxa.

FAD: First appearance dates in time bin nmbers or ages.

LAD: Last appearance dates in time bin numbers or ages.

duration: The durations of taxa in bin numbers or ages.

Value

A data.frame, with rows corresponding to tax entries.

Examples

```
data(corals)

# binned data
  flBinned <- fadlad(corals, tax="genus", bin="stg")
```

```
# using basic bin lengths
  flDual <- fadlad(corals, tax="genus", age=c("max_ma", "min_ma"))

# single age esimate
  data(stages)
  corals$mid <- stages$mid[corals$stg]
  flSingle <- fadlad(corals, tax="genus", age="mid")
```

`fill` *Filling of missing values in a vector, based on the marginal values of the gaps*

Description

The function will loop through a vector and will substitute `NA` values with the value it last encountered or replaced.

Usage

```
fill(x, forward = TRUE, inc = 0)
```

Arguments

`x`	(`vector`) Vector to be filled.
`forward`	(`logical`) Should the loop go forward or backward?
`inc`	(`numeric`) Only if `x` is `numeric`, the function will increase the substituted value by this amount (useful for filling in sequences).

Details

`NA`s won't be substituted when they are the first values the loop encounters.

Value

A logical vector.

Examples

```
# forward, replace with previous
dummy<- c(TRUE, FALSE, NA, TRUE, FALSE, NA)
fill(dummy)

# forward, replace with previous+1
dummy2 <- c(1,NA, 3, 1, 2, NA, NA, 9, NA,3)
fill(dummy2, inc=1)
```

```
# backward, replace with previous in loop direction
fill(dummy2, inc=0, forward=FALSE)
```

`georange` *Estimation of geographic ranges from occurrence data*

Description

Geographic range as a function of a set of coordinates or sample/site/cell membeships.

Usage

```
georange(x, lng = NULL, lat = NULL, loc = NULL, method = "co")
```

Arguments

`x`	(`data.frame`) Occurrence table containing the coordinates/locality memberships as variables.
`lng`	(`character`) The variable name of the longitudes, required for the `"co"`, `"mst"` and `"mgcd"` methods.
`lat`	(`character`) The variable name of the latitudes, required for the `"co"`, `"mst"` and `"mgcd"` methods.
`loc`	(`character`) The variable name of the locality entries: cells, site or samples, required for the `"lo"` method.
`method`	(`character`) Geographic range estimator method. Can take multiple entries (concatenating the results in a vector). The following methods are implemented: `"co"`: coordinate-based occupancy, the number of different coordinate pairs; `"lo"`: locality-based occupancy for sites, samples or geographic cells, number of different entries in a variable. `"mst"`, the total length of a minimum spanning tree of the point cloud, based on the great circle distances between points (requires the 'vegan' and 'icosa' packages). `"mgcd"`, maximum great-circle distance that can be measured in the point cloud (this version is limited to half the circumference of the equator, requires the 'icosa' package).

Details

Multiple estimators of geographic ranges are implemented based on coordinates or cell identifiers. The function outputs a vector of the results based on the calculation methods specified in `methods`.

Value

A numeric vector with geographic ranges (multiple methods).

Examples

```
data(corals)
# select a  taxon from a certain time slice
  bitax <- corals[corals$stg==69 & corals$genus=="Microsolena",]
  georange(bitax, lng="paleolng", lat="paleolat", method="co")
```

`indices` *Scalar indices of diversity*

Description

This function includes some indices that characterize a species-abundance/occurrence distribution.

Usage

```
indices(x, samp = NULL, method = NULL)
```

Arguments

`x`	either a `character` vector of occurrences or a table of counts (`matrix`).
`samp`	(`vector`): Only applicable if `x` is vector of occurrence entries. The id of the sampling units. A necessary variable for SCOR.
`method`	(`character`): The type of metric that is to be calculated. The default value (`NULL`) will calculate all implemented metrics.

Details

This set is not complete and does not intend to supercede additional R packages (e.g. vegan). However, some metrics are presented here as they are not implemented elsewhere or because they are invoked more frequently. The following entries can be added to the `method` argument of the function, which are also named accordingly in the output table/vector.

`"richness"`: The number of sampled species.

`"shannon"`: The Shannon entropy.

`dom`: The Berger-Parker dominance index, the proportion of occurrences in the time bin that belong to the most frequent taxon.

`"hill2"`: The second order Hill number (Jost, 2006; q=2), which will be calculated by default. You can specify additional Hill numbers with adding `"hillXX"` to the `method` argument, such as `"hill3"` for (q=3). The first Hill number is defined as the exponentiad version of Shannon entropy (Eq. 3 in Jost, 2006).

`"squares"`: The 'squares' richness estimator of J. Alroy (2018).

`"chao2"`: The Chao2 estimator for incidence-based data.

`"SCOR"`: The Sum Common Species Occurrence rate of Hannisdal et al. (2012). This method will only be calculated if the occurrence entries (vector) a collection vector is provided (see examples).

Value

A named numeric vector.

References

Alroy, J. 2018. Limits to species richness in terrestrial communities. Ecology Letters.

Hannisdal, B., Henderiks, J., & Liow, L. H. (2012). Long-term evolutionary and ecological responses of calcifying phytoplankton to changes in atmospheric CO2. Global Change Biology, 18(12), 3504–3516. https://doi.org/10.1111/gcb.12007

Jost, L. (2006). Entropy and diversity. Oikos, 113, 363–375. https://doi.org/10.1111/j.2006.0030-1299.14714.x

Examples

```
# the coral data
  data(corals)

# Pleistocene subset
  plei <- corals[corals$stg==94,]

# calculate everything
  pleiIndex<-indices(plei$genus, plei$coll)
```

`keys` *Keys to process stratigraphic, environmental and lithological information from the Paleobiology Database*

Description

Lists of entries treated as indicators of similar characteristics

Usage

```
data(keys)
```

Format

A `list` of 7 `lists`:

`tenInt` A `list` of `vectors`. Entries in the `early_interval` and `late_interval` variables of PaleoDB downloads indicate the collections' positions in the dynamic time scale. These entries were linked to 10 million year-resolution time scale stored in `tens`. These links were compiled using a download from the FossilWorks website (`http://fossilworks.org/`), on 08 June, 2018. You can check the lookup table `stratkeys` here. This is version 0.9.2

`stgInt` A `list` of `vectors`. Entries in the `early_interval` and `late_interval` variables of PaleoDB downloads indicate the collections' positions in the dynamic time scale. These entries were linked to stage-resolution time scale stored in `stages`. See `binInt` for version information.These entries are reliable only in the Post-Ordovician!

`reefs` A `list` of `vectors`. Entries in the `environment` field of the PaleoDB download indicate information regarding the likely reefal origin of carbonatic rocks. See the vignette ('§PhaneroCurve') on the exact use of these data. v0.9.

`lith` A `list` of `vectors`. Entries in the `lithology1` field of the PaleoDB download indicate information regarding the substrate of the embedding rocks. This key maps the entries to `siliciclastic`, `"carbonate"` or `"unknown"` substrates. v0.9.

`lat` A `list` of `vectors`. Entries in the `paleolat` field of the PaleoDB download indicate information regarding paleolatitude of the occurrences. This key maps the entries to `"tropical"` or `"non-tropical"` latitudes. v0.9.

`grain` A `list` of `vectors`. Entries in the `lithology1` field of the PaleoDB download indicate information regarding the grain sizes of the depositional environment. This key maps the entries to `"coarse"`, `"fine"` or `"unknown"` grain sizes. v0.9.

`depenv` A `list` of `vectors`. Entries in the `environment` field of the PaleoDB download indicate information regarding the onshore-offshore nature of the depositional environment. This key maps the entries to `"onshore"`, `"offshore"` or `"unknown"` environment. v0.9.3

Details

Entries in the stratigraphic, lithological and environment fields of current Paleobiology Database downloads are too numerous to form the basis of analyses without transformations. This variable includes potential groupings of entries that represent similar characteristics. These objects can be used by the `categorize` function to create new variables of stratigraphic, environmental and lithological information.

Source

Stratigraphic assignments are based on the download of collection data from Fossilworks (`http://fossilworks.org/`) and the dynamic time scale of the Paleobiology Database, written by J. Alroy. The assignment of numeric values were done by A. Kocsis. Environmental variables were grouped by W. Kiessling.

---

modeltab *Origination/extinction response table for statistical modelling.*

Description

This function takes an occurrence dataset and reformats it to a table that can be used as input for logistic models.

Usage

```
modeltab(x, tax, bin, taxvars = NULL, rt = FALSE, singletons = FALSE)
```

Arguments

x	(data.frame) Fossil occurrence table.
tax	(character) Variable name of the occurring taxa (variable type: factor or character - such as "genus")
bin	(character) Variable name of the bin numbers of the occurrences. This variable should be numeric and should increase as time passes by (use negative values for age estimates). The current version only supports discreet, non-negative integer interval numbers.
taxvars	(character) Taxon-specific column names of the variables that should be included in the output table. Only one entry/taxon is used, make sure that the data are clean.
rt	(logical) Should the range-through assumption be applied within the function? If set to TRUE then missing occurrences will be interpolated with FALSE values in both the ext and ori variables. .
singletons	(logical) Should single-interval taxa be removed from the final table? This is recommended, as it is impossible to get a FALSE response for these taxa.

Details

Every entry in the output table corresponds to one cell in the bin/tax matrix. This function omits duplicates and concatenates two logical vectors (response variables) to the occurrence dataset: The ori vector is TRUE in the interval when the taxon first appeared, and FALSE in all others. The ext vector is TRUE in the interval the taxon appeared for the last time, and FALSE in the rest.

Value

A data.frame with binary response variables.

Examples

```
# load necessary data
data(corals)
# simple table
modTab<-modeltab(corals, bin="stg", tax="genus", taxvars=c("ecology", "family"))
```

---

omit *Omission of taxa that have a poor occurrence record*

Description

Function to quickly omit single-collection and single-reference taxa.

Usage

```
omit(
  x,
  om = "ref",
  tax = "genus",
  bin = "bin",
  coll = NULL,
  ref = NULL,
  filterNA = FALSE
)
```

Arguments

x	(data.frame) Occurrence dataset, with bin, tax and coll as column names.
om	(character) The type of omission. "coll" omits occurrences of taxa that occurr only in one collection. "ref" omits occurrences of taxa that were described only in one reference. "binref" will omit the set of single reference taxa that were described by more than one references, but appear in only one reference in a time bin.
tax	(character) The name of the taxon variable.
bin	(character) The name of the bin variable (has to be numeric for the function to run). For time series, this is the time slice variable.
coll	(character) The variable name of the collection identifiers.
ref	(character) The variable name of the reference identifiers (optional).
filterNA	(logical) Additional entries can be added to influence the dataset that might not have reference or collection information (NA entries). These occurrences are treated as single-collection or single-reference taxa if the na.rm argument is set to FALSE (default). Setting this argument to TRUE will keep these entries. (see example)

Details

The function returns a logical vector, with a value for each row. TRUE values indicate rows to be omitted, FALSE values indicate rows to be kept. The function is embedded in the divDyn function, but can be called independently.

Value

A logical vector.

Examples

```
# omit single-reference taxa
  data(corals)
  data(stages)
  toOmit <- omit(corals, bin="stg", tax="genus", om="ref", ref="reference_no")
  x <- corals[!toOmit,]
```

```
# within divDyn
  # plotting
  tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
    ylab="range-through diversity (genera)", ylim=c(0,230))
  # multiple ref/slice required
  ddNoSing <- divDyn(corals, tax="genus", bin="stg", om="binref", ref="reference_no")
  lines(stages$mid, ddNoSing$divRT, lwd=2, col="red")

  # with the recent included (NA reference value)
  ddNoSingRec <- divDyn(corals, tax="genus", bin="stg",
    om="binref", filterNA=TRUE,ref="reference_no")
  lines(stages$mid, ddNoSingRec$divRT, lwd=2, col="blue")

  # legend
  legend("topleft", legend=c("no single-ref. taxa",
    "no single-ref. taxa,\n with recent"),
    col=c("red", "blue"), lwd=c(2,2))
```

`parts` *Plot time series counts or proportions as polygons*

Description

This function plots the changing shares of categories in association with an independent variable.

Usage

```
parts(
  x,
  b = NULL,
  ord = "up",
  prop = FALSE,
  plot = TRUE,
  col = NULL,
  xlim = NULL,
  border = NULL,
  ylim = c(0, 1),
  na.valid = FALSE,
  labs = TRUE,
  labs.args = NULL,
  vertical = FALSE
)
```

Arguments

`x`	`(numeric)`: The independent variable through which the proportion is tracked. Identical entries are used to assess which values belong together to a set. Their values represent the x coordinate over the plot.

b	(character or factor): A single vector with the category designations. This vector will be segmented using the entries of x.
ord	(character): The parameter of the variable order. Either "up" (increasing alphabetical order), "down" (decreasing alphabetical order) or the vector of categories in the desired order.
prop	(logical): Should the diagram show proportions (TRUE) or counts (FALSE)?
plot	(logical): If set to TRUE, then the function will plot the output. If set to FALSE, then a matrix with the relevant values will be returned. This output is similar to the output of table, but handles proportions instantly.
col	(character): The color of polygons, has to a be a vector with as many entries as there are categories in b. By default (col=NULL) this is grayscale.
xlim	(numeric): Two values, analogous to the xlim argument of plot, and has to exceed the range of x. The polygons that represent non-zero values with the lowest and highest values of x will be extended to these x coordinates.
border	(character): The a single color of the polygon borders. By default (border=NA), no borders are drawn.
ylim	(numeric): If prop=TRUE, then the argument controls the position of the proportions in the plotting area (useful to show proportions as a sub plot in a plot). If prop=FALSE, then the entire plotting area will be shifted by a single ylim value.
na.valid	(logical): If TRUE, than the missing values will be treated as an independent category. Entries where x is NA will be omitted either way.
labs	(logical): Should the category names be plotted?
labs.args	(list): Arguments for the text function. If one entry for each argument is provided, then it will be applied to all labels. If the number of elements in an argument equals the number of categories to be plotted, then one to one assignment will be used. For example, for 4 categories in total, if the labs.args list contains a col vector element with 4 values, see examples).
vertical	(logical): Horizontal or vertical plotting? If FALSE, the independent variable will be horizontal, if TRUE, the count/proportion variable will be horizontal. In the latter case xlim and ylim has reversed roles.

Details

This function is useful for displaying the changing proportions of a category as time progresses. Check out the examples for the most frequent implementations.

To be added: missing portions are omitted in this version, but should be represented as gaps in the polygons.

Value

The function has no return value.

Examples

```
# dummy examples
  # independent variable
  slc<-c(rep(1, 5), rep(2,7), rep(3,6))

  # the categories as they change
  v1<-c("a", "a", "b", "c", "c") # 1
  v2<-c("a", "b", "b", "b", "c", "d", "d") # 2
  v3<-c("a", "a", "a", "c", "c", "d") #3
  va<-c(v1, v2,v3)

  # basic function
    plot(NULL, NULL, ylim=c(0,1), xlim=c(0.5, 3.5))
    parts(slc, va, prop=TRUE)

  # vertical plot
    plot(NULL, NULL, xlim=c(0,1), ylim=c(0.5, 3.5))
    parts(slc, va, col=c("red" ,"blue", "green", "orange"), xlim=c(0.5,3.5),
      labs=TRUE, prop=TRUE, vertical=TRUE)

  # intensive argumentation
    plot(NULL, NULL, ylim=c(0,10), xlim=c(0.5, 3.5))
    parts(slc, va, ord=c("b", "c", "d", "a"), col=c("red" ,"blue", "green", "orange"),
  xlim=c(0.5,3.5), labs=TRUE, prop=FALSE,
  labs.args=list(cex=1.3, col=c("black", "orange", "red", "blue")))

  # just the values
    parts(slc, va, prop=TRUE,plot=FALSE)

# real example
  # the proportion of coral occurrences through time in terms of bathymetry
  data(corals)
  data(stages)

  # time scale plot
  tsplot(stages, shading="series", boxes="sys", xlim=c(250,0),
    ylab="proportion of occurrences", ylim=c(0,1))

  # plot of proportions
  cols <- c("#55555588","#88888888", "#BBBBBB88")
  types <- c("uk", "shal", "deep")

  parts(x=stages$mid[corals$stg], b=corals$bath,
   ord=types, col=cols, prop=TRUE,border=NA, labs=FALSE)

  # legend
  legend("left", inset=c(0.1,0), legend=c("unknown", "shallow", "deep"), fill=cols,
    bg="white", cex=1.4)
```

ranges *Plotting ranges and occurrence distributions through time*

Description

Visualization of occurrence data

Usage

```
ranges(
  dat,
  bin = NULL,
  tax = NULL,
  xlim = NULL,
  ylim = c(0, 1),
  total = "",
  filt = "include",
  occs = FALSE,
  labs = FALSE,
  decreasing = TRUE,
  group = NULL,
  gap = 0,
  labels.args = NULL,
  ranges.args = NULL,
  occs.args = NULL,
  total.args = NULL
)
```

Arguments

dat	(data.frame): The occurrence dataset or the FAD-LAD dataset that is to be plotted. The FAD dataset must have numeric variables named "FAD" and "LAD". Taxon ranges will be searched for in the row.names attribute of the table.
bin	(character): The column(s) containing the entries of the time dimension. Use one column name if you have one estimate for the occurrences and use two if you have a minimum and a maximum estimate. Reveresed axis (ages) are supported too.
tax	(character): The column containing the taxon entries.
xlim	(numeric) :This argument is used for the subsetting of the taxa. Only those taxa are shown that have ranges within the interval (but ranges are displayed outside of it, if you do not want to plot anything within an interval, use the clip function)
ylim	(numeric): Ranges will be distributed equally between the assigned ylim values. If set to NULL, than it will be based on the plotting area of the open device.

total	(character): The name of the range group to be plotted. When multiple groups are used (see group argument), this is set by the character values in the column.
filt	(character): When xlim filters the taxa, how should they be filtered. "include" (default) will show all ranges that have parts within the xlim interval. "orig" will show only those taxa that originate within the interval.
occs	(logical): Should the occurrence data be plotted? If you entered two bin column names, than occurrences will be plotted with the ranges of the estimates (segments).
labs	(character): Should the taxon labels be plotted?
decreasing	(logical): This parameter sets whether the series of ranges should start from the top decreasing=TRUE or bottom of the plot decreasing=FALSE.
group	(character): By default, all ranges in the plot are treated as parts of the same group. However, one subsetting variable can be named, by which the ranges will be grouped. This has to be a column name in the dataset (see examples).
gap	(numeric): Evaluated only when the group argument points to a valid column. The amount of space between the group-specific range charts, expressed as the proportion of the entire plotting area.
labels.args	(list): Arguments that will be passed to the text function that draws the labels of taxa. If valid grouping is present (see argument group), then vector entries will be distributed across the groups (see examples.)
ranges.args	(list): Arguments that will be passed to the segments function that draws ranges. If valid grouping is present (see argument group), then vector entries will be distributed across the groups (see examples.)
occs.args	(list): Arguments that will be passed to the points or segments functions that draw the occurence points/lines. If you provided two bin columns, occurrence lines will be drawn instead of points. If valid grouping is present (see argument group), then vector entries will be distributed across the groups (see examples.)
total.args	(list): Arguments that will be passed to the text function that draws the total label. If valid grouping is present (see argument group), then vector entries will be distributed across the groups (see examples.)

Details

This function will draw a visual representation of the occurrence dataset. The interpolated ranges will be drawn, as well as the occurrence points.

Value

The function has no return value.

Examples

```
# import
data(stages)
data(corals)
```

```
# all ranges - using the age uncertainties of the occurrences
tsplot(stages, boxes="sys", xlim=c(250,0))
ranges(corals, bin=c("max_ma", "min_ma"), tax="genus", occs=FALSE)

# or use single estimates: assign age esimates to the occurrences
corals$est<-stages$mid[corals$stg]

# all ranges (including the recent!!)
tsplot(stages, boxes="sys", xlim=c(250,0))
ranges(corals, bin="est", tax="genus", occs=FALSE)

# closing on the Cretaceous, with occurrences
tsplot(stages, boxes="series", xlim=c(145,65), shading="short")
ranges(corals, bin="est", tax="genus", occs=TRUE, ranges.args=list(lwd=0.1))

# z and az separately
tsplot(stages, boxes="series", xlim=c(145,65), shading="short")
ranges(corals, bin="est", tax="genus", occs=FALSE, group="ecology",
  ranges.args=list(lwd=0.1))

# same, show only taxa that originate within the interval
tsplot(stages, boxes="series", xlim=c(105,60), shading="short")
ranges(corals, bin="est", tax="genus", occs=TRUE, group="ecology", filt="orig" ,
  labs=TRUE, labels.args=list(cex=0.5))

# same using the age uncertainties of the occurrence age estimates
tsplot(stages, boxes="series", xlim=c(105,60), shading="short")
ranges(corals, bin=c("max_ma", "min_ma"), tax="genus", occs=TRUE, group="ecology", filt="orig" ,
   labs=TRUE, labels.args=list(cex=0.5))

# fully customized/ annotated
tsplot(stages, boxes="series", xlim=c(105,60), shading="short")
ranges(
  corals, # dataset
  bin="est", # bin column
  tax="genus", # taxon column
  occs=TRUE, # occurrence points will be plotted
  group="growth", # separate ranges based on growth types
  filt="orig" , # show only taxa that originate in the interval
  ranges.args=list(
    lwd=1, # set range width to 1
   col=c("darkgreen", "darkred") # set color of the ranges (by groups)
  ),
  total.args=list(
    cex=2, # set the size of the group identifier lablels
    col=c("darkgreen", "darkred") # set the color of the group identifier labels
  ),
  occs.args=list(
   col=c("darkgreen", "darkred"),
   pch=3
 ),
  labs=TRUE, # taxon labels will be plotted
```

```
  labels.args=list(
    cex=0.4, # the sizes of the taxon labels
col=c("darkgreen", "darkred") # set the color of the taxon labels by group
  )
)
```

---

ratesplit *Test of rate split (selectivity)*

Description

This function will determine whether there are meaningful differences between the taxonomic rates in the individual time bins of two subsets of an occurrence database.

Usage

```
ratesplit(
  x,
  sel,
  tax = "genus",
  bin = "stg",
  rate = "pc",
  method = "AIC",
  AICc = TRUE,
  na.rm = TRUE,
  alpha = NULL,
  output = "simple"
)
```

Arguments

x	(data.frame): The fossil occurrence data.
sel	(character): Variable name to do the splitting of the dataset. Can have only two levels.
tax	(character): Variable name of the occurring taxa (variable type: factor or character).
bin	(character): Variable name of the bin numbers of the particular occurrences (numeric). Bin numbers should be in ascending order,can contain NAs, it can start from a number other than 1 and must not start with 0.
rate	(character): The rate metric. Currently only the per capita rates of Foote (1999) are available (rate="pc").
method	(character): Either "AIC", "binom" or "combine". The "AIC" method calculates the Akaike weights of the single and dual rate models. The "binom" method assumes a binomial error distribution of the counts that are necessary for the rate calculations. The "combine" method shows slices that pass both tests, the "AIC" being usually the stronger.

AICc (logical): Only applicable for the "AIC" method. Toggles whether the small sample corrected AIC (AICc) should be used instead of the regular one.

na.rm (logical): Argument indicating whether the function should proceede when NAs are found in the sel column. Setting this argument to TRUE will proceede with the omission of these entries, while FALSE will coerce the function to output a single NA value.

alpha (numeric): Threshold to discriminate between meaningful and meaningless split. If method="AIC", the value corresponds to the minimum weight value the dual model should have. By default it is 0.89, which corresponds to the likelihood ratio of 8. If method="binom", the value corresponds to the alpha value of the binomial test (default: 0.05). If method="combine" than two alpha values are required (1st for the AIC, 2nd for the binomial test). If alpha is NULL, than the default values will be used.

output (character): Either "simple" or "full". "simple" returns the indices of the series where selectivity can be suggested. "full" returns a matrix of Akaike weights, or binomial probabilities.

Details

Splitting an occurrence database to its subsets secreases the amount of information passed to the rate calculations and therefore the precision of the individual estimates. Therefore, our ability to tell apart two similar values decreases with the number of sampled taxa. In order to assess the subsets individually and compare them, it is advised to test whether the split into two subsets is meaningful, given the total data. Examples of this use can be found in Kiessling and Simpson (2011) and Kiessling and Kocsis (2015). The meaningfulness of the split is dependent on the estimate accurracy and the magnitude of the difference. Two different methods are implemented: binom and combine.

References

Foote, M. (1999) Morphological Diversity In The Evolutionary Radiation Of Paleozoic and Post-Paleozoic Crinoids. Paleobiology 25, 1–115. doi:10.1017/S0094837300020236.

Kiessling, W., & Simpson, C. (2011). On the potential for ocean acidification to be a general cause of ancient reef crises. Global Change Biology, 17(1), 56-67.

Kiessling, W., & Kocsis, A. T. (2015). Biodiversity dynamics and environmental occupancy of fossil azooxanthellate and zooxanthellate scleractinian corals. Paleobiology, 41(3), 402-414.

Value

A list of two numeric vectors.

Examples

```
# example with the coral dataset of Kiessling and Kocsis (2015)
data(corals)
data(stages)

# split by ecology
  z<-corals[corals$ecology=="z",]
  az<-corals[corals$ecology=="az",]
```

```
# calculate diversity dynamics
ddZ<-divDyn(z, tax="genus", bin="stg")
ddAZ<-divDyn(az, tax="genus", bin="stg")

# origination rate plot
tsplot(stages, boxes="sys", shading="series", xlim=54:95,
  ylab="raw per capita originations")
lines(stages$mid, ddZ$oriPC, lwd=2, lty=1, col="blue")
lines(stages$mid, ddAZ$oriPC, lwd=2, lty=2, col="red")
legend("topright", inset=c(0.1,0.1), legend=c("z", "az"),
  lwd=2, lty=c(1,2), col=c("blue", "red"), bg="white")

# The ratesplit function
rs<-ratesplit(rbind(z, az), sel="ecology", tax="genus", bin="stg")
rs

# display selectivity with points
# select the higher rates
selIntervals<-cbind(ddZ$oriPC[rs$ori], ddAZ$oriPC[rs$ori])
groupSelector<-apply(selIntervals, 1, function(w) w[1]<w[2])
# draw the points
points(stages$mid[rs$ori[groupSelector]], ddAZ$oriPC[rs$ori[groupSelector]],
  pch=16, col="red", cex=2)
points(stages$mid[rs$ori[!groupSelector]], ddZ$oriPC[rs$ori[!groupSelector]],
  pch=16, col="blue", cex=2)
```

`repmatch` *Replicate matching and merging*

Description

This pseudo-generic function iterates a function on the subelements of a list of objects that have the same class and matching dimensions/names and reorganizes the result to match the structure of the replicates or a prototype template.

Usage

```
repmatch(x, FUN = NULL, proto = NULL, direct = c("dim", "name"), ...)
```

Arguments

`x`	(`list`): A `list` of replicates.
`FUN`	(`function`): A function to merge with and to be applied to the values of identical positions in different replicates. This function must have a single output value, `vectors` are not allowed. The default `NULL` option returns an element-wise reorganization of the data.

proto (same as x[[1]]): The prototype for matching/merging. The prototype is used as a check ("dim") or a template ("name") during the matching process, depending on the used directive (direct argument). It is an object with the same class as the replicates, and have the same dimensions and/or overlapping names. If the "name" directive is used and a prototype is provided, the funtion will force the output to have the same structure as the prototype, by omitting unnecessary information and inserting missing values (NAs). The prototype is expected to be an object that has more or equal elements than the replicates, otherwise the call will result in a warning.

direct (character): Matching directive(s). Can either be dimension-based ("dim") and/or name-based (name). Dimension-based directive matches the replicates if they have the same dimesions. The "name" directive requires named input (for matrices and data.frames colnames and rownames attributes). Replicates will be matched if the values have the same names. In case both directives are specified (default), dimension-based directive takes higher priority, if matching is unsuccessful with dimensions, names will be tried after.

... arguments passed to FUN.

Details

The function is designed to unify/merge objects that result from the same function applied to different source data (e.g. the results of subsample()). In its current form, the function supports vectors (including one-dimensional tables and arrays), matrix and data.frame objects.

Value

If FUN is a function, the output is vector for vector-like replicates, matrix when x is a list of matrix objects, and data.frames for data.frame replicates. In case FUN=NULL: if x is a list of vectors, the function will return a matrix; an array is returned, if x is a list of matrix class obejcts; if x is a list of data.frame objects, the function returns a data.frame.

Examples

```
# basic example
vect <- rnorm(100)
# make 50 replicates
repl <- rep(list(vect), 50)
repmatch(repl, FUN=mean, direct="dim")

# named input
  # two vectors
    # a
    a<- 1:10
    names(a) <- letters[1:length(a)]
    a[c(3,5,8)] <- NA
    a <- a[!is.na(a)]

    #b
    b<- 10:1
    names(b) <- letters[length(b):1]
```

```
    b[c(1, 3,6, length(b))]<- NA
    b <- b[!is.na(b)]

  # list
  x2 <- rep(c(list(a),list(b)), 3)

# simple match - falling through "dim" to "name" directive
repmatch(x2, FUN=NULL)

# prototyped
prot <- 1:10
names(prot) <-letters[1:10]

repmatch(x2, FUN=mean, proto=prot, na.rm=TRUE)
```

`seqduplicated` *Determination and omission of consecutive duplicates in a vector.*

Description

`seqduplicated()` The function determines which elements of a vector are duplicates (similarly to `duplicated`) in consecutive rows.

`collapse()` Omits duplicates similarly to `unique`, but only in consecutive rows, so the sequence of state changes remains, but without duplicates.

Usage

```
seqduplicated(x, na.rm = FALSE, na.breaks = TRUE)

collapse(x, na.rm = FALSE, na.breaks = TRUE)
```

Arguments

`x`	(`vector`): input object.
`na.rm`	(`logical`): Are `NA` entries to be treated as duplicates (`TRUE`) or just like a normal value (`FALSE`)?
`na.breaks`	(`logical`): If `na.rm=TRUE` and the `NA` values are surrounded by the same values, should the streak be treated as broken? Running `seqduplicated(,na.rm=TRUE)` on `(2,1,NA,1)` while setting `na.breaks` to `TRUE` will return `(FALSE,FALSE,TRUE,FALSE)`, and with `TRUE` it will return `(FALSE,FALSE,TRUE,TRUE)`. The results with the same argumentation of `collapse()` will be `(2,1)` and `(2,1,1)`.

Details

These functions are essentially about checking whether a value in a vector at index is the same as the value at the previous index. This seamingly primitive task had to be rewritten with Rcpp for speed and the appropriate handling of `NA` values.

Value

A logical vector.

Examples

```
# example vector
  examp <- c(4,3,3,3,2,2,1,NA,3,3,1,NA,NA,5, NA, 5)

# seqduplicated()
  seqduplicated(examp)

  # contrast with
  duplicated(examp)

  # with NA removal
  seqduplicated(examp, na.rm=TRUE)

# the same with collapse()
  collapse(examp)

  # contrast with
  unique(examp)

  # with NA removal
  collapse(examp, na.rm=TRUE)

  # with NA removal, no breaking
  collapse(examp, na.rm=TRUE, na.breaks=FALSE)
```

---

shades *Quantile plot of time series*

Description

This intermediate-level function will plot a time series with the quantiles shown with transparency values.

Usage

```
shades(
  x,
  y,
  col = "black",
  res = 10,
  border = NA,
  interpolate = FALSE,
```

```
  method = "symmetric",
  na.rm = FALSE
)
```

Arguments

x	(numeric): The x coordinates.
y	(numeric matrix): The series of distributions to be plotted. Every row represents a distribution of values. The number of rows must equal to the length of x.
col	(character): The color of the quantiles, currently just a single color is allowed.
res	(numeric): If a single value is entered, than this argument represents the number of quantiles to be shown (coerced to 150, if higher is entered). If it is vector of values, it will be interpreted as the vector of quantiles to be shown. If method="symmetric", only an odd number of quantiles are plotted.
border	(character): The color of the quantile lines. A single value, by default, no lines are drawn (border=NA).
interpolate	(logical): In case the symmetric method is chosen, the series of quantile values can be interpolated with a LOESS function.
method	(character): The default "symmetric" method will plot the mid quantile range with highest opacity and the shades will be more translucent at the tails of the distributions. The "decrease" method will decrease the opacity with higher quantiles, which can make more sense for bottom-bounded distributions (e.g. exponential).
na.rm	(logical): If set to FALSE, than rows that are missing from the dataset will be plotted as gaps in the shading. If set to TRUE, than these gaps will be skipped.

Value

The function has no return value.

Examples

```
# some random values accross the Phanerozoic
data(stages)
tsplot(stages, boxes="sys", shading="series", ylim=c(-5,5), ylab=c("normal distributions"))
  randVar <- t(sapply(1:95, FUN=function(x){rnorm(150, 0,1)}))
  shades(stages$mid, randVar, col="blue", res=10,method="symmetric")

# a bottom-bounded distribution (log normal)
tsplot(stages, boxes="sys", shading="series", ylim=c(0,30), ylab="log-normal distributions")
  randVar <- t(sapply(1:95, FUN=function(x){rlnorm(150, 0,1)}))
  shades(stages$mid, randVar, col="blue", res=c(0,0.33, 0.66, 1),method="decrease")
```

`singletons` *List of singleton taxa*

Description

The function returns lists of taxa that occurr with only one particular entry in a given variable.

Usage

```
singletons(
  dat,
  tax = "clgen",
  var = NULL,
  bin = NULL,
  bybin = FALSE,
  na.rm = TRUE
)
```

Arguments

`dat`	(`data.frame`): Occurrence dataset, with `bin`, `tax` and `coll` as column names.
`tax`	(`character`): The name of the taxon variable.
`var`	(`character`): The variable that is used to define singletons. Use the reference variable for single-reference taxa, and the collection variable for single-collection taxa, the bin identifier for single-interval taxa and so forth. If you set this to the default `NULL`, the function will return single-occurrence taxa.
`bin`	(`character`): Lists of taxa can be tabulated in every bin. Rows with `NA` entries in this column will be omitted.
`bybin`	(`logical`): The type of the filtering process. Was it supposed to be applied to bin-specific subsets (`TRUE`), or the whole data (`FALSE`)? Setting this argument to `TRUE` will return a `list` class object, where every element of the list is a bin-specific `character` vector. This settig also removes all `NA` entries form `bin` variable.
`na.rm`	(`logical`): If `var` is not `NULL`, setting this argument to `TRUE` removes all rows where var is `NA`. Otherwise these will be returned as singletons.

Details

Singletons are defined in number of ways in the literature. True singletons are species that are represented by only one specimen, but one can talk about single-occurrence, single-interval, single-reference or single collection taxa as well. These can be returned with this function.

As the time bin has particular importance, it is possible to filter singleton taxa in the context of a single bin. These can be returned with the `bybin` argument, that constrains and iterates the filtering to every bin. If this argument is set to `TRUE` and the variable in question is a references, than single-reference taxa will be taxa that occurred in only one reference within each bin - it does not necessarily mean that only one reference describes the taxon in the total database!

Value

A vector of character entries in tax.

Examples

```
# load example dataset
  data(corals)

# Example 1. single-occurrence taxa
  singOcc <- singletons(corals, tax="genus", bin="stg")

# Example 2. output for every bin
  singOccBin <- singletons(corals, tax="genus", bin="stg", bybin=TRUE)

# Example 3. single-interval taxa (all)
  singInt <- singletons(corals, tax="genus", var="stg")

# Example 4. single interval taxa (for every bin)
  singIntBin <- singletons(corals, tax="genus", var="stg", bin="stg", bybin=TRUE)

# Example 5. single reference taxa (total dataset)
  singRef <- singletons(corals, tax="genus", var="reference_no")

#  Example 6. single reference taxa (see description for differences )
  singRefBin <- singletons(corals, tax="genus", var="reference_no", bin="stg", bybin=TRUE)
```

---

slice *Discretization of continuous time dimension - slicing*

Description

The function will slices time with a given set of boundaries and produce a time scale object if desired.

Usage

```
slice(x, breaks, offset = 0, ts = TRUE, revtime = TRUE)
```

Arguments

x	(numeric) Vector of continouos age/time estimates.
breaks	(numeric) Vector of boundaries, the breaks argument of the cut function
offset	(numeric) Single value. If desired the resulting integer bin numbers can be offset by some amount.
ts	(logical) Should a time scale object be also produced when the function is run?
revtime	(logical) Should the time dimension be reversed? This argument is set to TRUE by default, meaning that the function will reverse the order of time: smaller values of x will be translated to higher values (slc) in the function output.

Details

Due to stratigraphic constraints, we can only process deep time data, when it is sliced to discrete bins. It is suggested that you do this separately for most of your analyses. This function is also used by the codedivDyn function when age entries are provided.

Value

Either of new entries and levels or time scale.

Examples

```
y<- runif(200, 0,100)
au <- slice(y, breaks=seq(0, 100, 10))
withOut <- slice(y, breaks=seq(0, 100, 10), ts=FALSE)
```

`stages` *95 bin Phanerozoic time scale based on the stratigraphic stages of Ogg et al. (2016).*

Description

Stage-level (age-level) timescale used in some analyses.

Usage

```
data(stages)
```

Format

A `data.frame` with 95 observations and 10 variables:

`sys` Abbreviations of geologic systems.

`system` Geologic periods.

`series` Geologic series.

`stage` Names of geologic stages.

`short` Abbreviations of geologic stages.

`bottom` Numeric ages of the bottoms boundaries (earliest ages) of the bins.

`mid` Numeric age midpoints of the bins, the averages of `bottom` and `top`.

`top` Numeric ages of the tops (latest ages) of the bins.

`dur` Numeric ages of the durations for the bins.

`stg` Integer number identifiers of the bins.

`systemCol` Hexadecimal color code of the systems.

`seriesCol` Hexadecimal color code of the series.

`col` Hexadecimal color code of the stages.

Details

This is an example time scale object that can be used in the Phanerozoic-scale analyses. Example occurrence datasets related to the package use the variable `stg` when referring to this timescale.

References

Ogg, J. G., G. Ogg, and F. M. Gradstein. 2016. A concise geologic time scale: 2016. Elsevier.

Source

Based on Ogg et al. (2016), compiled by Wolfgang Kiessling.

`stratkeys` *The FossilWorks-based lookup table for the stratigraphic assignments of collections in the Paleobiology Database*

Description

Table including the user-chosen interval data and the stratigraphic units of the dynamic timescale.

Usage

```
data(stratkeys)
```

Format

A `data.frame` with 761 observations of 8 variables:

`interval` The names of the registered intervals in the `early_interval`/`max_interval` and `late_interval`/`min_interval` columns.

`period` The period containing the interval.

`epoch` The epoch containing the interval.

`X10_my_bin` The 10 million year time scale interval containing the interval.

`ten` Numeric identifier of the 10 million year interval in the `tens` object.

`stage` The stage containing the interval.

`stg` Numeric identifier of the interval in the stage-level time scale provided as `stages` object.

Details

Since the separation of the FossilWorks (`http://fossilworks.org/`) portal from the Paleobiology Database (`https://paleobiodb.org/`) the access to the stratigraphic information in the database have been problematic. This table includes groupings of `early_interval`/`max_interval` entries of the dynamic timescale that users can choose during collection entry. The table assigns these intervals to some corresponding stratigraphic units from different time scales. These entries were distilled from those collections that only have a `max_interval` value. As there is a mismatch

between the data Paleobiology Database and FossilWorks this list is not comprehensive and a couple entries are probably missing. For this reason, this dataset is expected to be updated in the future.

This particular version (v0.9.2) is based on a download of all collections in FossilWorks between the Ediacaran and the Holocene. The download took place on 22 June, 2018. The entries were transformed to `keys` to be used with the `categorize` function. Some entries were corrected manually.

Source

http://fossilworks.org/

`streaklog` *Utility functions for slicing gappy time series*

Description

The function returns where the continuous streaks start and how long they are, which can be used for efficient and flexible subsetting.

Usage

```
streaklog(x)

whichmaxstreak(x, which = -1)
```

Arguments

`x`	(`vector`) A vector with missing values.
`which`	(`integer`) In case multiple streaks of the same length are found, which of them should be returned by the vector (integer).

Details

The output list of `streaklog` contains the following elements:

`starts`: the indices where the streaks start.

`streaks`: the lengths of the individual streaks (number of values).

`runs`: the number of streaks.

The function whichmaxstreak() will return the indices of those values that are in the longest continuous streak.

Value

A list (streaklog) or a numeric vector (whichmaxstreak).

Examples

```
# generate a sequence of values
  b<-40:1
# add some gaps
  b[c(1:4, 15, 19, 23:27)]  <- NA
# the functions
  streaklog(b)
  whichmaxstreak(b)
```

`subsample` *Subsampling wrapper function*

Description

The function will take a function that has an occurrence dataset as an argument, and reruns it iteratively on the subsets of the dataset.

Usage

```
subsample(
  x,
  q,
  tax = NULL,
  bin = NULL,
  FUN = divDyn,
  coll = NULL,
  iter = 50,
  type = "cr",
  keep = NULL,
  rem = NULL,
  duplicates = TRUE,
  output = "arit",
  useFailed = FALSE,
  FUN.args = NULL,
  na.rm = FALSE,
  counter = TRUE,
  ...
)
```

Arguments

`x`	(`data.frame`): Occurrence dataset, with `bin`, `tax` and `coll` as column names.
`q`	(`numeric`): Subsampling level argument (mandatory). Depends on the subsampling function, it is the number of occurrences for `"cr"`, and the number of desired occurrences to the power of `x`exp for O^x^W. It is also the quorum of the SQS method.
`tax`	(`character`): The name of the taxon variable.

bin (character): The name of the subsetting variable (has to be integer). For time series, this is the time-slice variable. Rows with NA entries in this column will be omitted.

FUN (function): The function to be iteratively executed on the results of the subsampling trials. If set to NULL, no function will be executed, and the subsampled datasets will be returned as a list. By default set to the divDyn function. The function must have an argument called x, that represents the dataset resulting from a subsampling trial (or the entire dataset). Arguments of the subsample function call will be searched for potential arguments of this function, which means that already provided variables (e.g. bin and tax) will also be used. You can also provide additional arguments (similarly to the apply iterator). Functions that allow arguments to pass through (that have argument '...') are not allowed, as well as functions that have the same arguments as subsample but would require different values.

coll (character): The variable name of the collection identifiers.

iter (numeric): The number of iterations to be executed.

type (character): The type of subsampling to be implemented. By default this is classical rarefaction ("cr"). ("oxw") stands for occurrence weighted by-list subsampling. If set to ("sqs"), the program will execute the shareholder quorum subsampling algorithm as it was suggested by Alroy (2010). Setting the argument to "none" will invoke no subsamling, but the applied function will be iterated on the trials, nevertheless.

keep (numeric): The bins, which will not be subsampled but will be added to the subsampling trials. NIf the number of occurrences does not reach the subsampling quota, by default it will not be represented in the subsampling trials. You can force their inclusion with the keep argument separetely (for all, see the useFailed argument).

rem (numeric): The bins, which will be removed from the dataset before the subsampling trials.

duplicates (logical): Toggles whether multiple entries from the same taxon ("tax") and collection ("coll") variables should be omitted. Useful for omitting occurrences of multiple species-level occurrences of the same genus. By default these are allowed through analyses (duplicates=TRUE), setting this to FALSE will require you to provide a collection variable. (coll)

output (character): If the function output are vectors or matrices, the "arit" and "geom" values will trigger simple averaging with arithmetic or geometric means. If the function output of a single trial is again a vector or a matrix, setting the output to "dist" will return the calculated results of every trial, organized in a list of independent variables (e.g. if the function output is value, the return will contain a single vector, if it is a vector, the output will be a list of vectors, if the function output is a data.frame, the output will be a list of matrix class objects). If output="list", the structure of the original function output will be retained, and the results of the individual trials will be concatenated to a list.

useFailed (logical): If the bin does not reach the subsampling quota, should the bin be used?

FUN.args	(list): Arguments passed to the applied function FUN but not used by the subsampling wrapper. Normally, the arguments of FUN can be added to the call of subsample, but in case you want to use different values for the same argument, then the arguments added here will be used for the call of FUN. For instance, if you want to call subsample with bin=NULL, but want to run FUN=divDyn with a valid bin column then you can add the column name here, e.g. FUN.args=list(bin="stg").
na.rm	(logical): The function call includes more column names that might contain missing values. If this flag is set to TRUE, all rows will be dropped that have missig values in the specificed columns. This might lead to the exclusion of some data you do not want to exclude.
counter	(logical): Should the loop counting be visible?
...	arguments passed to FUN and the type-specific subsampling functions: subtrialCR, subtrialOXW, subtrialSQS

Details

The subsample function implements the iterative framework of the sampling standardization procedure. The function 1. takes the dataset x, 2. runs function FUN on the dataset and creates a container for results of trials 3. runs one of the subsampling trial functions (e.g. subtrialCR) to get a subsampled 'trial dataset' 4. runs FUN on the trial dataset and 5. averages the results of the trials for a simple output of step 4. such as vectors, matrices and data.frames. For averaging, the vectors and matrices have to have the same output dimensions in the subsampling, as in the original object. For data.frames, the bin-specific information have to be in rows and the bin numbers have to be given in a variable bin in the output of FUN. For a detailed treatment on what the function does, please see the vignette ('Handout to the R package 'divDyn' v0.5.0 for diversity dynamics from fossil occurrence data'). Currently the Classical Rarefaction ("cr", Raup, 1975), the occurrence weighted by-list subsampling ("oxw", Alroy et al., 2001) and the Shareholder Quorum Subsampling methods are implemented ("sqs", Alroy, 2010).

References:

Alroy, J., Marshall, C. R., Bambach, R. K., Bezusko, K., Foote, M., Fürsich, F. T., ... Webber, A. (2001). Effects of sampling standardization on estimates of Phanerozoic marine diversification. Proceedings of the National Academy of Science, 98(11), 6261-6266.

Alroy, J. (2010). The Shifting Balance of Diversity Among Major Marine Animal Groups. Science, 329, 1191-1194. https://doi.org/10.1126/science.1189910

Raup, D. M. (1975). Taxonomic Diversity Estimation Using Rarefaction. Paleobiology, 1, 333-342. https: //doi.org/10.2307/2400135

Value

Either a list of replicates or an object matching the class of FUN.

Examples

```
data(corals)
data(stages)
# Example 1-calculate metrics of diversity dynamics
```

```
  dd <- divDyn(corals, tax="genus", bin="stg")
  rarefDD<-subsample(corals,iter=30, q=50,
  tax="genus", bin="stg", output="dist", keep=95)

# plotting
  tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
  ylab="range-through diversity (genera)", ylim=c(0,230))
  lines(stages$mid, dd$divRT, lwd=2)
  shades(stages$mid, rarefDD$divRT, col="blue")
  legend("topleft", legend=c("raw","rarefaction"),
    col=c("black", "blue"), lwd=c(2,2), bg="white")


# Example 2-SIB diversity
# draft a simple function to calculate SIB diversity
sib<-function(x, bin, tax){
  calc<-tapply(INDEX=x[,bin], X=x[,tax], function(y){
    length(levels(factor(y)))
  })
  return(calc[as.character(stages$stg)])
}
sibDiv<-sib(corals, bin="stg", tax="genus")

# calculate it with subsampling
rarefSIB<-subsample(corals,iter=25, q=50,
  tax="genus", bin="stg", output="arit", keep=95, FUN=sib)
rarefDD<-subsample(corals,iter=25, q=50,
  tax="genus", bin="stg", output="arit", keep=95)

# plot
tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
  ylab="SIB diversity (genera)", ylim=c(0,230))

lines(stages$mid, rarefDD$divSIB, lwd=2, col="black")
lines(stages$mid, rarefSIB, lwd=2, col="blue")


# Example 3 - different subsampling types with default function (divDyn)
# compare different subsampling types
  # classical rarefaction
  cr<-subsample(corals,iter=25, q=20,tax="genus", bin="stg", output="dist", keep=95)
  # by-list subsampling (unweighted) - 3 collections
  UW<-subsample(corals,iter=25, q=3,tax="genus", bin="stg", coll="collection_no",
    output="dist", keep=95, type="oxw", xexp=0)
  # occurrence weighted by list subsampling
  OW<-subsample(corals,iter=25, q=20,tax="genus", bin="stg", coll="collection_no",
    output="dist", keep=95, type="oxw", xexp=1)

 SQS<-subsample(corals,iter=25, q=0.4,tax="genus", bin="stg", output="dist", keep=95, type="sqs")

# plot
  tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
  ylab="range-through diversity (genera)", ylim=c(0,100))
```

```
  shades(stages$mid, cr$divRT, col="red")
  shades(stages$mid, UW$divRT, col="blue")
  shades(stages$mid, OW$divRT, col="green")
  shades(stages$mid, SQS$divRT, col="cyan")

  legend("topleft", bg="white", legend=c("CR (20)", "UW (3)", "OW (20)", "SQS (0.4)"),
    col=c("red", "blue", "green", "cyan"), lty=c(1,1,1,1), lwd=c(2,2,2,2))
```

---

subtrialCR *Subsampling trial functions*

Description

These functions create one subsampling trial dataset with a desired subsampling method

Usage

```
subtrialCR(
  x,
  q,
  bin = NULL,
  unit = NULL,
  keep = NULL,
  useFailed = FALSE,
  showFailed = FALSE
)

subtrialOXW(
  x,
  q,
  bin = NULL,
  coll = NULL,
  xexp = 1,
  keep = NULL,
  useFailed = FALSE,
  showFailed = FALSE
)

subtrialSQS(
  x,
  tax,
  q,
  bin = NULL,
  coll = NULL,
  ref = NULL,
  singleton = "occ",
```

```
  excludeDominant = FALSE,
  largestColl = FALSE,
  fcorr = "good",
  byList = FALSE,
  keep = NULL,
  useFailed = FALSE,
  showFailed = FALSE,
  appr = "under"
)
```

Arguments

x	(data.frame): Occurrence dataset, with bin, tax and coll as column names.
q	(numeric): Subsampling level argument (mandatory). Depends on the subsampling function, it is the number of occurrences for "cr", and the number of desired occurrences to the power of xexp for O^x^W. It is also the quorum of the SQS method.
bin	(character): The name of the subsetting variable (has to be integer). For time series, this is the time-slice variable. Rows with NA entries in this column will be omitted.
unit	(character): Argument of the CR subsampling type. The name of the variable that designates the subsampling units. In every bin, CR selects a certain number (quota) of entries from the dataset. By default (unit=NULL), the units will be the rows, and the q number of rows will be selected in each bin. However, this can be a higher level category that has multiple entries in the each bin. If unit is a valid column of the dataset x, then CR will select q number entries in this variable, and will return all the corresponding rows.
keep	(numeric): The bins, which will not be subsampled but will be added to the subsampling trials. NIf the number of occurrences does not reach the subsampling quota, by default it will not be represented in the subsampling trials. You can force their inclusion with the keep argument separetely (for all, see the useFailed argument). Only applicable when bin!=NULL.
useFailed	(logical): If the bin does not reach the subsampling quota, should the bin be used? If bin!=NULL and useFailed=TRUE then only TRUE values will be output (indicating the use of the full dataset).
showFailed	(logical): Toggles the output of the function. If set to TRUE the output will be a list, including both the default output (logical vector of rows) and the numeric vector of bins that did not have enough entries to reach the quota q. Only applicable when bin!=NULL.
coll	(character): The variable name of the collection identifiers.
xexp	(numeric): Argument of the OxW type. The exponent of by-list subsampling, by default it is 1.
tax	(character): The name of the taxon variable.
ref	(character): The name of the reference variable, optional - depending on the subsampling method.

`singleton` (`character`): A parameter of SQS. Either `"ref"`, `"occ"` or `FALSE`. If set to `"occ"`, the coverage estimator (e.g. Good's u) will be calculated based on the number of single-occurrence taxa. If set to "ref" the number of occurrences belonging to single-reference taxa will be used instead. In case of the inexact algorithm, if set to `FALSE` then coverage corrections of frequencies will not be applied.

`excludeDominant` (`logical`): Argument of SQS. This parameter sets whether the dominant taxon should be excluded from all calculations involving frequencies (this is the second correction of Alroy, 2010).

`largestColl` (`logical`): Parameter of SQS. This parameter sets whether the occurrences of taxa only ever found in the most diverse collection should be excluded from the count of single-publication occurrences. (this is the third correction of Alroy, 2010) Note that `largestColl=TRUE` is dependent on `excludeDominant=TRUE`. Setting `excludeDominant` to `FALSE` will turn this correction off.

`fcorr` (`character`): Parameter for the inexact method of SQS. either "good" or "alroy". This argument changes the frequency correction procedure of the 'inexact' version of SQS (Alroy 2010). As not all taxa are present in the samples, the sampled frequencies of taxa tend overestimate their frequencies in the sampling pool. In Alroy (2010) these are corrected using Good's u ("good", default), in the later versions of SQS this metric is changed to a different method using single occurrence and double occurrence taxa ("alroy").

`byList` (`character`): A parameter of the `"inexact"` method of SQS. Sets whether occurrences should be subsampled with (`FALSE`) or without (`TRUE`) breaking the collection integrity.

`appr` (`character`): A parameter of the inexact method of SQS. Either "over" (default) or ("under"). The current version is not concerned with small fluctuations around the drawn subsampling quorum. Therefore, in the inexact algorithm, sampling is finished when the subset either is immediately below the quorum (`"under"`) or above it (`"over"`).

Details

The essence of these functions are present within the subsampling wrapper function `subsample`. Each function implements a certain subsampling type. The return value of the funcfions by default is a `logical` vector indicating which rows of the original dataset should be present in the subsample. The inexact method for SQS is implemented here as it is computationally less demanding.

References:

Alroy, J., Marshall, C. R., Bambach, R. K., Bezusko, K., Foote, M., Fürsich, F. T., ... Webber, A. (2001). Effects of sampling standardization on estimates of Phanerozoic marine diversification. Proceedings of the National Academy of Science, 98(11), 6261-6266.

Alroy, J. (2010). The Shifting Balance of Diversity Among Major Marine Animal Groups. Science, 329, 1191-1194. https://doi.org/10.1126/science.1189910

Raup, D. M. (1975). Taxonomic Diversity Estimation Using Rarefaction. Paleobiology, 1, 333-342. https: //doi.org/10.2307/2400135

Value

A logical vector.

Examples

```
#one classical rarefaction trial
  data(corals)
# return 5 references for each stage
  bRows<-subtrialCR(corals, bin="stg", unit="reference_no", q=5)
  # control
  unCor<-unique(corals[bRows,c("stg", "reference_no")])
  table(unCor$stg)
```

---

sumstat *Occurrence database summary*

Description

The function calculates global statistics of the entire database

Usage

```
sumstat(
  x,
  tax = "genus",
  bin = "stg",
  coll = NULL,
  ref = NULL,
  duplicates = NULL
)
```

Arguments

x	(data.frame): The occurrence dataset.
tax	(character): The column name of taxon names.
bin	(character): The column name of bin names.
coll	(character): The column name of collection numbers. (optional)
ref	(character): The column name of reference numbers. (optional)
duplicates	(logical): The function will check whether there are duplicate occurrences (multiple species/genera). When set to NULL, nothing will happen, but the function will notify you if duplicates are present. If set to TRUE, the function will not do anything with these, if set to FALSE, the duplicates will be omitted.

Details

The function returns the following values.

`bins`: The total number of bins sampled.

`occs`: The total number of sampled occurrences.

`colls`: The total number of sampled collections.

`refs`: The total number of sampled references.

`taxa`: The total number of sampled taxa.

`gappiness`: The proportion of sampling gaps in the ranges of the taxa (without the range-endpoints).

Value

A named numeric vector.

Examples

```
data(corals)
  sumstat(corals, tax="genus", bin="stg", coll="collection_no", ref="reference_no")
```

---

survivors *Proportions of survivorship*

Description

This function will calculate both forward and backward survivorship proportions from a given occurrence dataset or FAD-LAD matrix.

Usage

```
survivors(
  x,
  tax = "genus",
  bin = "stg",
  method = "forward",
  noNAStart = FALSE,
  fl = NULL
)
```

Arguments

`x`	(`data.frame`) The data frame containing fossil occurrences.
`tax`	(`character`) The variable name of the occurring taxa (variable type: `factor` or `character`).
`bin`	(`character`) The variable name of the time slice numbers of the particular occurrences (variable type: `numeric`). Bin numbers should be in ascending order,can contain `NA`s, it can start from a number other than 1 and must not start with 0.

method	(character) Either "forward" or "backward".
noNAStart	(logical) Useful when the dataset does not start from bin 1. Then noNAStart=TRUE will cut the first part of the resulting table, so the first row will contain the estimates for the lowest bin number.
fl	(matrix or data.frame). If so desired, the function can be run on an FAD-LAD dataset, output by the fadlad function.

Details

Proportions of survivorship are great tools to visualize changes in the composition of a group over time (Raup, 1978). The curves show how a once coexisting set of taxa, called a cohort, loses its participants (forward survivorship) as time progress, or gains its elements as time is analyzed backwards. Each value corresponds to a cohort in a bin (*a*) and one other bin (*b*). The value expresses what proportion of the analyzed cohort (present together in bin *a*) is present in bin *b*.

References:

Raup, D. M. (1978). Cohort analysis of generic survivorship. Paleobiology, 4(1), 1-15.

Value

A numeric matrix of survivorship probabilities.

Examples

```
data(corals)
surv<-survivors(corals, tax="genus", bin="stg", method="forward")

# plot
data(stages)
tsplot(stages, shading="series", boxes="sys", xlim=c(260,0),
  ylab="proportion of survivors present", ylim=c(0.01,1),plot.args=list(log="y"))

for(i in 1:ncol(surv)) lines(stages$mid, surv[,i])
```

---

tabinate *Apply function to TAxon/BIN subset of occurrences and iterATE*

Description

The function takes another function and reruns it on every taxon- and/or bin-specific subsets of an occurrence dataset.

Usage

```
tabinate(x, bin = NULL, tax = NULL, FUN = NULL, ...)
```

Arguments

x	(data.frame) Fossil occurrence table.
bin	(character) Variable name of the bin numbers of the occurrences. This variable should be numeric.
tax	(character) Variable name of the occurring taxa (variable type: factor or character - such as "genus"
FUN	(function) The function applied to the subset of occurrences. The subset of occurence data will be passed to this function as x.
...	arguments passed to FUN

Details

The main tabinate function acts as a wrapper for any type of function that requires a subset of the occurrence dataset that represents either one bin or one tax entry or both. For example, the iterator can be used to calculate geographic ranges from occurrence coordinates (georange).

The output structure of FUN should be independent from the input subset, or the function will return an error. Setting both bin If bin=NULL and codetax=NULL, will run FUN on the entire dataset (no effect). Providing either bin or tax and keeping the other NULL will iterate FUN for every bin or tax entry (whichever is presented). The function returns a vector of values if the return value of FUN is a single value. In case it is a vector, the final output will be a matrix. When both bin and tax is presented, the function output will be a matrix (one output value for a taxon/bin subset) or an array (3d, when FUN returns a vector). Setting FUN to NULL will return the occurrence dataset as lists.

Value

The return object depends on the output of FUN, as well as the bin and tax input.

Examples

```
 data(corals)

# the number of different coordinate pairs in every time slice
  tabinate(corals, bin="stg", FUN=georange, lat="paleolat",
    lng="paleolng", method="co")
# geographic range (site occupancy) of every taxon in every bin
  tabinate(corals, bin="stg", tax="genus", FUN=georange,
    lat="paleolat", lng="paleolng", method="co")
```

---

tens *The 10 million year resolution timescale of the Paleobiology Database*

Description

Roughly 10 million year timescale used in some analyses.

Usage

```
data(tens)
```

Format

A `data.frame` with 49 observations and 9 variables:

X10 The name of the bin: Period and number.

ocean The primary state of the oceans from the point of carbonate precipitation. `ar` indicates aragonitic, `cc` indicates calcitic conditions. Based on §

ocean2 §

climate Primary climatic characteristic: `w` denotes warm, `c` denotes cold.

`bottom` Numeric ages of the bottom boundaries (earliest ages) of the bins.

`mid` Numeric ages midpoints of the bins, the averages of `bottom` and `top`.

`top` Numeric ages of the tops (latest ages) of the bins.

`dur` Numeric ages of the durations of the bins.

`ten` Integer number identifiers of the bins. §correct to num!

Details

This is an example time scale object that can be used in the Phanerozoic scale analyses. This time scale comprises 49 bins, roughly 10 million years of durations that result from the combination of certain standard stages.

Source

Executive committee meeting (2015) of old Paleobiology Database. Additional variables were added by Wolfgang Kiessling.

---

tsbars *Function to plot a series a values with bars that have variable widths*

Description

Function to use bars for time series.

Usage

```
tsbars(x, y, width = "max", yref = 0, gap = 0, vertical = TRUE, ...)
```

Arguments

x	(numeric) Vector specifying where the centers of the bars should be on the x axis.
y	(numeric) Vector containing the heights of the bars.
width	(numeric) Vector containing the widths of the bars. Recycling is not supported, has to be either a single numeric value, or a numeric vector with the same length as x and y. Automatic width calculation is possible, the default "max" option sets the bar width even and equal to the the maximum width that can be used evenly witout causing overlaps. The option "half", places the boundaries of the bars halfway between the points. This will make the bars' width asymmetrical around the x coordinates.
yref	(numeric) Single numeric value in the y dimension indicating common base for the bars.
gap	(numeric) The amount of gap there should be between the bars (in the unit of the plotting). Defaults to no gaps.
vertical	(logical) Switching this option to FALSE will reverse the x and y dimensions of the plot.
...	Arguments passed to rect.

Details

People often present time series with connected points, although the visual depiction implies a certain process that describes how the values change between the points. Instead of using simple scatter plots, Barplots can be used to describe series where a single value is the most descriptive of a discreet time bin. The tsbars() function draws rectangles of different widths with the rect function, to plot series in such a way.

Value

The function has no return value.

Examples

```
# an occurrence-based example
# needed data
  data(stages)
  data(corals)
# calculate diversites
  dd <-divDyn(corals, tax="genus", bin="stg")
# plot range-through diversities
  tsplot(stages, xlim=51:94, ylim=c(0,250), boxes="sys")
  tsbars(x=stages$mid, y=dd$divRT, width=stages$dur, gap=1, col=stages$col)
```

tsplot *Time series plotting using a custom time scale*

Description

This function allows the user to quickly plot a time scale data table

Usage

```
tsplot(
  tsdat,
  ylim = c(0, 1),
  xlim = NULL,
  prop = 0.05,
  gap = 0,
  bottom = "bottom",
  top = "top",
  xlab = "Age (Ma)",
  ylab = "",
  boxes = NULL,
  boxes.col = NULL,
  shading = NULL,
  shading.col = c("white", "gray80"),
  plot.args = NULL,
  boxes.args = NULL,
  labels = TRUE,
  labels.args = NULL,
  lplab = TRUE,
  rplab = TRUE
)
```

Arguments

tsdat	(data frame): The time scale data frame.
ylim	(numeric): The vertical extent of the plot, analogous to the same argument of plot. By default it is set to the [0,1] interval.
xlim	(numeric): The horizontal extent of the plot, analogous to the same argument of plot. By default it is set to plot the entire table. If a numeric vector with two values is supplied, it will be interpreted as the standard xlim argument and the plot will be displayed based on numerically constrained ages. If it is an integer vector with more than two values are plotted, the interval corresponding to the row indices of the table will be plotted.
prop	(numeric): Proportion of the vertical extent of the plot to display the the time scale at the bottom.
gap	(numeric): Proportion of the vertical extent of the plot that should be a gap betwen the time scale and the plot.

`bottom` (`character`): Column name of the table for the variable that contains the older ages of intervals.

`top` (`character`): Column name of the table for the variable that contains the earliest ages of intervals.

`xlab` (`character`): The label of the time axis.

`ylab` (`character`): The label of the data axis.

`boxes` (`character`): Column name indicating the names that should be plotted as boxes of the timescale.

`boxes.col` (`character`): Column name of the colour codes for the boxes. Each entry in this column has to correspond to an entry in the `boxes` column. It also overrides the `col` entries in the `boxes.args` argument.

`shading` (`character`): Column name used for the shading. By default, no shading will be drawn (`shading = NULL`).

`shading.col` (`character`): Colors that will be used for the shading, if shading is set. It is either a single column of the `tsdat` object with color codes, or multiple color entries. The provided colors will be repeated as many times as necessary.

`plot.args` (`list`): Arguments that will be passed to the main `plot` function. Can be useful for the suppression of axes, font change etc.

`boxes.args` (`list`): Arguments that will be passed to the `rect` function that draws the rectangles of time intervals.

`labels` (`logical`): Should the labels within the boxes be drawn? Setting this argumnet to `FALSE` will not call the `text` function that draws the labels.

`labels.args` (`list`): Arguments that will be passed to the `text` function that draws the labels. Can be `lists` of `lists` if multiple series of "boxes" are used.

`lplab` `logical`: When the left boundary of the plot does not match with any of the boundaries of the time scale boxes (and `labels=TRUE`), should the label of the partially drawn box be plotted?

`rplab` `logical`: When the right boundary of the plot does not match with any of the boundaries of the time scale boxes (and `labels=TRUE`), should the label of the partially drawn box be plotted?

Details

As most analysis use an individually compiled time scale object, in order to ensure compatibility between the analyzed and plotted values, the time scale table used for the analysis could be plotted rather than a standardized table. Two example tables have been included in the package (`stages` and `tens`) that can serve as templates.

Value

The function has no return value.

Examples

```
data(stages)
  tsplot(stages, boxes="sys", shading="series")
 # same with colours
  tsplot(stages, boxes="sys", shading="series", boxes.col="systemCol")

# only the Mesozoic, custom axes
  tsplot(stages, boxes="system", shading="stage", xlim=52:81,
    plot.args=list(axes=FALSE, main="Mesozoic"))
  axis(1, at=seq(250, 75, -25), labels=seq(250, 75, -25))
  axis(2)

# only the Triassic, use the supplied abbreviations
  tsplot(stages, boxes="short", shading="stage", xlim=c(250,199),
    ylab="variable", labels.args=list(cex=1.5, col="blue"),
    boxes.args=list(col="gray95"))

 # colourful plot with two levels of hierarchy
   tsplot(stages, boxes=c("short", "system"), shading="series",
     boxes.col=c("col", "systemCol"), xlim=c(52:69))
```

Index