Package 'palaeoverse'

February 16, 2023

Title Prepare and Explore Data for Palaeobiological Analyses

Version 1.1.1

Description Provides functionality to support data preparation and exploration for palaeobiological analyses, improving code reproducibility and accessibility. The wider aim of 'palaeoverse' is to bring the palaeobiological community together to establish agreed standards. The package currently includes functionality for data cleaning, binning (time and space), exploration, summarisation and visualisation. Reference datasets (i.e. Geological Time Scales <https://stratigraphy.org/chart>)
and auxiliary functions are also provided. Details can be found in:
Jones et al., (2022) <doi:10.31223/X5Z94Q>.

License GPL (>= 3)

Language en-GB

Encoding UTF-8

LazyData true

RoxygenNote 7.2.3

Depends R (>= 4.0)

Imports stats, utils, graphics, methods, curl, deeptime (>= 1.0.0), ape, sf, stringdist, geosphere, h3jsr (>= 1.3.0), httr, pbapply

Suggests rmarkdown, knitr, testthat (>= 3.0.0), vdiffr (>= 1.0.0), paleotree, phytools, covr

VignetteBuilder knitr

Config/testthat/edition 3

URL https://palaeoverse.palaeoverse.org, https://github.com/palaeoverse-community/palaeoverse, https://palaeoverse.org

BugReports https://github.com/palaeoverse-community/palaeoverse/issues

NeedsCompilation no

Author Lewis A. Jones [aut, cre] (<https://orcid.org/0000-0003-3902-8986>),
William Gearty [aut] (<https://orcid.org/0000-0003-0076-3262>),
Bethany J. Allen [aut] (<https://orcid.org/0000-0003-0282-6407>),
Kilian Eichenseer [aut] (<https://orcid.org/0000-0002-0477-8878>),
Christopher D. Dean [aut] (<https://orcid.org/0000-0001-6471-6903>),
Sofia Galvan [ctb] (<https://orcid.org/0000-0002-3092-4314>),
Miranta Kouvari [ctb] (<https://orcid.org/0000-0002-5442-6221>),
Pedro L. Godoy [ctb] (<https://orcid.org/0000-0003-4519-5094>),
Cecily Nicholl [ctb] (<https://orcid.org/0000-0003-2860-2604>),
Lucas Buffan [ctb] (<https://orcid.org/0000-0002-2353-1432>),
Erin M. Dillon [ctb] (<https://orcid.org/0000-0003-0249-027X>),
Joseph T. Flannery-Sutherland [aut]
(<https://orcid.org/0000-0001-8232-6773>),
A. Alessandro Chiarenza [ctb] (<https://orcid.org/0000-0001-5525-6730>)

Maintainer Lewis A. Jones <LewisAlan.Jones@uvigo.es>

Repository CRAN

Date/Publication 2023-02-16 09:50:02 UTC

R topics documented:

axis_geo *Add an axis with a geological timescale*

Description

axis_geo behaves similarly to axis in that it adds an axis to the specified side of a base R plot. The main difference is that it also adds a geological timescale between the plot and the axis. The default scale includes international periods from ICS. However, international epochs, stages, eons, and eras and any interval data hosted by Macrostrat are also available from the deeptime package (see get_scale_data). A custom interval dataset can also be provided (see Details below). The appearance of the axis is highly customizable (see Usage below), with the intent that plots will be publication-ready.

Usage

```
axis_geo(
  side = 1,
  intervals = "epochs",
  height = 0.05,
  fill = NULL,
  lab = TRUE,
  lab_col = NULL,
  lab_size = 1,
  rot = 0,
  abbr = TRUE,
  center_end_labels = TRUE,
  skip = c("Quaternary", "Holocene", "Late Pleistocene"),
  bord_col = "black",
  lty = par("lty"),
  lwd = par("lwd"),
  bkgd = "grey90",
  neg = FALSE,
  exact = FALSE,
  round = FALSE,
  tick_at = NULL,
  tick_labels = TRUE,
  phylo = FALSE,
  root.time = NULL,
  ...
)

axis_geo_phylo(...)
```

Arguments

side	integer. Which side to add the axis to (1: bottom, the default; 2: left; 3: top; 4: right).

intervals — The interval information to use to plot the axis: either A) a character string indicating a built-in or remotely hosted data.frame (see get_scale_data), or B) a custom data.frame of time interval boundaries (see Details).

height — numeric. The relative height (or width if side is 2 or 4) of the scale. This is relative to the height (if side is 1 or 3) or width (if side is 2 or 4) of the plot.

fill — character. The fill color of the boxes. The default is to use the color column included in intervals. If a custom dataset is provided with intervals without a color column and without specifying fill, a greyscale will be used. Custom fill colors can be provided with this option (overriding the color column) and will be recycled if/as necessary.

lab — logical. Should interval labels be included?

lab_col — character. The color of the labels. The default is to use the lab_color or lab_colour column included in intervals. If a custom dataset is provided with intervals without a lab_color or lab_colour column and without specifying lab_col, all labels will be black. Custom label colors can be provided with this option (overriding the lab_color or lab_colour column) and will be recycled if/as necessary.

lab_size — numeric. The size of the labels (see cex in graphics parameters).

rot — numeric. The amount of counter-clockwise rotation to add to the labels (in degrees). Note, labels for axes added to the left or right sides are already rotated 90 degrees.

abbr — logical. Should labels be abbreviated? This only works if the data has an abbr column, otherwise the name column will be used regardless of this setting.

center_end_labels — logical. Should labels be centered within the visible range of intervals at the ends of the axis?

skip — A character vector of interval names indicating which intervals should not be labeled. If abbr is TRUE, this can also include interval abbreviations. Quaternary, Holocene, and Late Pleistocene are skipped by default. Set to NULL if this is not desired.

bord_col — character. The border color of the interval boxes.

lty — character. Line type (see lty in graphics parameters).

lwd — numeric. Line width (see lwd in graphics parameters).

bkgd — character. The color of the background color of the scale when no intervals are being shown.

neg — logical. Set this to TRUE if your x-axis is using negative values. If the entire axis is already negative, this will be set to TRUE for you.

exact — logical. Set this to TRUE if you want axis tick marks and numeric tick labels placed at the interval boundaries.

round — integer. Number of decimal places to which exact axis labels should be rounded (using round). If no value is specified, the exact values will be used. Trailing zeros are always removed. tick_at and tick_labels can be used to include labels with trailing zeros.

tick_at	A numeric vector specifying custom points at which tick marks are to be drawn on the axis. If specified, this is passed directly to axis. The default is to compute tick mark locations automatically (see axTicks).
tick_labels	Either a) a logical value specifying whether (numerical) annotations should be made at the tick marks specified by at, or b) a custom character or expression vector of labels to be placed at the tick marks. If at is specified, this argument is passed directly to axis.
phylo	logical. Is the base plot a phylogeny generated by plot.phylo, plotTree, plotSimmap, etc?
root.time	numeric. If phylo is TRUE, this is the time assigned to the root node of the tree. By default, this is taken from the root.time element of the plotted tree.
...	Further arguments that are passed directly to axis.

Details

If a custom data.frame is provided (with intervals), it should consist of at least 3 columns of data. See deeptime::periods for an example.

- The name column (interval_name is also allowed) lists the names of each time interval. These will be used as labels if no abbreviations are provided.
- The max_age column (max_ma is also allowed) lists the oldest boundary of each time interval. Values should always be positive.
- The min_age column (min_ma is also allowed) lists the youngest boundary of each time interval. Values should always be positive.
- The abbr column is optional and lists abbreviations that may be used as labels.
- The color column (colour is also allowed) is also optional and lists a color for the background for each time interval (see the Color Specification section here).
- The lab_color (lab_colour is also allowed) column is also optional and lists a color for the label for each time interval (see the Color Specification section here).

intervals may also be a list if multiple time scales should be added to a single side of the plot. In this case, height, fill, lab, lab_col, lab_size, rot, abbr, center_end_labels, skip, bord_col, lty, and lwd can also be lists. If these lists are not as long as intervals, the elements will be recycled. If individual values (or vectors, e.g. for skip) are used for these parameters, they will be applied to all time scales (and recycled as necessary). If multiple scales are requested they will be added sequentially outwards starting from the plot border. The axis will always be placed on the outside of the last scale.

axis_geo_phylo(...) is shorthand for axis_geo(..., phylo = TRUE).

Value

No return value. Function is used for its side effect, which is to add an axis of the geological timescale to an already existing plot.

Authors

William Gearty & Kilian Eichenseer

Reviewer

Lewis A. Jones

Examples

```
# track user par
oldpar <- par(no.readonly = TRUE)
# single scale on bottom
par(mar = c(6.1, 4.1, 4.1, 2.1)) # modify margin
plot(0:100, axes = FALSE, xlim = c(100, 0), ylim = c(100, 0),
     xlab = NA, ylab = "Depth (m)")
box()
axis(2)
axis_geo(side = 1, intervals = "periods")
# the line argument here depends on the absolute size of the plot
title(xlab = "Time (Ma)", line = 4)

# stack multiple scales
par(mar = c(7.1, 4.1, 4.1, 2.1)) # further expand bottom margin
plot(0:100, axes = FALSE, xlim = c(100, 0), ylim = c(100, 0),
     xlab = NA, ylab = "Depth (m)")
box()
axis(2)
axis_geo(side = 1, intervals = list("epochs", "periods"))
# the line argument here depends on the absolute size of the plot
title(xlab = "Time (Ma)", line = 6)

# scale with MacroStrat intervals
par(mar = c(6.1, 4.1, 4.1, 2.1)) # modify margin
plot(0:30, axes = FALSE, xlim = c(30, 0), ylim = c(30, 0),
     xlab = NA, ylab = "Depth (m)")
box()
axis(2)
axis_geo(side = 1, intervals = "North American land mammal ages")
# the line argument here depends on the absolute size of the plot
title(xlab = "Time (Ma)", line = 4)

# scale with old GTS intervals
par(mar = c(6.1, 4.1, 4.1, 2.1)) # modify margin
plot(0:100, axes = FALSE, xlim = c(100, 0), ylim = c(100, 0),
     xlab = NA, ylab = "Depth (m)")
box()
axis(2)
axis_geo(side = 1, intervals = time_bins(rank = "period"))
# the line argument here depends on the absolute size of the plot
title(xlab = "Time (Ma)", line = 4)

# scale with custom intervals
intervals <- data.frame(min_age = c(0, 10, 25, 32),
                        max_age = c(10, 25, 32, 40),
                        name = c("A", "B", "C", "D"))
par(mar = c(6.1, 4.1, 4.1, 2.1)) # modify margin
```

```
plot(0:40, axes = FALSE, xlim = c(40, 0), ylim = c(40, 0),
     xlab = NA, ylab = "Depth (m)")
box()
axis(2)
axis_geo(side = 1, intervals = intervals)
# the line argument here depends on the absolute size of the plot
title(xlab = "Time (Ma)", line = 4)


# scale with phylogeny
library(phytools)
data(mammal.tree)
plot(mammal.tree)
axis_geo_phylo()
title(xlab = "Time (Ma)", line = 4)


# scale with fossil phylogeny
library(paleotree)
data(RaiaCopesRule)
plot(ceratopsianTreeRaia)
axis_geo_phylo()
title(xlab = "Time (Ma)", line = 4)

# reset user par
par(oldpar)
```

`bin_lat` *Assign fossil occurrences to latitudinal bins*

Description

A function to assign fossil occurrences to user-specified latitudinal bins.

Usage

```
bin_lat(occdf, bins, lat = "lat", boundary = FALSE)
```

Arguments

`occdf`	`dataframe`. A dataframe of the fossil occurrences you wish to bin. This dataframe should contain a column with the latitudinal coordinates of occurrence data.
`bins`	`dataframe`. A dataframe of the bins that you wish to allocate fossil occurrences to, such as that returned by `lat_bins()`. This dataframe must contain at least the following named columns: "bin", "max" and "min".
`lat`	`character`. The name of the column you wish to be treated as the input latitude (e.g. "lat" or "p_lat"). This column should contain numerical values. Defaults to "lat".

boundary — `logical`. If `TRUE`, occurrences falling on the boundaries of latitudinal bins will be duplicated and assigned to both bins. If `FALSE`, occurrences will be binned into the upper bin only (i.e. highest row number).

Value

A dataframe of the original input `occdf` with appended columns containing respective latitudinal bin information.

Developer(s)

Lewis A. Jones

Reviewer(s)

Sofia Galvan

Examples

```
# Load occurrence data
occdf <- tetrapods
# Generate latitudinal bins
bins <- lat_bins(size = 10)
# Bin data
occdf <- bin_lat(occdf = occdf, bins = bins, lat = "lat")
```

---

bin_space *Assign fossil occurrences to spatial bins*

Description

A function to assign fossil occurrences (or localities) to spatial bins/samples using a hexagonal equal-area grid.

Usage

```
bin_space(
  occdf,
  lng = "lng",
  lat = "lat",
  spacing = 100,
  sub_grid = NULL,
  return = FALSE,
  plot = FALSE
)
```

Arguments

occdf	dataframe. A dataframe of the fossil occurrences (or localities) you wish to bin. This dataframe should contain the decimal degree coordinates of your occurrences, and they should be of class numeric.
lng	character. The name of the column you wish to be treated as the input longitude (e.g. "lng" or "p_lng").
lat	character. The name of the column you wish to be treated as the input latitude (e.g. "lat" or "p_lat").
spacing	numeric. The desired spacing between the center of adjacent cells. This value should be provided in kilometres.
sub_grid	numeric. For an optional sub-grid, the desired spacing between the center of adjacent cells in the sub-grid. This value should be provided in kilometres. See details for information on sub-grid usage.
return	logical. Should the equal-area grid information and polygons be returned?
plot	logical. Should the occupied cells of the equal-area grid be plotted?

Details

This function assigns fossil occurrence data into equal-area grid cells using discrete hexagonal grids via the h3jsr package. This package relies on Uber's H3 library, a geospatial indexing system that partitions the world into hexagonal cells. In H3, 16 different resolutions are available (see here). In the implementation of the bin_space() function, the resolution is defined by the user-input spacing which represents the distance between the centroid of adjacent cells. Using this distance, the function identifies which resolution is most similar to the input spacing, and uses this resolution.

Additional functionality allows the user to simultaneously assign occurrence data to equal-area grid cells of a finer-scale grid (i.e. a 'sub-grid') within the primary grid via the sub_grid argument. This might be desirable for users to evaluate the differences in the amount of area occupied by occurrences within their primary grid cells. This functionality also allows the user to easily rarefy across sub-grid cells within primary cells to further standardise spatial sampling (see example for basic implementation).

Note: prior to implementation, coordinate reference system (CRS) for input data is defined as EPSG:4326 (World Geodetic System 1984). The user should transform their data accordingly if this is not appropriate. If you are unfamiliar with working with geographic data, we highly recommend checking out Geocomputation with R.

Value

If the return argument is set to FALSE, a dataframe is returned of the original input occdf with cell information. If return is set to TRUE, a list is returned with both the input occdf and grid information and polygons.

Developer(s)

Lewis A. Jones

Reviewer(s)

Bethany Allen & Kilian Eichenseer

Examples

```
# Get internal data
data("reefs")

# Reduce data for plotting
occdf <- reefs[1:250, ]

# Bin data using a hexagonal equal-area grid
ex1 <- bin_space(occdf = occdf, spacing = 500, plot = TRUE)

# Bin data using a hexagonal equal-area grid and sub-grid
ex2 <- bin_space(occdf = occdf, spacing = 1000, sub_grid = 250, plot = TRUE)

# EXAMPLE: rarefy
# Load data
occdf <- tetrapods[1:250, ]

# Assign to spatial bin
occdf <- bin_space(occdf = occdf, spacing = 1000, sub_grid = 250)

# Get unique bins
bins <- unique(occdf$cell_ID)

# n reps
n <- 10

# Rarefy data across sub-grid grid cells
# Returns a list with each element a bin with respective mean genus richness
df <- lapply(bins, function(x) {
  # subset occdf for respective grid cell
  tmp <- occdf[which(occdf$cell_ID == x), ]

  # Which sub-grid cells are there within this bin?
  sub_bin <- unique(tmp$cell_ID_sub)

  # Sample 1 sub-grid cell n times
  s <- sample(sub_bin, size = n, replace = TRUE)

  # Count the number of unique genera within each sub_grid cell for each rep
  counts <- sapply(s, function(i) {
    # Number of unique genera within each sample
    length(unique(tmp[which(tmp$cell_ID_sub == i), ]$genus))
  })

  # Mean richness across subsamples
  mean(counts)
})
```

bin_time *Assign fossil occurrences to time bins*

Description

A function to assign fossil occurrences to specified time bins based on different approaches commonly applied in palaeobiology.

Usage

```
bin_time(occdf, bins, method = "mid", reps = 100, fun = dunif, ...)
```

Arguments

occdf	dataframe. A dataframe of the fossil occurrences you wish to bin. This dataframe should contain the following named columns: "max_ma" and "min_ma". These columns should contain numeric values. If required, numeric ages can be generated from interval names via the look_up() function.
bins	dataframe. A dataframe of the bins that you wish to allocate fossil occurrences to such as that returned by time_bins(). This dataframe must contain at least the following named columns: "bin", "max_ma" and "min_ma". Columns "max_ma" and "min_ma" must be numeric values.
method	character. The method desired for binning fossil occurrences. Currently, five methods exist in this function: "mid", "majority", "all", "random", and "point". See Details for a description of each.
reps	numeric. A non-negative numeric specifying the number of replications for sampling. This argument is only useful in the case of the "random" or "point" method being specified in the method argument. Defaults to 100.
fun	function. A probability density function from the stats package such as dunif or dnorm. This argument is only useful if the "point" method is specified in the method argument.
...	Additional arguments available in the called function (fun). These arguments may be required for function arguments without default values, or if you wish to overwrite the default argument value (see example). x input values are generated internally based on the age range of the fossil occurrence and should not be manually provided. Note that x input values range between 0 and 1, and function arguments should therefore be scaled to be within these bounds.

Details

Five approaches (methods) exist in the bin_time() function for assigning occurrences to time bins:

- Midpoint: The "mid" method is the simplest approach and uses the midpoint of the fossil occurrence age range to bin the occurrence.

- Majority: The "majority" method bins an occurrence into the bin which it most overlaps with. As part of this implementation, the majority percentage overlap of the occurrence is also calculated and returned as an additional column in `occdf`. If desired, these percentages can be used to further filter an occurrence dataset.
- All: The "all" method bins an occurrence into every bin its age range covers. For occurrences with age ranges of more than one bin, the occurrence row is duplicated. Each occurrence is assigned an ID in the column `occdf$id` so that duplicates can be tracked. Additionally, `occdf$n_bins` records the number of bins each occurrence appears within.
- Random: The "random" method randomly samples X amount of bins (with replacement) from the bins that the fossil occurrence age range covers with equal probability regardless of bin length. The `reps` argument determines the number of times the sample process is repeated. All replications are stored as individual elements within the returned list with an appended `bin_assignment` and `bin_midpoint` column to the original input `occdf`. If desired, users can easily bind this list using `do.call(rbind, x)`.
- Point: The "point" method randomly samples X (`reps`) amount of point age estimates from the age range of the fossil occurrence. Sampling follows a user-input probability density function such as dnorm (see example 5). Users should also provide any additional arguments for the probability density function (see `...`). However, `x` (vector of quantiles) values should not be provided as these values are input from the age range of each occurrence. These values range between 0 and 1, and therefore function arguments should be scaled to be within these bounds. The `reps` argument determines the number of times the sample process is repeated. All replications are stored as individual elements within the returned list with an appended `bin_assignment` and `point_estimates` column to the original input `occdf`. If desired, users can easily bind this list using `do.call(rbind, x)`.

Value

For methods "mid", "majority" and "all", a `dataframe` of the original input `occdf` with the following appended columns is returned: occurrence id (`id`), number of bins that the occurrence age range covers (`n_bins`), bin assignment (`bin_assignment`), and bin midpoint (`bin_midpoint`). In the case of the "majority" method, an additional column of the majority percentage overlap (`overlap_percentage`) is also appended. For the "random" and "point" method, a `list` is returned (of length reps) with each element a copy of the `occdf` and appended columns (random: `bin_assignment` and `bin_midpoint`; point: `bin_assignment` and `point_estimates`).

Developer(s)

Christopher D. Dean & Lewis A. Jones

Reviewer(s)

William Gearty

Examples

```
#Grab internal tetrapod data
occdf <- tetrapods[1:100, ]
bins <- time_bins()
```

```
#Assign via midpoint age of fossil occurrence data
ex1 <- bin_time(occdf = occdf, bins = bins, method = "mid")

#Assign to all bins that age range covers
ex2 <- bin_time(occdf = occdf, bins = bins, method = "all")

#Assign via majority overlap based on fossil occurrence age range
ex3 <- bin_time(occdf = occdf, bins = bins, method = "majority")

#Assign randomly to overlapping bins based on fossil occurrence age range
ex4 <- bin_time(occdf = occdf, bins = bins, method = "random", reps = 5)

#Assign point estimates following a normal distribution
ex5 <- bin_time(occdf = occdf, bins = bins, method = "point", reps = 5,
                fun = dnorm, mean = 0.5, sd = 0.25)
```

`group_apply` *Apply a function over grouping(s) of data*

Description

A function to apply `palaeoverse` functionality across subsets (groups) of data, delineated using one or more variables. Functions which receive a `data.frame` as input (e.g. `nrow`, `ncol`, `lengths`, `unique`) may also be used.

Usage

```
group_apply(occdf, group, fun, ...)
```

Arguments

`occdf`	`dataframe`. A dataframe of fossil occurrences or taxa, as relevant to the desired function. This dataframe must contain the grouping variables and the necessary variables for the function you wish to call (see function-specific documentation for required columns).
`group`	`character`. A vector of column names, specifying the desired subgroups (e.g. "collection_no", "stage_bin"). Supplying more than one grouping variable will produce an output containing subgroups for each unique combination of values.
`fun`	`function`. The function you wish to apply to `occdf`. See details for compatible functions.
`...`	Additional arguments available in the called function. These arguments may be required for function arguments without default values, or if you wish to overwrite the default argument value (see examples).

Details

group_apply applies functions to subgroups of data within a supplied dataset, enabling the separate analysis of occurrences or taxa from different time intervals, spatial regions, or trait values. The function serves as a wrapper around palaeoverse functions. Other functions which can be applied to a data.frame (e.g. nrow, ncol, lengths, unique) may also be used.

All palaeoverse functions which require a dataframe input can be used in conjunction with the group_apply function. However, this is unnecessary for many functions (e.g. bin_time) as groups do not need to be partitioned before binning. This list provides users with palaeoverse functions that might be interesting to apply across group(s):

- tax_unique: return the number of unique taxa per grouping variable.
- tax_range_time: return the temporal range of taxa per grouping variable.
- tax_range_space: return the geographic range of taxa per grouping variable.
- tax_check: return potential spelling variations of the same taxon per grouping variable. Note: verbose needs to be set to FALSE.

Value

A data.frame of the outputs from the selected function, with appended column(s) indicating the user-defined groups. If a single vector is returned via the called function, it will be transformed to a data.frame with the column name equal to the input function.

Developer(s)

Lewis A. Jones & William Gearty

Reviewer(s)

Kilian Eichenseer & Bethany Allen

Examples

```
# Examples
# Get tetrapods data
occdf <- tetrapods[1:100, ]
# Remove NA data
occdf <- subset(occdf, !is.na(genus))
# Count number of occurrences from each country
ex1 <- group_apply(occdf = occdf, group = "cc", fun = nrow)
# Unique genera per collection with group_apply and input arguments
ex2 <- group_apply(occdf = occdf,
                   group = c("collection_no"),
                   fun = tax_unique,
                   genus = "genus",
                   family = "family",
                   order = "order",
                   class = "class",
                   resolution = "genus")
# Use multiple variables (number of occurrences per collection and formation)
```

```
ex3 <- group_apply(occdf = occdf,
                   group = c("collection_no", "formation"),
                   fun = nrow)
# Compute counts of occurrences per latitudinal bin
# Set up lat bins
bins <- lat_bins()
# bin occurrences
occdf <- bin_lat(occdf = occdf, bins = bins)
# Calculate number of occurrences per bin
ex4 <- group_apply(occdf = occdf, group = "lat_bin", fun = nrow)
```

GTS2012 *Geological Timescale 2012*

Description

A dataframe of the Geological Timescale 2012. Age data from the International Commission on Stratigraphy. Supplementary information is also included in the dataset for plotting functionality (e.g. GTS2012 colour scheme).

Usage

```
GTS2012
```

Format

A data frame with 186 rows and 9 variables:

interval_number Index number for the temporal order of all intervals present in the dataset.

interval_name Names of intervals in the dataset.

rank The temporal rank of intervals in the dataset.

max_ma The maximum age of the interval in millions of years before present.

mid_ma The midpoint age of the interval in millions of years before present.

min_ma The minimum age of the interval in millions of years before present.

duration_myr The duration of the interval in millions of years.

font Colour of font to use for plotting in conjunction with the colour column.

colour Colours of stages based on the ICS timescale.

References

Gradstein, F.M., Ogg, J.G., Schmitz, M.D. and Ogg, G.M. eds. (2012). Geologic Timescale 2012. Elsevier.

Source

Compiled by Lewis A. Jones (2022-07-02) from the ICS.

GTS2020 *Geological Timescale 2020*

Description

A dataframe of the Geological Timescale 2020. Age data from the International Commission on Stratigraphy. Supplementary information is included in the dataset for plotting functionality (e.g. GTS2020 colour scheme).

Usage

```
GTS2020
```

Format

A data frame with 189 rows and 9 variables:

interval_number Index number for the temporal order of all intervals present in the dataset.

interval_name Names of intervals in the dataset.

rank The temporal rank of intervals in the dataset.

max_ma The maximum age of the interval in millions of years before present.

mid_ma The midpoint age of the interval in millions of years before present.

min_ma The minimum age of the interval in millions of years before present.

duration_myr The duration of the interval in millions of years.

font Colour of font to use for plotting in conjunction with the colour column.

colour Colours of stages based on the ICS timescale.

References

Gradstein, F.M., Ogg, J.G., Schmitz, M.D. and Ogg, G.M. eds. (2020). Geologic Timescale 2020. Elsevier.

Source

Compiled by Lewis A. Jones (2022-07-02) from the ICS.

interval_key *Example dataset: Interval key for the look_up function*

Description

A table of geological intervals and the earliest and latest corresponding international geological stages from the International Commission on Stratigraphy (ICS). The table was compiled using regional stratigraphies, the GeoWhen Database, temporal information from the Paleobiology Database and the Geological Timescale 2022. Some assignments were made with incomplete information on the stratigraphic provenance of intervals. The assignments in this table should be verified before research use. They are provided here as an example of functionality only.

Usage

```
interval_key
```

Format

A data frame with 1323 rows and 3 variables:

interval_name Stratigraphic interval

early_stage Earliest (oldest) geological stage which overlaps with the interval

late_stage Latest (youngest) geological stage which overlaps with the interval

Source

Compiled by Kilian Eichenseer and Lewis Jones for assigning geological stages to ccurrences from the Paleobiology Database and the PaleoReefs Database.

lat_bins *Generate latitudinal bins*

Description

A function to generate latitudinal bins of a given size for a user-defined latitudinal range. If the desired size of the bins is not compatible with the defined latitudinal range, bin size can be updated to the nearest integer which is divisible into this range.

Usage

```
lat_bins(size = 10, max = 90, min = -90, fit = FALSE, plot = FALSE)
```

Arguments

size	numeric. A single numeric value defining the width of the latitudinal bins. This value must be more than 0, and less than or equal to 90 (defaults to 10).
max	numeric. A single numeric value defining the upper limit of the latitudinal range (defaults to 90).
min	numeric. A single numeric value defining the lower limit of the latitudinal range (defaults to -90).
fit	logical. Should bin size be checked to ensure that the entire latitudinal range is covered? If fit = TRUE, bin size is set to the nearest integer which is divisible by the user-input range. If fit = FALSE, and bin size is not divisible into the range, the upper part of the latitudinal range will be missing.
plot	logical. Should a plot of the latitudinal bins be generated?

Value

A dataframe of latitudinal bins of user-defined size.

Developer(s)

Lewis A. Jones

Reviewer(s)

Bethany Allen

Examples

```
# Generate 20 degrees latitudinal bins
bins <- lat_bins(size = 20)

# Generate latitudinal bins with closest fit to 13 degrees
bins <- lat_bins(size = 13, fit = TRUE)

# Generate latitudinal bins for defined latitudinal range
bins <- lat_bins(size = 10, max = 50, min = -50)
```

---

look_up *Look up geological intervals and assign geological stages*

Description

A function that uses interval names to assign either international geological stages and numeric ages from the International Commission on Stratigraphy (ICS), or user-defined intervals, to fossil occurrences.

Usage

```
look_up(
  occdf,
  early_interval = "early_interval",
  late_interval = "late_interval",
  int_key = FALSE,
  assign_with_GTS = "GTS2020",
  return_unassigned = FALSE
)
```

Arguments

occdf	dataframe. A dataframe of fossil occurrences or other geological data, with columns of class character specifying the earliest and the latest possible interval associated with each occurrence.
early_interval	character. Name of the column in occdf that contains the earliest interval from which the occurrences are from. Defaults to "early_interval".
late_interval	character. Name of the column in occdf that contains the latest interval from which the occurrences are from. Defaults to "late_interval".
int_key	dataframe. A dataframe linking interval names to international geological stage names from the ICS, or other, user-defined intervals. This dataframe should contain the following named columns containing character values: • interval_name contains the names to be matched from occdf • early_stage contains the names of the earliest stages corresponding to the intervals • late_stage contains the latest stage corresponding to the intervals Optionally, named numeric columns provide maximum and minimum ages for the intervals: • max_ma • min_ma If set to FALSE (default), stages and numerical ages can be assigned based on one of the GTS tables (see below).
assign_with_GTS	character or FALSE. Allows intervals to be searched in the GTS2020 (default) or the GTS2012 table. Set to FALSE to disable.
return_unassigned	logical. Return interval names which could not be assigned, instead of the dataframe with assignments. Defaults to FALSE.

Details

If int_key is set to FALSE (default), this function can be used to assign numerical ages solely based on stages from a GTS table, and to assign stages based on GTS interval names.

Instead of geological stages, the user can supply any names in the early_stage and late_stage column of int_key. assign_with_GTS should then be set to FALSE.

An exemplary int_key has been included within this package (interval_key). This key works well for assigning geological stages to many of the intervals from the Paleobiology Database and the PaleoReefs Database. palaeoverse cannot guarantee that all of the stage assignments with the exemplary key are accurate. The table corresponding to this key can be loaded with palaeoverse::interval_key.

Value

A dataframe of the original input data with the following appended columns is returned: early_stage and late_stage, corresponding to the earliest and latest international geological stage which could be assigned to the occurrences based on the given interval names. interval_max_ma and interval_min_ma return maximum and minimum interval ages if provided in the interval key, or if they can be fetched from GTS2012 or GTS2020. A column interval_mid_ma is appended to provide the midpoint ages of the intervals.

Developer(s)

Kilian Eichenseer & William Gearty

Reviewer(s)

Lewis A. Jones & Christopher D. Dean

Examples

```
## Just use GTS2020 (default):
# create exemplary dataframe
taxdf <- data.frame(name = c("A", "B", "C"),
early_interval = c("Maastrichtian", "Campanian", "Sinemurian"),
late_interval = c("Maastrichtian", "Campanian", "Bartonian"))
# assign stages and numerical ages
taxdf <- look_up(taxdf)

## Use exemplary int_key
# Get internal reef data
occdf <- reefs
 # assign stages and numerical ages
occdf <- look_up(occdf,
                 early_interval = "interval",
                 late_interval = "interval",
                 int_key = interval_key)

## Use exemplary int_key and return unassigned
# Get internal tetrapod data
occdf <- tetrapods
```

```
# assign stages and numerical ages
occdf <- look_up(occdf, int_key = palaeoverse::interval_key)
# return unassigned intervals
unassigned <- look_up(occdf, int_key = palaeoverse::interval_key,
                      return_unassigned = TRUE)

## Use own key and GTS2012:
# create example data
occdf <- data.frame(
  stage = c("any Permian", "first Permian stage",
            "any Permian", "Roadian"))
# create example key
interval_key <- data.frame(
  interval_name = c("any Permian", "first Permian stage"),
  early_stage = c("Asselian", "Asselian"),
  late_stage = c("Changhsingian", "Asselian"))
# assign stages and numerical ages:
occdf <- look_up(occdf,
                 early_interval = "stage", late_interval = "stage",
                 int_key = interval_key, assign_with_GTS = "GTS2012")
```

`palaeorotate` *Palaeorotate fossil occurrences*

Description

A function to generate palaeocoordinates for fossil occurrence data (i.e. reconstruct the geographic distribution of organisms' remains at time of deposition). Each occurrence is assigned palaeocoordinates based on its current geographic position and age estimate.

Usage

```
palaeorotate(
  occdf,
  lng = "lng",
  lat = "lat",
  age = "age",
  model = "MERDITH2021",
  method = "point",
  uncertainty = TRUE,
  round = 3
)
```

Arguments

`occdf`	`dataframe`. Fossil occurrences to be palaeogeographically reconstructed. `occdf` should contain columns with longitudinal and latitudinal values, as well as age estimates. The age of rotation should be supplied in millions of years before present.

`lng`	`character`. The name of the column you wish to be treated as longitude (defaults to "lng").
`lat`	`character`. The name of the column you wish to be treated as latitude (defaults to "lat").
`age`	`character`. The name of the column you wish to be treated as the age for rotation (defaults to "age").
`model`	`character`. The name(s) of the plate rotation model(s) to be used to reconstruct palaeocoordinates. See details for available models.
`method`	`character`. Method used to calculate palaeocoordinates for fossil occurrences. Either "grid" to use reconstruction files, or "point" (default) to use the GPlates API service. See details section for specific details.
`uncertainty`	`logical`. Should the uncertainty in palaeogeographic reconstructions be returned? If set to TRUE (default), the palaeolatitudinal range and maximum geographic distance (in km) between output palaeocoordinates are calculated. This argument is only relevant if more than one plate rotation model is specified in `model`.
`round`	`numeric`. Numeric value indicating the number of decimal places `lng`, `lat` and `age` should be rounded to. This functionality is only relevant for the "point" `method`. Rounding can speed up palaeorotation by reducing the number of unique coordinate pairs. Defaults to a value of 3. If no rounding is desired, set this value to `NULL`.

Details

This function can generate palaeocoordinates using two different approaches (`method`):

- Reconstruction files: The "grid" `method` uses reconstruction files to spatiotemporally link present-day geographic coordinates and age estimates with an equal-area hexagonal grid (spacing = 100 km) rotated to the midpoint of Phanerozoic (0–540 Ma) stratigraphic stages (Geological Time Scale, 2020). The grid was generated using the `h3jsr` R package and 'h3_resolution' 3 (see `h3_info_table`). If specific ages of rotation are required, or fine-scale spatial analyses are being conducted, use of the "point" `method` (see GPlates API below) is recommended (particularly if occurrences are close to plate boundaries). As implemented, when using the "grid" `method`, coordinates within the same grid cell will be assigned equivalent palaeocoordinates due to spatial aggregation. However, the reconstruction files provide pre-generated palaeocoordinates enabling efficient estimation of the past distribution of fossil occurrences. The reconstruction files along with additional documentation are deposited on Zenodo. Note: each reconstruction file is 5–10 MB in size.
- GPlates API: The "point" `method` uses the GPlates Web Service to reconstruct palaeocoordinates for point data. The use of this `method` is slower than the "grid" `method` if many unique time intervals exist in your dataset. However, it provides palaeocoordinates with higher precision.

Available models and timespan for each `method`:

- "MERDITH2021" (Merdith et al., 2021)
  - 0–540 Ma (grid)

  - 0–1000 Ma (point)
- "MULLER2016" (Müller et al., 2016)
  - 0–230 Ma (grid/point)
- "MATTHEWS2016_pmag_ref" (Matthews et al., 2016)
  - 0–410 Ma (grid/point)
- "SETON2012" (Seton et al., 2012)
  - 0–200 Ma (grid/point)
- "GOLONKA" (Wright et al., 2013)
  - 0–540 Ma (grid/point)
- "PALEOMAP" (Scotese & Wright, 2018)
  - 0–540 Ma (grid)
  - 0–750 Ma (point)

Access is also provided for the following mantle reference frame models. However, they are generally not recommended for reconstructing palaeocoordinates.

- "MULLER2022" (Müller et al., 2022)
  - 0–540 Ma (grid)
  - 0–1000 Ma (point)
- "MULLER2019" (Müller et al., 2019)
  - 0–250 Ma (grid/point)
- "MATTHEWS2016_mantle_ref" (Matthews et al., 2016)
  - 0–410 Ma (grid/point)

Value

A `dataframe` containing the original input occurrence dataframe and the reconstructed coordinates (i.e. "p_lng", "p_lat"). The "grid" `method` also returns the age of rotation ("rot_age") and the reference coordinates rotated ("rot_lng" and "rot_lat"). If only one model is requested, a column containing the rotation model used ("rot_model") is also appended. Otherwise, the name of each model is appended to the name of each column containing palaeocoordinates (e.g. "p_lng_GOLONKA"). If `uncertainty` is set to `TRUE`, the palaeolatitudinal range ("range_p_lat") and the maximum geographic distance ("max_dist") in km between palaeocoordinates will also be returned (the latter calculated via `distGeo`).

References

- Matthews, K.J., Maloney, K.T., Zahirovic, S., Williams, S.E., Seton, M., and Müller, R.D. (2016). Global plate boundary evolution and kinematics since the late Paleozoic. Global and Planetary Change, 146, 226-250. doi:10.1016/j.gloplacha.2016.10.002.
- Merdith, A., Williams, S.E., Collins, A.S., Tetley, M.G., Mulder, J.A., Blades, M.L., Young, A., Armistead, S.E., Cannon, J., Zahirovic, S., Müller. R.D. (2021). Extending full-plate tectonic models into deep time: Linking the Neoproterozoic and the Phanerozoic. Earth-Science Reviews, 214(103477). doi:10.1016/j.earscirev.2020.103477.

- Müller, R. D., Flament, N., Cannon, J., Tetley, M. G., Williams, S. E., Cao, X., Bodur, Ö. F., Zahirovic, S., and Merdith, A. (2022). A tectonic-rules-based mantle reference frame since 1 billion years ago – implications for supercontinent cycles and plate–mantle system evolution, Solid Earth, 13, 1127–1159. doi:10.5194/se1311272022.
- Müller, R. D., Zahirovic, S., Williams, S. E., Cannon, J., Seton, M., Bower, D. J., Tetley, M. G., Heine, C., Le Breton, E., Liu, S., Russell, S. H. J., Yang, T., Leonard, J., and Gurnis, M. (2019). A global plate model including lithospheric deformation along major rifts and orogens since the Triassic. Tectonics, 38(6) 1884-1907. doi:10.1029/2018TC005462.
- Müller R.D., Seton, M., Zahirovic, S., Williams, S.E., Matthews, K.J., Wright, N.M., Shephard, G.E., Maloney, K.T., Barnett-Moore, N., Hosseinpour, M., Bower, D.J., Cannon, J. (2016). Ocean basin evolution and global-scale plate reorganization events since Pangea breakup. Annual Review of Earth and Planetary Sciences 44(1), 107-138. doi:10.1146/annurevearth060115012211.
- Scotese, C., & Wright, N. M. (2018). PALEOMAP Paleodigital Elevation Models (PaleoDEMs) for the Phanerozoic. PALEOMAP Project.
- Seton, M., Müller, R.D., Zahirovic, S., Gaina, C., Torsvik, T.H., Shephard, G., Talsma, A., Gurnis, M., Turner, M., Maus, S., Chandler, M. (2012). Global continental and ocean basin reconstructions since 200 Ma. Earth-Science Reviews, 113(3-4), 212-270. doi:10.1016/j.earscirev.2012.03.002.
- Wright, N., Zahirovic, S., Müller, R. D., & Seton, M. (2013). Towards community-driven paleogeographic reconstructions: integrating open-access paleogeographic and paleobiology data with plate tectonics. Biogeosciences, 10(3), 1529-1541. doi:10.5194/bg1015292013.

See GPlates documentation for additional information and details.

Developer(s)

Lewis A. Jones

Reviewer(s)

Kilian Eichenseer, Lucas Buffan & Will Gearty

Examples

```
## Not run:
#Generic example with a few occurrences
occdf <- data.frame(lng = c(2, -103, -66),
                lat = c(46, 35, -7),
                age = c(88, 125, 200))

#Calculate palaeocoordinates using reconstruction files
ex1 <- palaeorotate(occdf = occdf, method = "grid")

#Calculate palaeocoordinates using the GPlates API
ex2 <- palaeorotate(occdf = occdf, method = "point")

#Calculate uncertainity in palaeocoordinates from models
ex3 <- palaeorotate(occdf = occdf,
                    method = "grid",
```

```
                    model = c("MERDITH2021",
                              "GOLONKA",
                              "PALEOMAP"),
                    uncertainty = TRUE)

#Now with some real fossil occurrence data!

#Grab some data from the Paleobiology Database
data(tetrapods)

#Assign midpoint age of fossil occurrence data for reconstruction
tetrapods$age <- (tetrapods$max_ma + tetrapods$min_ma)/2

#Rotate the data
ex3 <- palaeorotate(occdf = tetrapods)

#Calculate uncertainity in palaeocoordinates from models
ex4 <- palaeorotate(occdf = tetrapods,
                    model = c("MERDITH2021",
                              "GOLONKA",
                              "PALEOMAP",
                              "SETON2012"),
                    uncertainty = TRUE)

## End(Not run)
```

---

phylo_check *Check phylogeny tip names*

Description

A function to check the list of tip names in a phylogeny against a vector of taxon names, and if desired, to trim the phylogeny to only include taxon names within the vector.

Usage

```
phylo_check(tree = NULL, list = NULL, out = "full_table", sort = "presence")
```

Arguments

tree	phylo. A phylo object containing the phylogeny.
list	character. A vector of taxon names. Binomials can be separated with either a space or an underscore. The names should not contain any other punctuation.
out	character. Determine whether to return either a dataframe describing which taxa are included or not included in the tree ("full_table", the default), the same table but with taxa included in both the tree and the list removed ("diff_table"), the counts of taxa included and not included in the tree ("counts"), or the phylogeny trimmed to only include taxa in the provided list ("tree").
sort	character. If out = "full_table" or out = "diff_table", sort the names by presence in the tree ("presence", the default), or alphabetically ("az").

Details

Phylogenies can be read into R from .txt or .tree files containing the Newick formatted tree using `ape::read.tree()`, and can be saved as files using `ape::write.tree()`. When out = "tree", tips are trimmed using `ape::drop.tip()`; if your tree is not ultrametric (i.e. the tip dates are not all the same), we recommend using `paleotree::fixRootTime()` to readjust your branch lengths following pruning.

Value

If out = "full_table", a `dataframe` describing whether taxon names are present in the list and/or the tree. If out = "diff_table", a `dataframe` describing which taxon names are present in the list or the tree, but not both. If out = "counts", a summary table containing the number of taxa in the list but not the tree, in the tree but not the list, and in both. If out = "tree", a phylo object consisting of the input phylogeny trimmed to only include the tips present in the list.

Developer(s)

Bethany Allen

Reviewer(s)

William Gearty & Pedro Godoy

Examples

```
# track user par
oldpar <- par(no.readonly = TRUE)
#Read in example tree of ceratopsians from paleotree
library(paleotree)
data(RaiaCopesRule)
#Set smaller margins for plotting
par(mar = rep(0.5, 4))
plot(ceratopsianTreeRaia)

#Specify list of names
dinosaurs <- c("Nasutoceratops_titusi", "Diabloceratops_eatoni",
"Zuniceratops_christopheri", "Psittacosaurus_major",
"Psittacosaurus_sinensis", "Avaceratops_lammersi",
"Xenoceratops_foremostensis", "Leptoceratops_gracilis",
"Triceratops_horridus", "Triceratops_prorsus")

#Table of taxon names in list, tree or both
ex1 <- phylo_check(tree = ceratopsianTreeRaia, list = dinosaurs)

#Counts of taxa in list, tree or both
ex2 <- phylo_check(tree = ceratopsianTreeRaia, list = dinosaurs,
                   out = "counts")

#Trim tree to tips in the list
my_ceratopsians <- phylo_check(tree = ceratopsianTreeRaia, list = dinosaurs,
out = "tree")
```

```
plot(my_ceratopsians)
# reset user par
par(oldpar)
```

`reefs` *Example dataset: Phanerozoic reefs from the PaleoReefs Database*

Description

A dataset of Phanerozoic reef occurrences from the PaleoReefs Database (PARED). This example dataset includes a subset of the available data from PARED, but can be used to demonstrate how the functions in the `palaeoverse` package might be applied.

Usage

```
reefs
```

Format

A data frame with 4363 rows and 14 variables:

r_number Reference number given to the particular fossil reef in PARED

name Reference name given to the particular fossil reef in PARED

formation The geological formation to which the fossil reef belongs

system The stratigraphic system to which the fossil reef belongs

series The stratigraphic series to which the fossil reef belongs

interval The stratigraphic interval to which the fossil reef belongs

biota_main The main biota present within the fossil reef

biota_sec The secondary biota present within the fossil reef

lng The modern-day longitude of the fossil reef

lat The modern-day latitude of the fossil reef

country The country or ocean the fossil reef is located in

authors The authors of the publication documenting the fossil reef

title The title of the publication documenting the fossil reef

year The year of the publication documenting the fossil reef

References

Kiessling, W. & Krause, M. C. (2022). PaleoReefs Database (PARED) (1.0) Data set. doi:10.5281/zenodo.6037852

Source

Compiled by Lewis A. Jones. Downloaded on the 25th July 2022. doi:10.5281/zenodo.6037852

tax_check *Taxonomic spell check*

Description

A function to check for and count potential spelling variations of the same taxon. Spelling variations are checked within alphabetical groups (default), or within higher taxonomic groups if provided.

Usage

```
tax_check(
  taxdf,
  name = "genus",
  group = NULL,
  dis = 0.05,
  start = 1,
  verbose = TRUE
)
```

Arguments

taxdf	dataframe. A dataframe with named columns containing taxon names (e.g. "species", "genus"). An optional column containing the groups (e.g. "family", "order") which taxon names belong to may also be provided (see group for details). NA values or empty strings in the name and group columns (i.e. "" and " ") are ignored.
name	character. The column name of the taxon names you wish to check (e.g. "genus").
group	character. The column name of the higher taxonomic assignments in taxdf you wish to group by. If NULL (default), name comparison will be conducted within alphabetical groups.
dis	numeric. The dissimilarity threshold: a value greater than 0 (completely dissimilar), and less than 1 (completely similar). Potential synonyms above this threshold are not returned. This value is set to 0.05 by default, but the user might wish to experiment with this value for their specific data.
start	numeric. The number of shared characters at the beginning of potential synonyms that should match. Potential synonyms below this value will not be returned. By default this value is set to 1 (i.e. the first letter of synonyms must match).
verbose	logical. Should the results of the non-letter character check be reported to the user? If TRUE, the result will only be reported if such characters are detected in the taxon names.

Details

When higher taxonomy is provided, but some entries are missing, comparisons will still be made within alphabetical groups of taxa which lack higher taxonomic affiliations. The function also performs a check for non-letter characters which are not expected to be present in correctly-formatted taxon names. This detection may be made available to the user via the `verbose` argument. Comparisons are performed using the Jaro dissimilarity metric via `stringdist::stringdistmatrix()`.

As all string distance metrics rely on approximate string matching, different metrics can produce different results. This function uses Jaro distance as it was designed with short, typed strings in mind, but good practice should include comparisons using multiple metrics, and ultimately specific taxonomic vetting where possible. A more complete implementation and workflow for cleaning taxonomic occurrence data is available in the `fossilbrush` R package on CRAN.

Value

If verbose = `TRUE` (default), a `list` with three elements. The first element in the list (synonyms) is a `data.frame` with each row reporting a pair of potential synonyms. The first column "group" contains the higher group in which they occur (alphabetical groupings if `group` is not provided). The second column "greater" contains the most common synonym in each pair. The third column "lesser" contains the least common synonym in each pair. The third and fourth column (`count_greater`, `count_lesser`) contain the respective counts of each synonym in a pair. If no matches were found for the filtering arguments, this element is `NULL` instead. The second element (`non_letter_name`) is a vector of taxon names which contain non-letter characters, or `NULL` if none were detected. The third element (non_letter_group) is a vector of taxon groups which contain non-letter characters, or `NULL` if none were detected. If verbose = `FALSE`, a `data.frame` as described above is returned, or `NULL` if no matches were found.

Reference

van der Loo, M. P. J. (2014). The stringdist package for approximate string matching. The R Journal 6, 111-122.

Developer(s)

Joseph T. Flannery-Sutherland & Lewis A. Jones

Reviewer(s)

Lewis A. Jones, Kilian Eichenseer & Christopher D. Dean

Examples

```
# load occurrence data
data("tetrapods")
# Check taxon names alphabetically
ex1 <- tax_check(taxdf = tetrapods, name = "genus", dis = 0.1)
# Check taxon names by group
ex2 <- tax_check(taxdf = tetrapods, name = "genus",
                 group = "family", dis = 0.1)
```

`tax_expand_lat` *Generate pseudo-occurrences from latitudinal range data*

Description

A function to generate pseudo-occurrences for taxa based on latitudinal ranges (e.g. the output of the 'lat' method in `tax_range_space`). While the resulting pseudo-occurrences should not be treated as equivalent to actual occurrence data (e.g. like that from the Paleobiology Database), such pseudo-occurrences may be useful for performing statistical analyses where the row representing a taxon must be replicated for each latitudinal bin through which the taxon ranges.

Usage

```
tax_expand_lat(taxdf, bins, max_lat = "max_lat", min_lat = "min_lat")
```

Arguments

`taxdf`	`dataframe`. A dataframe of taxa (such as the output of the 'lat' method in `tax_range_space`) with columns containing latitudinal range data (maximum and minimum latitude). Column names are assumed to be "max_lat" and "min_lat", but may be updated via the `max_lat` and `min_lat` arguments. Each row should represent a unique taxon. Additional columns may be included (e.g. taxon names, additional taxonomy, etc) and will be included in the returned `data.frame`.
`bins`	`dataframe`. A dataframe of the bins that you wish to allocate fossil occurrences to, such as that returned by `lat_bins`. This dataframe must contain at least the following named columns: "bin", "max" and "min".
`max_lat`	`character`. The name of the column you wish to be treated as the maximum latitude of the latitudinal range (e.g. "max_lat").
`min_lat`	`character`. The name of the column you wish to be treated as the minimum latitude of the latitudinal range (e.g. "min_lat").

Value

A `dataframe` where each row represents a latitudinal bin which a taxon ranges through. The columns are identical to those in the user-supplied data with additional columns included to identify bins. Output will be returned in the order of supplied bins.

Developer(s)

Lewis A. Jones & William Gearty

Reviewer(s)

Christopher D. Dean

Examples

```
bins <- lat_bins()
taxdf <- data.frame(name = c("A", "B", "C"),
                    max_lat = c(60, 20, -10),
                    min_lat = c(20, -40, -60))
ex <- tax_expand_lat(taxdf = taxdf,
                     bins = bins,
                     max_lat = "max_lat",
                     min_lat = "min_lat")
```

`tax_expand_time` *Generate pseudo-occurrences from temporal range data*

Description

A function to generate interval-level pseudo-occurrences for taxa based on temporal ranges (e.g. the output of `tax_range_time`). While the resulting pseudo-occurrences should not be treated as equivalent to actual occurrence data (e.g. like that from the Paleobiology Database), such pseudo-occurrences may be useful for performing statistical analyses where the row representing a taxon must be replicated for each interval through which the taxon persisted.

Usage

```
tax_expand_time(
  taxdf,
  max_ma = "max_ma",
  min_ma = "min_ma",
  scale = "GTS2020",
  rank = "stage",
  ext_orig = TRUE
)
```

Arguments

`taxdf`	`dataframe`. A dataframe of taxa (such as that produced by `tax_range_time`) with columns for the maximum and minimum ages (FADs and LADs). Each row should represent a unique taxon. Additional columns may be included (e.g. taxon names, additional taxonomy, etc) and will be included in the returned `data.frame`. If required, `numeric` ages can be generated from interval names via the `look_up` function.
`max_ma`	`character`. The name of the column you wish to be treated as the maximum limit (FADs) of the age range (e.g. "max_ma").
`min_ma`	`character`. The name of the column you wish to be treated as the minimum limit (LADs) of the age range (e.g. "min_ma").
`scale`	`character`. Specify the desired geological timescale to be used, either "GTS2020" or "GTS2012".

rank	character. Specify the desired stratigraphic rank. Choose from: "stage", "epoch", "period", "era", and "eon".
ext_orig	logical. Should two additional columns be added to identify the intervals in which taxa originated and went extinct?

Value

A dataframe where each row represents an interval during which a taxon in the original user-supplied data persisted. The columns are identical to those in the user-supplied data with additional columns included to identify the intervals. If ext_orig is TRUE, two additional columns are added to identify in which intervals taxa originated and went extinct.

Developer(s)

William Gearty & Lewis A. Jones

Reviewer(s)

Lewis A. Jones

Examples

```
taxdf <- data.frame(name = c("A", "B", "C"),
                    max_ma = c(150, 60, 30),
                    min_ma = c(110, 20, 0))
ex <- tax_expand_time(taxdf)
```

---

tax_range_space *Calculate the geographic range of fossil taxa*

Description

A function to calculate the geographic range of fossil taxa from occurrence data. The function can calculate geographic range in four ways: convex hull, latitudinal range, maximum Great Circle Distance, and the number of occupied equal-area hexagonal grid cells.

Usage

```
tax_range_space(
  occdf,
  name = "genus",
  lng = "lng",
  lat = "lat",
  method = "lat",
  spacing = 100,
  coords = FALSE
)
```

Arguments

occdf	dataframe. A dataframe of fossil occurrences. This dataframe should contain at least three columns: names of taxa, longitude and latitude (see name, lng, and lat arguments).
name	character. The name of the column you wish to be treated as the input names (e.g. "species" or "genus"). NA data should be removed prior to function call.
lng	character. The name of the column you wish to be treated as the input longitude (e.g. "lng" or "p_lng"). NA data should be removed prior to function call.
lat	character. The name of the column you wish to be treated as the input latitude (e.g. "lat" or "p_lat"). NA data should be removed prior to function call.
method	character. How should geographic range be calculated for each taxon in occdf? Four options exist in this function: "con", "lat", "gcd", and "occ". See Details for a description of each.
spacing	numeric. The desired spacing (in km) between the center of adjacent grid cells. Only required if the method argument is set to "occ". The default is 100.
coords	logical. Should the output coordinates be returned for the "con" and "gcd" method?

Details

Four commonly applied approaches (Darroch et al. 2020) are available using the tax_range_space function for calculating ranges:

- Convex hull: the "con" method calculates the geographic range of taxa using a convex hull for each taxon in occdf, and calculates the area of the convex hull (in km^2) using geosphere::areaPolygon(). The convex hull method works by creating a polygon that encompasses all occurrence points of the taxon.
- Latitudinal: the "lat" method calculates the palaeolatitudinal range of a taxon. It does so for each taxon in occdf by finding their maximum and minimum latitudinal occurrence (from input lat). The palaeolatitudinal range of each taxon is also calculated (i.e. the difference between the minimum and maximum latitude).
- Maximum Great Circle Distance: the "gcd" method calculates the maximum Great Circle Distance between occurrences for each taxon in occdf. It does so using geosphere::distHaversine(). This function calculates Great Circle Distance using the Haversine method with the radius of the Earth set to the 6378.137 km. Great Circle Distance represents the shortest distance between two points on the surface of a sphere. This is different from Euclidean Distance, which represents the distance between two points on a plane.
- Occupied cells: the "occ" method calculates the number and proportion of occupied equal-area grid cells. It does so using discrete hexagonal grids via the h3jsr package. This package relies on Uber's H3 library, a geospatial indexing system that partitions the world into hexagonal cells. In H3, 16 different resolutions are available (see here). In the implementation of the tax_range_space() function, the resolution is defined by the user-input spacing which represents the distance between the centroid of adjacent cells. Using this distance, the function identifies which resolution is most similar to the input spacing, and uses this resolution.

Value

A dataframe with method-specific columns:

- For the "con" method, a dataframe with each unique taxa (taxon) and taxon ID (taxon_id) by convex hull coordinate (lng & lat) combination, and area (area) in km^2 is returned.
- For the "lat" method, a dataframe with unique taxa (taxon), taxon ID (taxon_id), maximum latitude of occurrence (max_lat), minimum latitude of occurrence (min_lat), and latitudinal range (range_lat) is returned.
- For the "gcd" method, a dataframe with each unique taxa (taxon) and taxon ID (taxon_id) by coordinate combination (lng & lat) of the two most distant points, and the Great Circle Distance (gcd) between these points in km is returned.
- For the "occ" method, a dataframe with unique taxa (taxon), taxon ID (taxon_id), the number of occupied cells (n_cells), proportion of occupied cells from all occupied by occurrences (proportional_occ), and the spacing between cells (spacing) in km is returned. Note: the number of occupied cells and proportion of occupied cells is highly dependent on the user-defined spacing. For the "con", "lat" and "gcd" method, values of zero indicate that the respective taxon is a singleton (i.e. represented by only one occurrence).

Reference(s)

Darroch, S. A., Casey, M. M., Antell, G. S., Sweeney, A., & Saupe, E. E. (2020). High preservation potential of paleogeographic range size distributions in deep time. The American Naturalist, 196(4), 454-471.

Developer(s)

Lewis A. Jones

Reviewer(s)

Bethany Allen & Christopher D. Dean

Examples

```
# Grab internal data
occdf <- tetrapods[1:100, ]
# Remove NAs
occdf <- subset(occdf, !is.na(genus))
# Convex hull
ex1 <- tax_range_space(occdf = occdf, name = "genus", method = "con")
# Latitudinal range
ex2 <- tax_range_space(occdf = occdf, name = "genus", method = "lat")
# Great Circle Distance
ex3 <- tax_range_space(occdf = occdf, name = "genus", method = "gcd")
# Occupied grid cells
ex4 <- tax_range_space(occdf = occdf, name = "genus",
                       method = "occ", spacing = 500)
# Convex hull with coordinates
ex5 <- tax_range_space(occdf = occdf, name = "genus", method = "con",
coords = TRUE)
```

tax_range_time *Calculate the temporal range of fossil taxa*

Description

A function to calculate the temporal range of fossil taxa from occurrence data.

Usage

```
tax_range_time(
  occdf,
  name = "genus",
  min_ma = "min_ma",
  max_ma = "max_ma",
  by = "FAD",
  plot = FALSE
)
```

Arguments

occdf	dataframe. A dataframe of fossil occurrences containing at least three columns: names of taxa, maximum age and minimum age (see name, lng, and lat arguments). These ages should constrain the age range of the fossil occurrence and are assumed to be in millions of years before present.
name	character. The name of the column you wish to be treated as the input names, e.g. "genus" (default).
min_ma	character. The name of the column you wish to be treated as the minimum limit of the age range, e.g. "min_ma" (default).
max_ma	character. The name of the column you wish to be treated as the maximum limit of the age range, e.g. "max_ma" (default).
by	character. How should the output be sorted? Either: "FAD" (first-appearance date; default), "LAD" (last-appearance data), or "name" (alphabetically by taxon names).
plot	logical. Should a plot of the ranges be generated?

Details

The temporal range(s) of taxa are calculated by extracting all unique taxa (name column) from the input occdf, and checking their first and last appearance. The temporal duration of each taxon is also calculated. A plot of the temporal range of each taxon is also returned if plot = TRUE. If the input data columns contain NAs, these should be removed prior to function call.

Note: this function provides output based solely on the user input data. The true duration of a taxon is likely confounded by uncertainty in dating occurrences, and incomplete sampling and preservation.

Value

A dataframe containing the following columns: unique taxa (taxon), taxon ID (taxon_id), first appearance of taxon (max_ma), last appearance of taxon (min_ma), duration of temporal range (range_myr), and number of occurrences per taxon (n_occ) is returned.

Developer(s)

Lewis A. Jones

Reviewer(s)

Bethany Allen & Christopher D. Dean

Examples

```
# Grab internal data
occdf <- tetrapods
# Remove NAs
occdf <- subset(occdf, !is.na(order) & order != "NO_ORDER_SPECIFIED")
# Temporal range
ex <- tax_range_time(occdf = occdf, name = "order", plot = TRUE)
```

---

tax_unique *Filter occurrences to unique taxa*

Description

A function to filter a list of taxonomic occurrences to unique taxa of a predefined resolution. Occurrences identified to a coarser taxonomic resolution than the desired level are retained if they belong to a clade which is not otherwise represented in the dataset (see details section for further information). This has previously been described as "cryptic diversity" (e.g. Mannion et al. 2011).

Usage

```
tax_unique(
  occdf = NULL,
  binomial = NULL,
  species = NULL,
  genus = NULL,
  ...,
  name = NULL,
  resolution = "species",
  append = FALSE
)
```

Arguments

occdf	dataframe. A dataframe containing information on the occurrences or taxa to filter.
binomial	character. The name of the column in occdf containing the genus and species names of the occurrences, either in the form "genus species" or "genus_species".
species	character. The name of the column in occdf containing the species-level identifications (i.e. the specific epithet).
genus	character. The name of the column in occdf containing the genus-level identifications.
...	character. Other named arguments specifying columns of higher levels of taxonomy (e.g. subfamily, order, superclass). The names of the arguments will be the column names of the output, and the values of the arguments correspond to the columns of occdf. The given order of the arguments is the order in which they are filtered. Therefore, these arguments must be in ascending order from lowest to highest taxonomic rank (see examples below). At least one higher level of taxonomy must be specified.
name	character. The name of the column in occdf containing the taxonomic names at mixed taxonomic levels; the data column "accepted_name" in a Paleobiology Database occurrence dataframe is of this type.
resolution	character. The taxonomic resolution at which to identify unique occurrences, either "species" (the default) or "genus".
append	logical. Should the original dataframe be returned with the unique names appended as a new column?

Details

Palaeobiologists usually count unique taxa by retaining only unique occurrences identified to a given taxonomic resolution, however this function retains occurrences identified to a coarser taxonomic resolution which are not already represented within the dataset. For example, consider the following set of occurrences:

- *Albertosaurus sarcophagus*
- *Ankylosaurus* sp.
- Aves indet.
- Ceratopsidae indet.
- Hadrosauridae indet.
- *Ornithomimus* sp.
- *Tyrannosaurus rex*

A filter for species-level identifications would reduce the species richness to two. However, none of these clades are nested within one another, so each of the indeterminately identified occurrences represents at least one species not already represented in the dataset. This function is designed to deal with such taxonomic data, and would retain all seven 'species' in this example.

Taxonomic information is supplied within a dataframe, in which columns provide identifications at different taxonomic levels. Occurrence data can be filtered to retain either unique species, or

unique genera. If a species-level filter is desired, the minimum input requires either (1) `binomial`, (2) `species` and `genus`, or (3) `name` and `genus` columns to be entered, as well as at least one column of a higher taxonomic level. In a standard Paleobiology Database occurrence dataframe, species names are only captured in the 'accepted_name' column, so a species-level filter should use 'genus = "genus"' and 'name = "accepted_name"' arguments. If a genus-level filter is desired, the minimum input requires either (1) `binomial` or (2) `genus` columns to be entered, as well as at least one column of a higher taxonomic level.

Missing data should be indicated with NAs, although the function can handle common labels such as "NO_FAMILY_SPECIFIED" within Paleobiology Database datasets.

The function matches taxonomic names at face value, so homonyms may be falsely filtered out.

Value

A `dataframe` of taxa, with each row corresponding to a unique "species" or "genus" in the dataset (depending on the chosen resolution). The dataframe will include the taxonomic information provided into the function, as well as a column providing the 'unique' names of each taxon. If `append` is `TRUE`, the original dataframe (`occdf`) will be returned with these 'unique' names appended as a new column. Occurrences that are identified to a coarse taxonomic resolution and belong to a clade which is already represented within the dataset will have their 'unique' names listed as `NA`.

References

Mannion, P. D., Upchurch, P., Carrano, M. T., and Barrett, P. M. (2011). Testing the effect of the rock record on diversity: a multidisciplinary approach to elucidating the generic richness of sauropodomorph dinosaurs through time. Biological Reviews, 86, 157-181. doi:10.1111/j.1469-185X.2010.00139.x.

Developer(s)

Bethany Allen & William Gearty

Reviewer(s)

Lewis A. Jones & William Gearty

Examples

```
#Retain unique species
occdf <- tetrapods[1:100, ]
species <- tax_unique(occdf = occdf, genus = "genus", family = "family",
order = "order", class = "class", name = "accepted_name")

#Retain unique genera
genera <- tax_unique(occdf = occdf, genus = "genus", family = "family",
order = "order", class = "class", resolution = "genus")

#Append unique names to the original occurrences
genera_append <- tax_unique(occdf = occdf, genus = "genus", family = "family",
order = "order", class = "class", resolution = "genus", append = TRUE)
```

```
#Create dataframe from lists
occdf2 <- data.frame(species = c("rex", "aegyptiacus", NA), genus =
c("Tyrannosaurus", "Spinosaurus", NA), family = c("Tyrannosauridae",
"Spinosauridae", "Diplodocidae"))
dinosaur_species <- tax_unique(occdf = occdf2, species = "species", genus =
"genus", family = "family")

#Retain unique genera per collection with group_apply
genera <- group_apply(occdf = occdf,
                      group = c("collection_no"),
                      fun = tax_unique,
                      genus = "genus",
                      family = "family",
                      order = "order",
                      class = "class",
                      resolution = "genus")
```

`tetrapods` *Example dataset: Early tetrapod data from the Paleobiology Database*

Description

A dataset of tetrapod occurrences ranging from the Carboniferous through to the Early Triassic, from the Palaeobiology Database. Dataset includes a range of variables relevant to common palaeobiological analyses, relating to identification, geography, environmental context, traits and more. Additional information can be found here. The downloaded data is unaltered, with the exception of removing some superfluous variables, and can be used to demonstrate how the functions in the palaeoverse package might be applied.

Usage

```
tetrapods
```

Format

A data frame with 5270 rows and 32 variables:

occurrence_no Reference number given to the particular occurrence in the Paleobiology Database

collection_no Reference number given to the Paleobiology Database collection (locality) that the occurrence belongs to

identified_name Taxon name as it appears in the original publication, which may include expressions of uncertainty (e.g. "cf.", "aff.", "?") or novelty (e.g. "n. gen.", "n. sp.")

identified_rank The taxonomic rank, or resolution, of the identified name

accepted_name Taxon name once the identified name has passed through the Paleobiology Database's internal taxonomy, which collapses synonyms, amends binomials which have been altered (e.g. species moving to another genus) and updates taxa which are no longer valid (e.g. *nomina dubia*)

accepted_rank The taxonomic rank, or resolution, of the accepted name

early_interval The oldest (or only) time interval within which the occurrence is thought to have been deposited

late_interval The youngest time interval within which the occurrence is thought to have been deposited

max_ma, min_ma The age range given to the occurrence

phylum, class, order, family, genus The taxa (of decreasing taxonomic level) which the occurrence is identified as belonging to

abund_value, abund_unit The number (and units) of fossils attributed to the occurrence

lng, lat The modern-day longitude and latitude of the fossil locality

collection_name The name of the Paleobiology Database collection which the occurrence belongs to, typically a spatio-temporally restricted locality

cc The country (code) where the fossils were discovered

formation, stratgroup, member The geological units from which the fossils were collected

zone The biozone which the occurrence is attributed to

lithology1 The main lithology of the beds in the section where the fossils were collected

environment The inferred environmental conditions in the place of deposition

pres_mode The mode of preservation of the fossils found in the collection (not necessarily of that specific occurrence), which will include information on whether they are body or trace fossils

taxon_environment The environment within which the taxon is thought to have lived, collated within the Paleobiology Database

motility, life_habit, diet Various types of trait data for the taxon, collated within the Paleobiology Database

References

Uhen MD et al. (in prep). Paleobiology Database User Guide.

Source

Compiled by Bethany Allen, current version downloaded on 14th July 2022. See item descriptions for details.

`time_bins` *Generate time bins*

Description

A function to generate time bins for a given study interval. This function is flexible in that either stage-level or higher stratigraphic-level (e.g. period) time bins can be called. In addition, near equal-length time bins can be generated by grouping stages together. For example, for 10 Myr as a target bin size, the function will generate groups of bins that have a mean bin length close to 10 Myr. However, users may also want to consider grouping stages based on other reasoning e.g. availability of outcrop (see Dean et al. 2020).

Usage

```
time_bins(
  interval = "Phanerozoic",
  rank = "stage",
  size = NULL,
  assign = NULL,
  scale = "GTS2020",
  plot = FALSE
)
```

Arguments

interval	character or numeric. Interval name available in GTS2020 or GTS2012. If a single interval name is provided, this interval is returned. If two interval names are provided, these intervals and those existing between are returned. If a single numeric age is provided, the interval that covers this age is returned. If two numeric ages are provided, the intervals occurring in the range of these ages are returned. Defaults to "Phanerozoic".
rank	character. Which stratigraphic rank is desired? Choose from: "stage", "epoch", "period", "era", and "eon". If scale is a dataframe, this argument is ignored.
size	numeric. If equal-length time bins are desired, specify the length in millions of years (Myr) of the time bins desired.
assign	numeric. A numeric vector of age estimates to use to assign to requested bins. If assign is specified, a numeric vector is returned of the midpoint age of the specified bins. Note this is the simplified approach of assignment in palaeoverse included for data with 'known' point-age estimates. For a wider range of binning methods, see palaeoverse::bin_time().
scale	character or data.frame. Specify the desired geological timescale to be used "GTS2020" (default), "GTS2012" or a user-input data.frame. If a data.frame is provided, it must contain at least the following named columns: "interval_name", "max_ma", and "min_ma". Column names "name", "max_age", and "min_age" are also accepted, but these are assumed to be equivalent to the aforementioned. As such, age data should be provided in Ma.
plot	logical. Should a plot of time bins be generated?

Details

This function uses either the Geological Time Scale 2020, Geological Time Scale 2012, or a user-input data.frame (see scale argument) to generate time bins. Interval data hosted by Macrostrat are also compatiable and accessible via the deeptime R pacakge (get_scale_data). Additional information on included Geological Time Scales and source can be accessed via:

- GTS2020
- GTS2012

Available intervals names are accessible via the interval_name column in GTS2012 and GTS2020. Data of the Geological Timescale 2020 and 2012 were compiled by Lewis A. Jones (2022-07-02).

Value

A `dataframe` of time bins for the specified intervals or a list with a `dataframe` of time bins and a named `numeric` vector (bin number) of binned age estimates (midpoint of specified bins) if assign is specified.

References

Dean, C.D., Chiarenza, A.A. and Maidment, S.C., 2020. Formation binning: a new method for increased temporal resolution in regional studies, applied to the Late Cretaceous dinosaur fossil record of North America. Palaeontology, 63(6), 881-901. doi:10.1111/pala.12492.

Developer(s)

Lewis A. Jones

Reviewer(s)

Kilian Eichenseer & William Gearty

Examples

```
#Using numeric age
ex1 <- time_bins(interval = 10, plot = TRUE)

#Using numeric age range
ex2 <- time_bins(interval = c(50, 100), plot = TRUE)

#Using a single interval name
ex3 <- time_bins(interval = c("Maastrichtian"), plot = TRUE)

#Using a range of intervals and near-equal duration bins
ex4 <- time_bins(interval = c("Fortunian", "Meghalayan"),
                 size = 10, plot = TRUE)

#Assign bins based on given age estimates
ex5 <- time_bins(interval = c("Fortunian", "Meghalayan"),
                 assign = c(232, 167, 33))

#Use user-input data.frame to generate near-equal length bins
scale <- data.frame(interval_name = 1:5,
                    min_ma = c(0, 18, 32, 38, 45),
                    max_ma = c(18, 32, 38, 45, 53))
ex6 <- time_bins(scale = scale, size = 20, plot = TRUE)

#Use North American land mammal ages from deeptime/Macrostrat
scale <- deeptime::get_scale_data(name = "North American land mammal ages")
ex7 <- time_bins(scale = scale, size = 10)
```

Index