Package ‘StratPal’

May 21, 2025

Title Stratigraphic Paleobiology Modeling Pipelines

Version 0.5.0

Description The fossil record is a joint expression of ecological, taphonomic, evolutionary, and stratigraphic processes (Holland and Patzkowsky, 2012, ISBN:978-0226649382).
This package allowing to simulate biological processes in the time domain (e.g., trait evolution, fossil abundance, phylogenetic trees), and examine how their expression in the rock record (stratigraphic domain) is influenced based on age-depth models, ecological niche models, and taphonomic effects.
Functions simulating common processes used in modeling trait evolution or event type data such as first/last occurrences are provided and can be used standalone or as part of a pipeline. The package comes with example data sets and tutorials in several vignettes, which can be used as a template to set up one's own simulation.

License Apache License (>= 2)

Encoding UTF-8

RoxygenNote 7.3.2

Imports admtools (>= 0.6.0), paleoTS

Suggests ape, FossilSim, knitr, rmarkdown, spelling, testthat (>= 3.0.0)

VignetteBuilder knitr

Depends R (>= 4.2)

LazyData true

URL `https://mindthegap-erc.github.io/StratPal/` , `https://github.com/MindTheGap-ERC/StratPal`

BugReports `https://github.com/MindTheGap-ERC/StratPal/issues`

Config/testthat/edition 3

Language en-US

NeedsCompilation no

Author Niklas Hohmann [aut, cre] (ORCID: <`https://orcid.org/0000-0003-1559-1838`>)

Maintainer Niklas Hohmann <`N.H.Hohmann@uu.nl`>

Repository CRAN

Date/Publication 2025-05-21 08:40:02 UTC

Contents

`apply_niche` *apply niche model*

Description

Models niches by removing events (fossil occurrences) or specimens when they are outside of their niche. For event type data, this is done using the function `thin`, for `pre_paleoTS` this is done by applying the function `prob_remove` on the specimens. For `fossils` objects produced by the `FossilSim` package, fossil observations are removed according to the specified niche Combines the functions `niche_def` and `gc` ("gradient change") to determine how the taxons' collection probability changes with time/position. This is done by composing `niche_def` and `gc`. The result is then used to remove events/specimens in `x`.

Usage

```
apply_niche(x, niche_def, gc)
```

Arguments

`x`	events type data, e.g. vector of times/ages of fossil occurrences or their stratigraphic position, or a `pre_paleoTS` object (e.g. produced by `stasis_sl`), or a `fossils` object produced by the `FossilSim` package.
`niche_def`	function, specifying the niche along a gradient. Should return 0 when taxon is outside of niche, and 1 when inside niche. Values between 0 and 1 are interpreted as collection probabilities. Must be vectorized, meaning if given a vector, it must return a vector of equal length.
`gc`	function, stands for "gradient change". Specifies how the gradient changes, e.g. with time. Must be vectorized, meaning if given a vector, it must return a vector of equal length.

Value

for a numeric vector input, returns a numeric vector, timing/location of events (e.g. fossil ages/locations) preserved after the niche model is applied. For a `pre_paleoTS` object as input, returns a `pre_paleoTS` object with specimens removed according to the niche model. For a `fossils` object, returns a `fossils` object with some occurrences removed according to the niche definition

See Also

- `snd_niche()` and `bounded_niche()` for template niche models, `discrete_niche()` and `discrete_gradient()` to construct niches from discrete categories.
- `vignette("advanced_functionality)` for how to create user-defined niche models
- `apply_taphonomy()` to model taphonomic effects based on a similar principle
- `thin()` and `prob_remove()` for the underlying mathematical procedures

Examples

```
### example for event type data
## setup
# using water depth as gradient
 t = scenarioA$t_myr
 wd = scenarioA$wd_m[,"8km"]
 gc = approxfun(t, wd)
 plot(t, gc(t), type = "l", xlab = "Time", ylab = "water depth [m]",
  main = "gradient change with time")
 # define niche
 # preferred wd 10 m, tolerant to intermediate wd changes (standard deviation 10 m), non-terrestrial
 niche_def = snd_niche(opt = 10, tol = 10, cutoff_val = 0)
 plot(seq(-1, 50, by = 0.5), niche_def(seq(-1, 50, by = 0.5)), type = "l",
 xlab = "water depth", ylab = "collection probability", main = "Niche def")
 # niche pref with time
 plot(t, niche_def(gc(t)), type = "l", xlab = "time",
 ylab = "collection probability", main = "collection probability with time")

 ## simulate fossil occurrences
 foss_occ = p3(rate = 100, from = 0, to = max(t))
 # foss occ without niche pref
```

```
hist(foss_occ, xlab = "time")
foss_occ_niche = apply_niche(foss_occ, niche_def, gc)
# fossil occurrences with niche preference
hist(foss_occ_niche, xlab = "time")

# see also
#vignette("event_data")
# for a detailed example on niche modeling for event type data

### example for pre_paleoTS objects
# we reuse the niche definition and gradient change from above!
x = stasis_sl(seq(0, max(t), length.out = 10))
plot(reduce_to_paleoTS(x), main = "Trait evolution before niche modeling")
y = apply_niche(x, niche_def, gc)
plot(reduce_to_paleoTS(y), main = "Trait evolution after niche modeling")
# note that there are fewer sampling sites
# bc the taxon does not appear everywhere
# and there are fewer specimens per sampling site

### example for fossils objects
# we reuse the niche definition and gradient change from above
# simulate tree
tree = ape::rlineage(birth = 2, death = 0, Tmax = 2)
# create fossils object
f = FossilSim::sim.fossils.poisson(rate = 2, tree = tree)
# plot fossils along tree before niche model is applied
FossilSim:::plot.fossils(f, tree = tree)
# introduce niche model
f_mod = f |>
  admtools::rev_dir(ref = max(t)) |> # reverse direction bc FossilSim uses age not time
  apply_niche(niche_def, gc) |>
  admtools::rev_dir(ref = max(t))
# plot fossils along tree after introduction of niche model
FossilSim:::plot.fossils(f_mod, tree = tree)
# note how only fossils in the interval where environmental conditions are suitable are preserved
# note that FossilSim uses age before the present, which is why we use admtools::rev_dir
```

`apply_taphonomy` *model taphonomic effects*

Description

Models taphonomy by combining the change in taphonomic conditions with the preservation potential as a function of taphonomic conditions to determine how preservation potential changes. This is then used to systematically remove (thin) the event data using `thin`/ remove specimens from the `pre_paleoTS` object using `prob_remove`.

Usage

```
apply_taphonomy(x, pres_potential, ctc)
```

Arguments

`x`	event type data, e.g. times/ages of fossil occurrences or their stratigraphic position, or a `pre_paleoTS` or a `fossils` object.
`pres_potential`	function. Takes taphonomic conditions as input and returns the preservation potential (a number between 0 and 1). Must be vectorized, meaning if given a vector, it must return a vector of equal length.
`ctc`	function, change in taphonomic conditions (ctc) with time or stratigraphic position. . Must be vectorized, meaning if given a vector, it must return a vector of equal length.

Value

if given event type data, a numeric vector, location/timing of events (e.g. fossil occurrences) after the taphonomic filter is applied. If given a `pre_paleoTS` object, returns another `pre_paleoTS` object with reduced number of specimens. If given a `fossils` object as created by the `FossilSim` package, returns another `fossils` object with some occurrences removed according to preservation potential.

See Also

- `apply_niche()` for modeling niche preferences based on the same principle. Internally, these functions are structured identically.
- `thin()` and `prob_remove()` for the underlying mathematical procedures.

Examples

```
# see
#vignette("advanced_functionality")
# for details on usage
# or the documentation of apply_ecology for equivalent application to ecology
```

`bounded_niche` *define niche from boundaries*

Description

Defines a simple niche model where the niche defined is given by a lower limit (`g_min`) and an upper limit (`g_max`) of a gradient the taxon can tolerate

Usage

```
bounded_niche(g_min, g_max)
```

Arguments

`g_min`	lowest value of the gradient the taxon can tolerate
`g_max`	highest value of the gradient the taxon can tolerate

Value

a function describing the niche for usage with `apply_niche`. The function returns 1 if the taxon is within its niche (the gradient is between `g_min` and `g_max`), and 0 otherwise

See Also

- `snd_niche()` for an alternative niche model
- `discrete_niche()` for defining niches based on discrete categories
- `apply_niche()` for the function that uses the function returned
- `vignette("advanced_functionality")` for details how to create user-defined niche models

Examples

```
x = seq(0, 10, by = 0.2)
f = bounded_niche(2,5)
plot(x, f(x), type = "l",
xlab = "Gradient", ylab = "Observation probability",
main = "Observation probability of taxon")

# see also
#vignette("event_data")
# for details how to use this functionality
```

`discrete_gradient` *construct discretized gradient*

Description

Constructs a discretized gradient along time/height. The gradient value between `bounds[i]` and `bounds[i+1]` (not including) is `vals[i]`, values above/below the largest/smallest value of bounds are assigned `outval`. Helper function for usage with `discrete_niche`

Usage

```
discrete_gradient(vals, bounds, outval = "")
```

Arguments

`vals`	vector, values of the gradient
`bounds`	vector of strictly increasing values, e.g. times or stratigraphic heights.
`outval`	value, gradient value assigned outside of values covered by `bounds`

Value

a functions assigning continuous values (e.g., times or heights) discrete niches

See Also

- `discrete_niche()` to construct niches based on discretized gradients
- `apply_niche()` to combine `discrete_niche()` and `discrete_gradient()` to model the effects of niches
- `vignette("advanced_functionality")` for details how to create user-defined niche models

Examples

```
# see examples in `discrete_niche` for a use case
# and examples in `apply_niche` for the general application to different data types
# or the vignette on event data for more context
```

`discrete_niche` *niche from discrete data*

Description

Defines a niche model where the gradient based on discrete bins (given by `binS`)

Usage

```
discrete_niche(bins, rec_prob, outval = 0)
```

Arguments

`bins`	vector, bins on which the niche is defined. Can e.g., be numeric or character
`rec_prob`	numeric vector, recovery probability for the bins. Must contain values between 0 and 1
`outval`	recovery probability for values not in `bin`

Value

a function describing the niche for usage with `apply_niche`. The function takes (vectors of) values from `bin` as input and returns recovery probability for this bin.

See Also

- `discrete_gradient()` to construct gradients based on discrete categories.
- `snd_niche()` to define niches along a continuous gradient based on a scaled normal distribution
- `bounded_niche()` to define niches along a continuous gradient based on hard boundaries
- `apply_niche()` for the function used to apply niches to time series or events
- `vignette("advanced_functionality")` for details on how to create user-defined niche models

Examples

```
# example workflow of how to construct discrete niches. For details on
#how this can be used in conjunction with apply_niche, see documentation
#therein of the vignette on event data
# we model a simple niche, separated into "shallow water" and "deep water"
bins = c("shallow water", "deep water")
# taxon is more abundant in shallow water
rec_prob = c(0.9, 0.1)
# 90 % recovery probability in shallow water, 10 % in deep water
niche = discrete_niche(bins = bins, rec_prob = rec_prob)
# lets assume for the first 1 Myr, water is shallow, followed by 0.8 Myr of deep water,
# and then 1 Myr of shallow water again
# define discretized gradient using discrete_gradient
gradient = discrete_gradient(vals = c("shallow water", "deep water", "shallow water"),
bounds = c(0,1,1.8,2.8))

# assuming constant fossil abundance before ecological effects, how many fossils do we recover?
foss_occ = p3(rate = 100, from = 0, to = 2.8)
occ_after_ecol = apply_niche(foss_occ, niche_def = niche, gc = gradient)
hist(occ_after_ecol, xlab = "Myr")
# Between 1 nad 1.8 Myr fossil abundance is reduced because this coincides with deep whater
# in which the recovery potential of the taxon is reduced (from 90 % to 10 %)
```

`ornstein_uhlenbeck` *simulate ornstein-uhlenbeck (OU) process*

Description

Simulates an Ornstein-Uhlenbeck process using the Euler-Maruyama method. The process is simulated on a scale of `0.25 * min(diff(t))` and then interpolated to the values of `t`. Note that different parametrizations of OU processes are used in the literature. Here we use the parametrization common in mathematics. This translates to the parametrization used in evolutionary biology (specifically, the one in Hansen (1997)) as follows:

- `sigma` is identical
- `mu` used in the `StratPal` package corresponds to `theta` sensu Hansen (1997)
- `theta` as used in the `StratPal` package corresponds to `alpha` sensu Hansen (1997)

Usage

```
ornstein_uhlenbeck(t, mu = 0, theta = 1, sigma = 1, y0 = 0)
```

Arguments

`t`	times at which the process is simulated. Can be heterodistant
`mu`	number, long term mean
`theta`	number, mean reversion speed
`sigma`	positive number, strength of randomness
`y0`	number, initial value (value of process at the first entry of t)

Value

A list with two elements: `t` and `y`. `t` is a duplicate of the input `t`, `y` are the values of the OU process at these times. Output list is of S3 class `timelist` (inherits from `list`) and can thus be plotted directly using `plot`, see `?admtools::plot.timelist`

References

- Hansen, Thomas F. 1997. "Stabilizing Selection and the Comparative Analysis of Adaptation." Evolution 51 (5): 1341–51. doi:10.1111/j.15585646.1997.tb01457.x.

See Also

- `ornstein_uhlenbeck_sl()` for simulation on specimen level - for use in conjunction with `paleoTS` package
- `random_walk()` and `stasis()` to simulate other modes of evolution

Examples

```
library("admtools") # required for plotting of results
t = seq(0, 3, by = 0.01)
l = ornstein_uhlenbeck(t, y0 = 3) # start away from optimum (mu)
plot(l, type = "l")
l2 = ornstein_uhlenbeck(t, y0 = 0) # start in optimum
lines(l2$t, l2$y, col = "red")
```

`ornstein_uhlenbeck_sl` *simulate ornstein-uhlenbeck (OU) process (specimen level)*

Description

Simulates an Ornstein-Uhlenbeck process on specimen level (_sl). The mean trait value is simulated using the Euler-Maruyama method. The process is simulated on a scale of `0.25 * min(diff(t))` and then interpolated to the values of `t`. At each sampling location there are `n_per_sample` specimens that are normally distributed around the mean trait value with a variance of `intrapop_var`. Note that different parametrizations of OU processes are used in the literature. Here we use the parametrization common in mathematics. This translates to the parametrization used in evolutionary biology (specifically, the one in Hansen (1997)) as follows:

- `sigma` is identical
- `mu` used in the `StratPal` package corresponds to `theta` sensu Hansen (1997)
- `theta` as used in the `StratPal` package corresponds to `alpha` sensu Hansen (1997)

Usage

```
ornstein_uhlenbeck_sl(
  t,
  mu = 0,
  theta = 1,
  sigma = 1,
  y0 = 0,
  intrapop_var = 1,
  n_per_sample = 10
)
```

Arguments

`t`	times at which the process is simulated. Can be heterodistant
`mu`	number, long term mean
`theta`	number, mean reversion speed
`sigma`	positive number, strength of randomness
`y0`	number, initial value (value of process at the first entry of t)
`intrapop_var`	intrapopulation variance, determines how much specimens from the same population vary
`n_per_sample`	integer, number of specimens sampled per population/sampling locality

Value

an object of S3 class `pre_paleoTS`, inherits from `timelist` and `list`. The list has two elements: `t`, containing a vector of times of sampling, and `vals`, a list of trait values of the same length as `t`, with element containing trait values of individual specimens. This object can be transformed using `apply_taphonomy`, `apply_niche` or `time_to_strat`, and then reduced to a `paleoTS` object using `reduce_to_paleoTS`. This can then be used to test for different modes of evolution.

See Also

- `ornstein_uhlenbeck()` to model mean trait values,
- `reduce_to_paleoTS()` to transform outputs into `paleoTS` format
- `stasis_sl()`, `strict_stasis_sl()` and `random_walk_sl()` to simulate other modes of evolution

Examples

```
library("paleoTS")
x = ornstein_uhlenbeck_sl(1:5)
y = reduce_to_paleoTS(x) # turn into paleoTS format
plot(y) # plot using the paleoTS package

# see also
#vignette("paleoTS_functionality")
#for details and advanced usage
```

`p3` *simulate Poisson point process*

Description

Simulates events in the interval `from` to `to` based on a Poisson point process with rate `rate`. If the parameter `n` is used, the number of fossils is conditioned to be `n` In the context of paleontology, these events can be interpreted as fossil occurrences or first/last occurrences of species. In this case, the rate is the average number of fossil occurrences (resp first/last occurrences) per unit

Usage

```
p3(rate, from, to, n = NULL)
```

Arguments

`rate`	strictly positive number, rate of events (avg events per unit)
`from`	lowest boundary of observed interval
`to`	upper boundary of observed interval
`n`	integer of NULL (default). Number of events to return. If NULL, the number is random and determined by the rate parameter

Value

a numeric vector with timing/location of events.

See Also

`p3_var_rate()` for the variable rate implementation

Examples

```
# for fossil occ.
x = p3(rate = 5, from = 0, to = 1) # 5 fossil occurrences per myr on avg.
hist(x, xlab = "Time (Myr)", ylab = "Fossil Occurrences" )

x = p3(rate = 3, from = 0, to = 4)
hist(x, main = paste0(length(x), " samples")) # no of events is random

x = p3(rate = 3, from = 0, to = 4, n = 10)
hist(x, main = paste0(length(x), " samples")) # no of events is fixed to n

# see also
#vignette("event_data")
# for details on usage and applications to paleontology
```

`p3_var_rate` *simulate variable rate Poisson point process*

Description

simulates events based on a variable rate Poisson point process. Rates can be either specified by a function passed to `x`, or by providing two vectors `x` and `y`. In this case the rate is specified by `approxfun(x, y, rule = 2)`, i.e. by linear interpolation between the values of x (abscissa) and y (ordinate). See `?approxfun` for details. In the context of paleontology, these events can be interpreted as fossil occurrences or first/last occurrences of species. In this case, the rate is the average number of fossil occurrences (resp first/last occurrences) per unit

Usage

```
p3_var_rate(x, y = NULL, from = 0, to = 1, f_max = 1, n = NULL)
```

Arguments

`x`	numeric vector or function. If x is a function, it is used to specify the variable rate. If x is a vector, x and y together specify the variable rate using linear interpolation
`y`	numeric vector or NULL. If not NULL, determines the variable rate. This is done by using linear interpolation between the values of y. Here x specifies the ordinate and y the abscissa
`from`	lower boundary of the observed interval
`to`	upper boundary of the observed
`f_max`	maximum value of `x` in the interval from `x_min` to `x_max`. If `x` attains values larger than `f_max` a warning is throw, `f_max` is adjusted, and sampling is started again
`n`	NULL or an integer. Number of events drawn. If NULL, the number of events is determined by the rate (specified by x and y). If an integer is passed, n events are returned.

Value

numeric vector, timing/location of events. Depending on the modeling framework, these events can represent location/age of fossils, or first/last occurrences of a group of taxa.

See Also

`p3()` for the constant rate implementation, `rej_samp()` for the underlying random number generation.

Examples

```
# assuming events are fossil occurrences
# then rate is the avg rate of fossil occ. per unit
#linear decrease in rate from 50 at x = 0 to 0 at x = 1
x = c(0, 1)
y = c(50, 0)
s = p3_var_rate(x, y, f_max = 50)
hist(s, xlab = "Time (myr)", main = "Fossil Occurrences")
# conditioned to return 100 samples
s = p3_var_rate(x, y, f_max = 50, n = 100)
# hand over function
s = p3_var_rate(x = sin, from = 0 , to = 3 * pi, n = 50)
hist(s) # note that negative values of f (sin) are ignored in sampling

# see also
#vignette("event_data")
# for details on usage and applications to paleontology
```

`plot.pre_paleoTS` *plot pre-paleoTS objects*

Description

This functions throws an error on purpose, as `pre_paleoTS` objects can not be plotted directly. To plot them, first use `reduce_to_paleoTS` and use `plot` on the results

Usage

```
## S3 method for class 'pre_paleoTS'
plot(x, ...)
```

Arguments

`x`	object
`...`	other arguments

See Also

`reduce_to_paleoTS()`

Examples

```
## Not run:
x = stasis_sl(1:4)
# throws error
plot(x)
library("paleoTS")
```

```
# correct way to plot pre-paleoTS objects
y = reduce_to_paleoTs(x)
plot(y)
# this plots via the procedures of the paleoTS package (which must be installed and loaded)

## End(Not run)
```

`prob_remove` *probabilistic removal of elements*

Description

probabilistic removal of elements from x. For each element, the probability to be preserved is independent and specified by prob

Usage

```
prob_remove(x, prob)
```

Arguments

`x`	vector
`prob`	number between 0 and 1, probability to preserve elements

Value

a vector of the same type as x

See Also

- `apply_niche()` and `apply_taphonomy()` for functions that use this function for transformation of `pre_paleoTS` objects

Examples

```
x = prob_remove(1:10, 0.5)
x
x = prob_remove(1:10, 0.5)
x
```

`random_walk` *simulate (un)biased random walk*

Description

Simulates a (continuous time) random walk as a Brownian drift. For `mu = 0` the random walk is unbiased, otherwise it is biased.

Usage

```
random_walk(t, sigma = 1, mu = 0, y0 = 0)
```

Arguments

`t`	numeric vector with strictly increasing elements, can be heterodistant. Times at which the random walk is evaluated
`sigma`	positive number, variance parameter
`mu`	number, directionality parameter
`y0`	number, starting value (value of the random walk at the first entry of `t`)

Value

A list with elements `t` and `y`. `t` is a duplicate of the input parameter and is the times at which the random walk is evaluated. `y` are the values of the random walk at said times. Output list is of S3 class `timelist` (inherits from `list`) and can thus be plotted directly using `plot`, see `?admtools::plot.timelist`

See Also

- `stasis()` and `ornstein_uhlenbeck()` to simulate other modes of evolution
- `random_walk_sl()` to simulate random walk on specimen level - for usage in conjunction with the `paleoTS` package

Examples

```
library("admtools") # required for plotting of results
t = seq(0, 1, by = 0.01)
l = random_walk(t, sigma = 3) # high variability, no direction
plot(l, type = "l")
l2 = random_walk(t, mu = 1) # low variabliity, increasing trend
lines(l2$t, l2$y, col = "red")
```

`random_walk_sl` *simulate (un)biased random walk (specimen level)*

Description

Simulates a (continuous time) random walk as a Brownian drift on specimen level. For `mu = 0` the random walk is unbiased, otherwise it is biased.

Usage

```
random_walk_sl(
  t,
  sigma = 1,
  mu = 0,
  y0 = 0,
  intrapop_var = 1,
  n_per_sample = 10
)
```

Arguments

`t`	numeric vector with strictly increasing elements, can be heterodistant. Times at which the random walk is evaluated
`sigma`	positive number, variance parameter
`mu`	number, directionality parameter
`y0`	number, starting value (value of the random walk at the first entry of `t`)
`intrapop_var`	intrapopulation variance, determines how much specimens from the same population vary
`n_per_sample`	integer, number of specimens sampled per population/sampling locality

Value

an object of S3 class `pre_paleoTS`, inherits from `timelist` and `list`. The list has two elements: `t`, containing a vector of times of sampling, and `vals`, a list of trait values of the same length as `t`, with element containing trait values of individual specimens. This object can be transformed using `apply_taphonomy`, `apply_niche` or `time_to_strat`, and then reduced to a `paleoTS` object using `reduce_to_paleoTS`. This can then be used to test for different modes of evolution.

See Also

- `random_walk()` for the equivalent function to simulate mean trait values
- `reduce_to_paleoTS()` to transform outputs into `paleoTS` format.
- `stasis_sl()`, `strict_stasis_sl()` and `ornstein_uhlenbeck_sl()` to simulate other modes of evolution

Examples

```
library("paleoTS")
x = random_walk_sl(1:5)
y = reduce_to_paleoTS(x) # turn into paleoTS format
plot(y) # plot using the paleoTS package
# see also
#vignette("paleoTS_functionality")
#for details and advanced usage
```

`reduce_to_paleoTS` *reduce pre-paleoTS format to paleoTS*

Description

paleoTS is a format for paleontological time series. It is a summary format where interpopulation variance is provided as a parameter. As a result, taphonomic and ecological effects that act on individual specimens can not be modeled for paleoTS objects. To resolve this, the pre_paleoTS format tracks each specimen individually. This function reduces the pre-paleoTS format into standard paleoTS object, which can be used by the paleoTS package.

Usage

```
reduce_to_paleoTS(x, min_n = 1, na.rm = TRUE, ...)
```

Arguments

`x`	a `pre_paleoTS` object
`min_n`	minimum number of specimens. If the number of specimens at a sampling location falls below this number, the sampling location will be removed
`na.rm`	Logical. If sampling locations are NA (e.g., because of erosion), should the sample be removed?
`...`	other options. currently unused

Value

a `paleoTS` object

See Also

- `stasis_sl()`, strict_stasis_sl, random_walk_sl, and `ornstein_uhlenbeck_sl()` to simulate trait evolution on specimen level (sl), returning an object of type `pre_paleoTS`

Examples

```
x = stasis_sl(t = 0:5)    # create pre_paleoTS object representing stasis on specimen level
y = reduce_to_paleoTS(x)   # reduce to standard paleoTS format
plot(y)
# now analyses using the paleoTS package can be applied to y
```

`rej_samp` *random numbers from rejection sampling*

Description

Rejection sampling from the (pseudo) pdf `f` in the interval between `x_min` and `x_max`. Returns `n` samples. Note that values of `f` below 0 are capped to zero

Usage

```
rej_samp(f, x_min, x_max, n = 1L, f_max = 1, max_try = 10^4)
```

Arguments

`f`	function. (pseudo) pdf from which the sample is drawn
`x_min`	number, lower limit of the examined interval
`x_max`	number, upper limit of the examined interval
`n`	integer. number of samples drawn
`f_max`	number, maximum value of `f` in the interval from `x_min` to `x_max`. If f attains values larger than `f_max` a warning is throw, `f_max` is adjusted, and sampling is started again
`max_try`	maximum number of tries in the rejection sampling algorithm. If more tries are needed, an error is thrown. If this is the case, inspect of your function `f` is well-defined and positive, and if `f_max` provides a reasonable upper bound on it. Adjust `max_try` if you are certain that both is the case, e.g. if `f` is highly irregular.

Value

numeric vector, sample of size `n` drawn from the (pseudo) pdf specified by `f`

See Also

`p3_var_rate()` for the derived variable rate Poisson point process implementation.

Examples

```
f = sin
x = rej_samp(f, 0, 3*pi, n = 100)
hist(x) # note that no samples are drawn where sin is negative
```

`scenarioA` *example data, scenario A from Hohmann et al. (2024)*

Description

Scenario A as described in Hohmann et al. (2024), published in Hohmann et al. (2023). Contains data from a carbonate platform simulated using CarboCAT Lite (Burgess 2013, 2023)

Usage

```
scenarioA
```

Format

A list with 6 elements:

- `t_myr` : numeric vector. timesteps of the simulation in Myr
- `sl_m` : numeric vector. eustatic sea level in m
- `dist_from_shore` : character vector. Distance from shore in km of locations at which the observations were made. Available distances are "2km", "4km", "6km", "8km", "10km", "12km".
- `h_m` : matrix of size length(t_myr) x length(dist_from_shore). Accumulated sediment height in m at examined locations
- `wd_m`: matrix of size length(t_myr) x length(dist_from_shore). Water depth in m at examined locations
- `strat_col`: list with length(dist_from shore) elements. Represents a stratigraphic column. Each element is a list with two elements:
  - `bed_thickness_m`: numeric vector. Bed thickness in m
  - `facies_code` : integer vector. facies code of the bed

References

- Burgess, Peter. 2013. "CarboCAT: A cellular automata model of heterogeneous carbonate strata." Computers & Geosciences. doi:10.1016/j.cageo.2011.08.026.
- Burgess, Peter. 2023. "CarboCATLite v1.0.1." Zenodo. doi:10.5281/zenodo.8402578
- Hohmann, Niklas; Koelewijn, Joël R.; Burgess, Peter; Jarochowska, Emilia. 2024. "Identification of the mode of evolution in incomplete carbonate successions." BMC Ecology and Evolution 24, 113. doi:10.1186/s12862024022872.
- Hohmann, Niklas, Koelewijn, Joël R.; Burgess, Peter; Jarochowska, Emilia. 2023. “Identification of the Mode of Evolution in Incomplete Carbonate Successions - Supporting Data.” Open Science Framework. doi:10.17605/OSF.IO/ZBPWA, published under the CC-BY 4.0 license.

`snd_niche` *simple niche model*

Description

Defines niche model based in the "Probability of collection" model by Holland and Patzkowsky (1999). The collection probability follows the shape of a bell curve across a gradient, where `opt` determines the peak (mean) of the bell curve, and `tol` the standard deviation. "snd" stands for "scaled normal distribution", as the collection probability has the shape of the probability density of the normal distribution.

Usage

```
snd_niche(opt, tol, prob_modifier = 1, cutoff_val = NULL)
```

Arguments

`opt`	optimum value, gradient value where collection probability is highest
`tol`	tolerance to changes in gradient. For large values, collection probability drops off slower away from `opt`
`prob_modifier`	collection probability modifier, collection probability at `opt`.
`cutoff_val`	NULL or a number. If a number, all collection probabilities at gradient values below `cutoff_value` are set to 0. This can for example be used to model exclusively marine species when the gradient is water depth (see examples).

Value

a function for usage with `apply_niche`.

References

- Holland, Steven M. and Patzkowsky, Mark E. 1999. "Models for simulating the fossil record." Geology. https://doi.org/10.1130/0091-7613(1999)027%3C0491:MFSTFR%3E2.3.CO;2

See Also

- `apply_niche()` for usage of the returned function
- `bounded_niche()` for another niche model
- `discrete_niche()` and `discrete_gradient()` to define niches based on discrete categories
- `vignette("advenced_functionality")` for details on how to create user defined niche models

Examples

```
# using water depth as niche
wd = seq(-3, 40, by = 0.5)
f = snd_niche(opt = 10, tol = 5)

plot(wd, f(wd), xlab = "Water depth", ylab = "Prob. of collection")
# set cutoff value at to 0 to model non-terrestrial species.
f = snd_niche(opt = 10, tol = 5, cutoff_val = 0)
plot(wd, f(wd), xlab = "Water depth", ylab = "Prob. of collection")

# see also
#vignette("event_data")
#for examples how to use it for niche modeling
```

`stasis` *simulate phenotypic stasis*

Description

Simulates stasis of mean trait values as independent, normally distributed random variables with mean `mean` and standard deviation `sd`

Usage

```
stasis(t, mean = 0, sd = 1)
```

Arguments

`t`	times at which the traits are determined
`mean`	number, mean trait value
`sd`	strictly positive number, standard deviation of traits

Value

A list with two elements: `t` and `y`. `t` is a duplicate of the input `t`, `y` are the corresponding trait values. Output list is of S3 class `timelist` (inherits from `list`) and can thus be plotted directly using `plot`, see `?admtools::plot.timelist`

See Also

- `random_walk()` and `ornstein_uhlenbeck()` to simulate other modes of evolution
- `stasis_sl()` to simulate stasis on specimen level - for usage in conjunction with the `paleoTS` package.

Examples

```
library("admtools") # required for plotting of results
t = seq(0, 1, by = 0.01)
l = stasis(t)
plot(l, type = "l") # plot lineage
l2 = stasis(t, mean = 0.5, sd = 0.3) # simulate second lineage
lines(l2$t, l2$y, col = "red") # plot second lineage
```

`stasis_sl` *simulate phenotypic stasis (specimen level)*

Description

simulates stasis as independent, normally distributed random variables with mean `mean` and standard deviation `sd`, draws `n_per_sample` samples from each sampling location (population) that have specified variance `intrapop_var`

Usage

```
stasis_sl(t, mean = 0, sd = 1, intrapop_var = 1, n_per_sample = 10)
```

Arguments

`t`	times at which the traits are determined
`mean`	mean trait value
`sd`	strictly positive number, standard deviation of traits around the mean
`intrapop_var`	intrapopulation variance, determines how much specimens from the same population vary
`n_per_sample`	integer, number of specimens sampled per population/sampling locality

Value

an object of S3 class `pre_paleoTS`, inherits from `timelist` and `list`. The list has two elements: `t`, containing a vector of times of sampling, and `vals`, a list of trait values of the same length as `t`, with element containing trait values of individual specimens. This object can be transformed using `apply_taphonomy`, `apply_niche` or `time_to_strat`, and then reduced to a `paleoTS` object using `reduce_to_paleoTS`. This can then be used to test for different modes of evolution.

See Also

- `stasis()` for the version that simulates stasis of mean trait values
- `strict_stasis_sl()` for more narrow definition of stasis
- `reduce_to_paleoTS()` to transform into the outputs into `paleoTS` format (e.g., for plotting or further analysis)
- `random_walk_sl()` and `ornstein_uhlenbeck_sl()` for other modes of evolution

Examples

```
library("paleoTS")
x = stasis_sl(1:5, mean = 2, sd = 2)
y = reduce_to_paleoTS(x) # turn into paleoTS format
plot(y) # plot using paleoTS package
# see also
#vignette("paleoTS_functionality")
#for details and advanced usage
```

`strict_stasis_sl` *simulate strict phenotypic stasis (specimen level)*

Description

simulates strict stasis on the population level (Hunt et al. 2015). This means each population has the same mean trait value, and all deviations are due to the fact that specimens traits differ from this value due to randomness.

Usage

```
strict_stasis_sl(t, mean = 0, intrapop_var = 1, n_per_sample = 10)
```

Arguments

`t`	times at which the traits are determined
`mean`	mean trait value
`intrapop_var`	intrapopulation variance, determines how much specimens from the same population vary
`n_per_sample`	integer, number of specimens sampled per population/sampling locality/time

Value

an object of S3 class `pre_paleoTS`, inherits from `timelist` and `list`. The list has two elements: `t`, containing a vector of times of sampling, and `vals`, a list of trait values of the same length as `t`, with element containing trait values of individual specimens. This object can be transformed using `apply_taphonomy`, `apply_niche` or `time_to_strat`, and then reduced to a `paleoTS` object using `reduce_to_paleoTS`. This can then be used to test for different modes of evolution.

References

- Hunt, Gene, Melanie J. Hopkins, and Scott Lidgard. 2015. “Simple versus Complex Models of Trait Evolution and Stasis as a Response to Environmental Change.” Proceedings of the National Academy of Sciences of the United States of America 112 (16): 4885–90. https://doi.org/10.1073/pnas.1403662111.

See Also

- `stasis_sl()` for the (non-strict) equivalent
- `reduce_to_paleoTS()` to transform outputs into `paleoTS` format
- `random_walk_sl()` and `ornstein_uhlenbeck_sl()` for other modes of evolution

Examples

```
library("paleoTS")
x = strict_stasis_sl(1:5, mean = 2, intrapop_var = 2) # simulate strict stasis
y = reduce_to_paleoTS(x)   # transform into paloeTS format
plot(y) # plot using paleoTS package

# see also
#vignette("paleoTS_functionality")
#for details and advanced usage
```

`thin` *thin a series of events (e.g. fossil occurrences)*

Description

Thins a vector of events using the function thin, meaning the probability that the ith event in x is preserved is given by *thin(x(i))*. Values of `thin` below 0 and above 1 are ignored. Is used to model niche preferences in `apply_niche` and taphonomic effects in `apply_taphonomy`.

Usage

```
thin(x, thin)
```

Arguments

`x`	numeric vectors with events (e.g. locations, height, times)
`thin`	a function used for thinning

Value

numeric vector, events after thinning. Depending on the modeling framework, these events can represent fossil ages/locations or first/last occurrences, and the thinning taphonomic or ecological effects.

See Also

- `apply_niche()` and `apply_taphonomy()` for use cases with biological meaning. Use `thin` to model effects of taphonomy and ecology for event data.

Examples

```
x = p3(rate = 100, from = 0, to = 3 * pi) # simulate Poisson point process
y = thin(x, sin)
hist(y) # not how negative values of sin are treated as 0
yy = thin(x, function(x) 5 * sin(x))
hist(yy) # note how values of 5 * sin above 1 are not affecting the thinning
```

Index