## Package 'StratPal'

November 24, 2025

Title Stratigraphic Paleobiology Modeling Pipelines

Version 0.7.0

**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, biostratigraphy 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>)

2 all\_preserved

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

**Repository** CRAN

**Date/Publication** 2025-11-24 14:10:02 UTC

## **Contents**

	all_preserved							 								2
	apply_niche							 								3
	apply_taphonomy .							 								5
	bounded_niche							 								6
	discrete_gradient .							 								7
	discrete_niche							 								8
	gradient_from_data							 								9
	last_occ							 								10
	ornstein_uhlenbeck							 								11
	ornstein_uhlenbeck	_sl .						 								12
	p3							 								14
	p3_var_rate							 								15
	perfect_preservation	ı						 								16
	plot.pre_paleoTS .							 								17
	prob_remove							 								17
	random_walk							 								18
	random_walk_sl							 								19
	range_offset							 								20
	reduce_to_paleoTS							 								22
	rej_samp							 								23
	scenarioA							 								24
	snd_niche							 								25
	stasis							 								26
	stasis_sl							 								27
	strict_stasis_sl							 								28
	thin							 								29
	trivial_gradient							 								30
	trivial_niche							 								31
Index																32
all_	preserved	Inde	estru	ıctil	ole j	foss	sils									

## Description

Models perfect taphonomic conditions. Mainly used as default input to last\_occ and range\_offset or passed as pres\_potential argument to apply\_taphonomy.

apply\_niche 3

## Usage

```
all_preserved(x)
```

#### **Arguments**

Х

taphonomic conditions at which the preservation probability is evaluated

#### Value

A vector of the same length as x with all entries replaced by 1.

#### See Also

```
last_occ(), range_offset(), perfect_preservation(), and apply_taphonomy()
```

## **Examples**

```
x = p3(\text{rate} = 10, 0, 1) \text{ # model fossils}

y = apply\_taphonomy(x, pres\_potential = all\_preserved, ctc = perfect\_preservation)

all(x == y) \text{ # true, all fossils are preserved}
```

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**

Х

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.

4 apply\_niche

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, trivial\_niche() to model organisms without niche specifications
- 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

```
### 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 = "1",
 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
```

apply\_taphonomy 5

```
### 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)) \mid > \# 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

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.

6 bounded\_niche

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.
- perfect\_preservation() and all\_preserved() to model perfect preservation of fossils

#### **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

discrete\_gradient 7

## See Also

- snd\_niche() for an alternative niche model
- discrete\_niche() for defining niches based on discrete categories
- trivial\_niche() to model organisms without niche specifications
- 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 = "1",
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

8 discrete\_niche

## 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 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.

- 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

gradient\_from\_data 9

## **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 %)
```

gradient\_from\_data

gradient or taphonomic conditions from data

## **Description**

Constructs a gradient or taphonomic conditions from a dataframe or list. If x is a dataframe, the first two columns are used, if x is a list the first two elements are used. The first entry is used as time/stratigraphic position, and the second as gradient/recovery probability.

## Usage

```
gradient_from_data(x)
```

#### Arguments

Χ

list or data frame

#### Value

a function for usage as gradient or taphonomic conditions, which can be passed to apply\_taphonomy or apply\_niche

10 last\_occ

## **Examples**

```
# water depth 2 km offshore as gradient
t = scenarioA$t_myr
l = list(t, scenarioA$wd_m[,"2km"])
f = gradient_from_data(l)
plot(t, f(t), type = "1")
```

last\_occ

last occurrence of taxon

## **Description**

Determines the time and position of a taxons last occurrence as a function of time of extinction, fossil abundance, stratigraphy, ecology, and taphonomy. Effectively a high-level wrapper around p3/p3\_var\_rate, apply\_niche, apply\_taphonomy and time\_to\_strat/strat\_to\_time from the admtools package.

#### Usage

```
last_occ(
    t_ext,
    rate,
    adm,
    niche = trivial_niche,
    gc = trivial_gradient,
    niche_domain = "time",
    pres_potential = all_preserved,
    ctc = perfect_preservation,
    taphonomy_domain = "strat"
)
```

#### **Arguments**

t\_ext true time of extinction rate either positive number or a function. If a number, rate of fossil occurrences passed to p3, if a function passed as rate function to p3\_var\_rate adm age-depth model niche model, by default the trivial niche niche gradient change for the niche mode, by default the trivial gradient. See ?apply\_niche gc "time" or "strat" - in which domain should the niche model be applied? niche\_domain preservation potential of fossils, by default all preserved (no taphonomic effects) pres\_potential change in taphonomic conditions, by default no change in conditions. See ctc ?apply\_taphonomy

taphonomy\_domain

"time" or "strat" - in which domain should the taphonomic effects be applied?

ornstein\_uhlenbeck 11

#### Value

a named vector with two entries:

- "h": stratigraphic position of the last occurrence
- "t": time of the last occurrence

## See Also

```
range_offset() to quantify biostratigraphic precision
```

## **Examples**

```
# last occurrences 2 km from shore
h = scenarioA$h_m[,"2km"]
adm = admtools::tp_to_adm(t = scenarioA$t_myr, h = h)
l_occ = last_occ(t_ext = 1.8, rate = 5, adm = adm)
l_occ # show timing and position of last occurrences
```

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(1, type = "1")  
l2 = ornstein_uhlenbeck(t, y0 = 0) # start in optimum lines(12$t, 12$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

```
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
```

14 *p3* 

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
```

```
# 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 15

p3_var_rate simulate variable rate Poisson point process	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
у	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.

perfect\_preservation

## **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
```

## Description

Models perfect taphonomic conditions. Mainly used as default input to last\_occ and range\_offset or passed as ctc argument to apply\_taphonomy.

## Usage

```
perfect_preservation(x)
```

## **Arguments**

Х

time/stratigraphic position at which the taphonomic conditions are determined

#### Value

A vector of the same length as x with all entries replaced by 1.

## See Also

```
last_occ(), range_offset(), all_preserved(), and apply_taphonomy()
```

```
x = p3(\text{rate} = 10, 0, 1) \text{ # model fossils}

y = apply\_taphonomy(x, pres\_potential = all\_preserved, ctc = perfect\_preservation)

all(x == y) \text{ # true, all fossils are preserved}
```

plot.pre\_paleoTS 17

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

18 random\_walk

## 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 a which the random walk is evaluated
sigm	positive number, variance parameter
mu	number, directionality parameter
y0	number, starting value (value of the random walk at the first entry of t)

19 random\_walk\_sl

## 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\_s1() 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)
1 = random_walk(t, sigma = 3) # high variability, no direction
plot(1, type = "1")
12 = random_walk(t, mu = 1) # low variabliity, increasing trend
lines(12\$t, 12\$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

numeric vector with strictly increasing elements, can be heterodistant. Times at which the random walk is evaluated positive number, variance parameter

sigma

20 range\_offset

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
```

range\_offset range offset of taxon

## Description

Determines temporal and stratigraphic range offset, a measure of biostratigraphic precision, as a function of time of extinction, fossil abundance, stratigraphy, ecology, and taphonomy. Effectively a high-level wrapper around last\_occ, p3/p3\_var\_rate, apply\_niche, apply\_taphonomy and time\_to\_strat/strat\_to\_time from the admtools package.

range\_offset 21

## Usage

```
range_offset(
  t_ext,
  rate,
  adm,
  niche = trivial_niche,
  gc = trivial_gradient,
  niche_domain = "time",
  pres_potential = all_preserved,
  ctc = perfect_preservation,
  taphonomy_domain = "strat"
)
```

## **Arguments**

true time of extinction						
either positive number or a function. If a number, rate of fossil occurrences passed to p3, if a function passed as rate function to p3_var_rate						
age-depth model						
niche model, by default the trivial niche						
$gradient\ change\ for\ the\ niche\ mode,\ by\ default\ the\ trivial\ gradient.\ See\ \verb§?apply\_niche§$						
"time" or "strat" - in which domain should the niche model be applied?						
preservation potential of fossils, by default all preserved (no taphonomic effects)						
change in taphonomic conditions, by default no change in conditions. See ?apply_taphonomy						
taphonomy_domain						
"time" or "strat" - in which domain should the taphonomic effects be applied?						

## Value

a named vector with two entries:

- "h": distance between last occurrence and true height of extinction
- "t": time between last occurrence and true time of extinction

## See Also

last\_occ() determine position/time of last occurrence of taxon

```
# last occurrences 2 km from shore
h = scenarioA$h_m[,"2km"]
adm = admtools::tp_to_adm(t = scenarioA$t_myr, h = h)
offset = range_offset(t_ext = 1.8, rate = 5, adm = adm)
offset # show timing and position of last occurrences
```

22 reduce\_to\_paleoTS

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

```
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 23

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.

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

24 scenarioA

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 25

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

- apply\_niche() for usage of the returned function
- bounded\_niche() for another niche model
- trivial\_niche() to model organisms without niche specifications
- 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

26 stasis

## **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

- 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.

stasis\_sl 27

## **Examples**

```
library("admtools") # required for plotting of results t = seq(0, 1, by = 0.01)  
l = stasis(t)  
plot(1, type = "1") # 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 trait value

sd strictly positive number, standard deviation of traits around the mean

intrapop\_var intrapopulation variance, determines how much specimens from the same pop-

ulation 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.

- 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

28 strict\_stasis\_sl

## **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 trait value

intrapop\_var intrapopulation variance, determines how much specimens from the same pop-

ulation 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.

thin 29

## 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.

30 trivial\_gradient

## **Examples**

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

trivial\_gradient

model absence of environmental gradients

## **Description**

Models a constant gradient with value 1. Mainly used as default input to last\_occ and range\_offset.

## Usage

```
trivial_gradient(x)
```

## Arguments

Х

time/stratigraphic position at which the gradient is determined

## Value

A vector of the same length as x with all entries replaced by 1.

## See Also

```
last_occ(), range_offset(), trivial_niche(), and apply_niche()
```

```
x = p3(rate = 10, from = 0, to = 1) # model fossil occurrences
# apply trivial niche model
y = apply_niche(x, niche_def = trivial_niche, gc = trivial_gradient)
all(x == y) # true, no fossils were removed
```

trivial\_niche 31

trivial\_niche

trivial niche model

## **Description**

Models a trivial niche, meaning the niche of a taxon that has no environmental preferences. Mainly used as default input to last\_occ and range\_offset. When passed to apply\_niche, this will effectively result in no niche model being applied.

## Usage

```
trivial_niche(x)
```

## **Arguments**

Х

gradient value at which the niche is evaluated

#### Value

A vector of the same length as x with all entries replaced by 1.

## See Also

```
last_occ(), range_offset(), trivial_gradient() and apply_niche()
```

```
x = p3(rate = 10, from = 0, to = 1) # model fossil occurrences
# apply trivial niche model
y = apply_niche(x, niche_def = trivial_niche, gc = trivial_gradient)
all(x == y) # true, no fossils were removed
```

# **Index**

* datasets scenarioA, 24	random_walk_s1, 19, 22 random_walk_s1(), <i>13</i> , <i>19</i> , <i>27</i> , <i>29</i>
all_preserved, 2 all_preserved(), 6, 16 apply_niche, 3 apply_niche(), 6–8, 18, 25, 29–31 apply_taphonomy, 5 apply_taphonomy(), 3, 4, 16, 18, 29 bounded_niche, 6 bounded_niche(), 4, 8, 25 discrete_gradient, 7 discrete_gradient(), 4, 8, 25 discrete_niche, 8	<pre>range_offset, 20 range_offset(), 3, 11, 16, 30, 31 reduce_to_paleoTS, 22 reduce_to_paleoTS(), 13, 17, 20, 27, 29 rej_samp, 23 rej_samp(), 15  scenarioA, 24 snd_niche, 25 snd_niche(), 4, 7, 8 stasis, 26 stasis(), 12, 19, 27 stasis_sl, 27</pre>
discrete_niche(), 4, 7, 8, 25	stasis_sl(), <i>13</i> , <i>20</i> , <i>22</i> , <i>26</i> , <i>29</i> strict_stasis_sl, <i>22</i> , <i>28</i> strict_stasis_sl(), <i>13</i> , <i>20</i> , <i>27</i>
gradient_from_data, 9  last_occ, 10  last_occ(), 3, 16, 21, 30, 31  ornstein_uhlenbeck, 11 ornstein_uhlenbeck(), 13, 19, 26 ornstein_uhlenbeck_sl, 12 ornstein_uhlenbeck_sl, 12 ornstein_uhlenbeck_sl(), 12, 20, 22, 27, 29	thin, 29 thin(), 4, 6 trivial_gradient, 30 trivial_gradient(), 31 trivial_niche, 31 trivial_niche(), 4, 7, 25, 30
p3, 14 p3(), 15 p3_var_rate, 15 p3_var_rate(), 14, 23 perfect_preservation, 16 perfect_preservation(), 3, 6 plot.pre_paleoTS, 17 prob_remove, 17 prob_remove(), 4, 6	
random_walk, 18 random_walk(), <i>12</i> , <i>20</i> , <i>26</i>	