Package 'shar'

November 16, 2022

Type Package

Title Species-Habitat Associations

Version 2.0.2

Maintainer Maximilian H.K. Hesselbarth <mhk.hesselbarth@gmail.com>

Description Analyse species-habitat associations in R. Therefore, information about the location of the species (as a point pattern) is needed together with environmental conditions (as a categorical raster). To test for significance habitat associations, one of the two components is randomized. Methods are mainly based on Plotkin et al. (2000) <doi:10.1006/jtbi.2000.2158> and Harms et al. (2001) <doi:10.1111/j.1365-2745.2001.00615.x>.

License GPL (>= 3)

URL https://r-spatialecology.github.io/shar/

BugReports https://github.com/r-spatialecology/shar/issues/

Depends R (>= 3.1.0)

Imports classInt, graphics, grDevices, methods, spatstat.explore, spatstat.model, spatstat.geom, spatstat.random, stats, terra, utils

RoxygenNote 7.2.1

Suggests covr, dplyr, knitr, rmarkdown, spatstat (>= 2.0-0), testthat (>= 3.0.0)

VignetteBuilder knitr

Encoding UTF-8

LazyData true

NeedsCompilation no

Author Maximilian H.K. Hesselbarth [aut, cre] (<https://orcid.org/0000-0003-1125-9918>), Marco Sciaini [aut] (<https://orcid.org/0000-0002-3042-5435>), Zeke Marshall [ctb] (<https://orcid.org/0000-0001-9260-7827>), Thomas Etherington [ctb] (<https://orcid.org/0000-0002-3187-075X>)

Repository CRAN

Date/Publication 2022-11-16 15:40:02 UTC

R topics documented:

`calculate_energy` *calculate_energy*

Description

Calculate mean energy

Usage

```
calculate_energy(
  pattern,
  weights = c(0.5, 0.5),
  return_mean = FALSE,
  comp_fast = 1000,
  verbose = TRUE
)
```

Arguments

pattern	List with reconstructed patterns.
weights	Vector with weights used to calculate energy. The first number refers to Gest(r), the second number to pcf(r).
return_mean	Logical if the mean energy is returned.
comp_fast	Integer with threshold at which summary functions are estimated in a computational fast way.
verbose	Logical if progress report is printed.

Details

The function calculates the mean energy (or deviation) between the observed pattern and all reconstructed patterns (for more information see Tscheschel & Stoyan (2006) or Wiegand & Moloney (2014)). The pair correlation function and the nearest neighbour distance function are used to describe the patterns. For large patterns `comp_fast = TRUE` decreases the computational demand, because no edge correction is used and the pair correlation function is estimated based on Ripley's K-function. For more information see `estimate_pcf_fast`.

Value

vector

References

Kirkpatrick, S., Gelatt, C.D.Jr., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. <https://doi.org/10.1126/science.220.4598.671>

Tscheschel, A., Stoyan, D., 2006. Statistical reconstruction of random point patterns. Computational Statistics and Data Analysis 51, 859–871. <https://doi.org/10.1016/j.csda.2005.09.007>

Wiegand, T., Moloney, K.A., 2014. Handbook of spatial point-pattern analysis in ecology. Chapman and Hall/CRC Press, Boca Raton. ISBN 978-1-4200-8254-8

See Also

`plot_energy`
`reconstruct_pattern`
`fit_point_process`

Examples

```
pattern_random <- fit_point_process(species_a, n_random = 19)
calculate_energy(pattern_random)
calculate_energy(pattern_random, return_mean = TRUE)

## Not run:
marks_sub <- spatstat.geom::subset.ppp(species_a, select = dbh)
marks_recon <- reconstruct_pattern_marks(pattern_random$randomized[[1]], marks_sub,
n_random = 19, max_runs = 1000)
calculate_energy(marks_recon, return_mean = FALSE)
```

```
## End(Not run)
```

`classify_habitats` *classify_habitats*

Description

Classify habitats

Usage

```
classify_habitats(raster, return_breaks = FALSE, ...)
```

Arguments

`raster`	SpatRaster with continuous environmental values.
`return_breaks`	Logical if breaks should be returned as well.
`...`	Arguments passed on to `classIntervals`.

Details

Classifies a SpatRaster from the `raster` packages with continuous values into n discrete classes. The `cut` function used to classify the raster, uses `include.lowest = TRUE`.

For more information about the classification methods, see `classIntervals` from the `classInt` package and/or the provided References. The help page of `classIntervals` also includes further possible arguments to find breaks (e.g., different styles, number of classes, fixed breaks, etc.).

Value

SpatRaster

References

Armstrong, M.P., Xiao, N., Bennett, D.A., 2003. Using genetic algorithms to create multicriteria class intervals for choropleth maps. Annals of the Association of American Geographers 93, 595–623. <https://doi.org/10.1111/1467-8306.9303005>

Dent, B.D., 1999. Cartography: Thematic map design, 5th ed. WCB/McGraw-Hill, Boston, USA. ISBN 978-0-697-38495-9

Fisher, W.D., 1958. On grouping for maximum homogeneity. Journal of the American Statistical Association 53, 789–798. <https://doi.org/10.1080/01621459.1958.10501479>

Jenks, G.F., Caspall, F.C., 1971. Error in choroplethic maps: Definition, measurement, reduction. Annals of the Association of American Geographers 61, 217–244. <https://doi.org/10.1111/j.1467-8306.1971.tb00779.x>

Jiang, B., 2013. Head/tail breaks: A new classification scheme for data with a heavy-tailed distribution. The Professional Geographer 65, 482-494. <https://doi.org/10.1080/00330124.2012.700499>

Slocum, T.A., McMaster, R.B., Kessler, F.C., Howard, H.H., 2009. Thematic cartography and geovisualization, 3rd ed. ed, Prentice Hall Series in Geographic Information Science. Pearson Prentice Hall, Upper Saddle River, USA. ISBN 978-0-13-229834-6

Wand, M. P., 1995. Data-based choice of histogram binwidth. The American Statistician 51, 59-64. <https://doi.org/10.1080/00031305.1997.10473591>

See Also

classIntervals

Examples

```
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")

landscape_classified <- classify_habitats(terra::rast(landscape), style = "fixed",
fixedBreaks = c(0, 0.25, 0.75, 1.0), return_breaks = TRUE)
```

---

estimate_pcf_fast *estimate_pcf_fast*

Description

Fast estimation of the pair correlation function

Usage

```
estimate_pcf_fast(pattern, ...)
```

Arguments

pattern	ppp object with point pattern.
...	Arguments passed down to link{Kest} or pcf.fv.

Details

The functions estimates the pair correlation functions based on an estimation of Ripley's K-function. This makes it computationally faster than estimating the pair correlation function directly. It is a wrapper around Kest and pcf.fv.

Value

fv.object

References

Chiu, S.N., Stoyan, D., Kendall, W.S., Mecke, J., 2013. Stochastic geometry and its applications, 3rd ed, Wiley Series in Probability and Statistics. John Wiley & Sons Inc, Chichester, UK. ISBN 978-0-470-66481-0

Ripley, B.D., 1977. Modelling spatial patterns. Journal of the Royal Statistical Society. Series B (Methodological) 39, 172–192. <https://doi.org/10.1111/j.2517-6161.1977.tb01615.x>

Stoyan, D., Stoyan, H., 1994. Fractals, random shapes and point fields. John Wiley & Sons, Chichester. ISBN 978-0-471-93757-9

See Also

Kest
pcf.fv

Examples

```
pcf_species_b <- estimate_pcf_fast(species_a)
```

`fit_point_process` *fit_point_process*

Description

Fit point process to randomize data

Usage

```
fit_point_process(
  pattern,
  n_random = 1,
  process = "poisson",
  return_input = TRUE,
  simplify = FALSE,
  verbose = TRUE
)
```

Arguments

`pattern`	ppp object with point pattern
`n_random`	Integer with number of randomizations.
`process`	Character specifying which point process model to use. Either `"poisson"` or `"cluster"`.
`return_input`	Logical if the original input data is returned.
`simplify`	Logical if only pattern will be returned if `n_random = 1` and `return_input = FALSE`.
`verbose`	Logical if progress report is printed.

Details

The functions randomizes the observed point pattern by fitting a point process to the data and simulating `n_random` patterns using the fitted point process. It is possible to choose between a Poisson process or a Thomas cluster process model. For more information about the point process models, see e.g. Wiegand & Moloney (2014).

Value

rd_pat

References

Plotkin, J.B., Potts, M.D., Leslie, N., Manokaran, N., LaFrankie, J.V., Ashton, P.S., 2000. Species-area curves, spatial aggregation, and habitat specialization in tropical forests. Journal of Theoretical Biology 207, 81–99. <https://doi.org/10.1006/jtbi.2000.2158>

Wiegand, T., Moloney, K.A., 2014. Handbook of spatial point-pattern analysis in ecology. Chapman and Hall/CRC Press, Boca Raton. ISBN 978-1-4200-8254-8

Examples

```
pattern_fitted <- fit_point_process(pattern = species_a, n_random = 39)
```

---

gamma_test *Gamma test*

Description

Randomized data for species b using the gamma test.

Usage

```
gamma_test
```

Format

rd_pat object.

landscape *Example landscape (random cluster neutral landscape model).*

Description

An example map to show landscapetools functionality generated with the `NLMR::nlm_fbm()` algorithm.

Usage

```
landscape
```

Format

A SpatRaster object.

Source

Simulated neutral landscape model with R. https://github.com/ropensci/NLMR/

list_to_randomized *list_to_randomized*

Description

Convert list to rd_* object.

Usage

```
list_to_randomized(list, observed = NULL)
```

Arguments

`list`	List
`observed`	Observed

Details

Convert list of randomized point pattern or raster layer to a rd_* object that can be used with all functions of the package. The main purpose of this utility function is to allow an easy parallelization of the randomization approach.

For more information, please see the "Parallelization" article.

Value

rd_pat, rd_ras

See Also

randomize_raster
translate_raster
reconstruct_pattern

Examples

```
## Not run:
fit_list <- lapply(X = 1:39, FUN = function(i) {fit_point_process(pattern = species_a,
n_random = 1, simplify = TRUE, return_input = FALSE, verbose = FALSE)})

list_to_randomized(list = fit_list, observed = species_a)

## End(Not run)
```

---

pack_randomized *pack_randomized*

Description

Save randomized raster object

Usage

```
pack_randomized(raster)
```

Arguments

raster	rd_ras object with randomized raster.

Details

Because of how SpatRaster are saved (need to be packed), this function wraps all raster objects and prepares them for saving first. For further details, see wrap.

Value

rd_ras

See Also

unpack_randomized wrap

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
landscape_random <- randomize_raster(landscape_classified, n_random = 3)
x <- pack_randomized(raster = landscape_random)

## End(Not run)
```

`plot.rd_mar` *plot.rd_mar*

Description

Plot method for rd_pat object

Usage

```
## S3 method for class 'rd_mar'
plot(
  x,
  what = "sf",
  n = NULL,
  probs = c(0.025, 0.975),
  comp_fast = 1000,
  ask = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

`x`	rd_mar object with randomized patterns.
`what`	Character specifying to plot summary functions of point patterns (`what = "sf"`) or actual patterns (`what = "pp"`).
`n`	Integer with number or vector of ids of randomized pattern to plot. See Details section for more information.
`probs`	Vector with quantiles of randomized data used for envelope construction.
`comp_fast`	Integer with threshold at which summary functions are estimated in a computational fast way.
`ask`	Logical if the user is asked to press <RETURN> before second summary function is plotted (only used if `what = "sf"`).
`verbose`	Logical if progress report is printed.
`...`	Not used.

Details

The function plots the pair correlation function and the nearest neighbour function of the observed pattern and the reconstructed patterns (as "simulation envelopes"). For large patterns `comp_fast = TRUE` decreases the computational demand because no edge correction is used and the pair correlation function is estimated based on Ripley's K-function. For more information see `estimate_pcf_fast`.

It is also possible to plot `n` randomized patterns and the observed pattern using `what = "pp"`. If `n` is a single number, `n` randomized patterns will be sampled to plot. If `n` is a vector, the corresponding patterns will be plotted.

Value

void

See Also

`reconstruct_pattern`
`fit_point_process`

Examples

```
## Not run:
pattern_recon <- reconstruct_pattern(species_a, n_random = 1, max_runs = 1000,
simplify = TRUE, return_input = FALSE)
marks_sub <- spatstat.geom::subset.ppp(species_a, select = dbh)
marks_recon <- reconstruct_pattern_marks(pattern_recon, marks_sub,
n_random = 19, max_runs = 1000)
plot(marks_recon)

## End(Not run)
```

`plot.rd_pat` *plot.rd_pat*

Description

Plot method for rd_pat object

Usage

```
## S3 method for class 'rd_pat'
plot(
  x,
  what = "sf",
  n = NULL,
  probs = c(0.025, 0.975),
  comp_fast = 1000,
```

```
  ask = TRUE,
  verbose = TRUE,
  ...
)
```

Arguments

x	rd_pat object with randomized patterns.
what	Character specifying to plot summary functions of point patterns (what = "sf") or actual patterns (what = "pp").
n	Integer with number or vector of ids of randomized pattern to plot. See Details section for more information.
probs	Vector with quantiles of randomized data used for envelope construction.
comp_fast	Integer with threshold at which summary functions are estimated in a computational fast way.
ask	Logical if the user is asked to press <RETURN> before second summary function is plotted (only used if what = "sf").
verbose	Logical if progress report is printed.
...	Not used.

Details

The function plots the pair correlation function and the nearest neighbour function of the observed pattern and the reconstructed patterns (as "simulation envelopes"). For large patterns comp_fast = TRUE decreases the computational demand because no edge correction is used and the pair correlation function is estimated based on Ripley's K-function. For more information see estimate_pcf_fast.

It is also possible to plot n randomized patterns and the observed pattern using what = "pp". If n is a single number, n randomized patterns will be sampled to plot. If n is a vector, the corresponding patterns will be plotted.

Value

void

See Also

reconstruct_pattern
fit_point_process

Examples

```
## Not run:
pattern_random <- fit_point_process(species_a, n_random = 39)
plot(pattern_random)

pattern_recon <- reconstruct_pattern(species_b, n_random = 19,
max_runs = 1000, method = "hetero")
plot(pattern_recon)
```

```
## End(Not run)
```

---

plot.rd_ras *plot.rd_ras*

Description

Plot method for rd_ras object

Usage

```
## S3 method for class 'rd_ras'
plot(x, n = NULL, col, verbose = TRUE, nrow, ncol, ...)
```

Arguments

x	rd_ras object with randomized raster.
n	Integer with number or vector of ids of randomized raster to plot. See Details section for more information.
col	Vector with color palette used for plotting.
verbose	Logical if messages are printed.
nrow, ncol	Integer with number of rows and columns of plot grid.
...	Not used.

Details

Function to plot randomized raster. If n is a single number, n randomized raster will be sampled to plot. If n is a vector, the corresponding raster will be plotted. col, nrow, ncol are passed to plot.

Value

void

See Also

randomize_raster
translate_raster

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
landscape_random <- randomize_raster(landscape_classified, n_random = 19)
plot(landscape_random)

## End(Not run)
```

plot_energy *plot_energy*

Description

Plot energy of pattern reconstruction

Usage

```
plot_energy(pattern, col = NULL)
```

Arguments

pattern	rd_pat or rd_mar object with randomized patterns.
col	Vector with colors. Must be the same length as n_random.

Details

The function plots the decrease of the energy over time, i.e. the iterations. This can help to identify if the chosen max_runs for the reconstruction were sufficient. The pattern object must have been created using reconstruct_pattern_* .

Value

void

See Also

reconstruct_pattern
fit_point_process

Examples

```
## Not run:
pattern_recon <- reconstruct_pattern(species_a, n_random = 3, max_runs = 1000)
plot_energy(pattern_recon)

marks_sub <- spatstat.geom::subset.ppp(species_a, select = dbh)
marks_recon <- reconstruct_pattern_marks(pattern_recon$randomized[[1]], marks_sub,
n_random = 1, max_runs = 1000)
plot_energy(marks_recon)

## End(Not run)
```

print.rd_mar *print.rd_mar*

Description

Print method for rd_mar object

Usage

```
## S3 method for class 'rd_mar'
print(x, digits = 4, ...)
```

Arguments

x	rd_mar object with randomized patterns.
digits	Integer with number of decimal places (round) to be printed.
...	Arguments passed to cat.

Details

Printing method for random patterns created with reconstruct_pattern_marks.

Value

void

See Also

reconstruct_pattern_marks

Examples

```
## Not run:
pattern_recon <- reconstruct_pattern(species_a, n_random = 1, max_runs = 1000,
simplify = TRUE, return_input = FALSE)
marks_sub <- spatstat.geom::subset.ppp(species_a, select = dbh)
marks_recon <- reconstruct_pattern_marks(pattern_recon, marks_sub,
n_random = 19, max_runs = 1000)
print(marks_recon)

## End(Not run)
```

print.rd_pat *print.rd_pat*

Description

Print method for rd_pat object

Usage

```
## S3 method for class 'rd_pat'
print(x, digits = 4, ...)
```

Arguments

x	rd_pat object with randomized patterns.
digits	Integer with number of decimal places (round).
...	Arguments passed to cat.

Details

Printing method for random patterns created with reconstruct_pattern_*.

Value

void

See Also

reconstruct_pattern
fit_point_process

Examples

```
pattern_random <- fit_point_process(species_a, n_random = 199)
print(pattern_random)

## Not run:
pattern_recon <- reconstruct_pattern(species_b, n_random = 19, max_runs = 1000,
method = "hetero")
print(pattern_recon)

## End(Not run)
```

print.rd_ras *print.rd_ras*

Description

Print method for rd_ras object

Usage

```
## S3 method for class 'rd_ras'
print(x, ...)
```

Arguments

x	rd_ras object with randomized raster.
...	Arguments passed to cat.

Details

Printing method for random patterns created with randomize_raster or translate_raster.

Value

void

See Also

randomize_raster
translate_raster

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
landscape_random <- randomize_raster(landscape_classified, n_random = 19)

print(landscape_random)

## End(Not run)
```

`randomize_raster` *randomize_raster*

Description

Randomized-habitats procedure

Usage

```
randomize_raster(
  raster,
  n_random = 1,
  directions = 4,
  return_input = TRUE,
  simplify = FALSE,
  verbose = TRUE
)
```

Arguments

`raster`	SpatRaster with discrete habitat classes.
`n_random`	Integer with number of randomizations.
`directions`	Interger with cells neighbourhood rule: 4 (rook's case), 8 (queen's case).
`return_input`	Logical if the original input data is returned.
`simplify`	Logical if only the raster will be returned if `n_random = 1` and `return_input = FALSE`.
`verbose`	Logical if progress report is printed.

Details

The function randomizes a habitat map with discrete classes (as SpatRaster) as proposed by Harms et al. (2001) as "randomized-habitats procedure". The algorithm starts with an empty habitat map and starts to assign random neighbouring cells to each habitat (in increasing order of abundance in observed map). We modified the procedure slightly by increasing a probability to jump to a non-neighbouring cell as the current patch becomes larger.

In case the SpatRaster contains NA cells, this needs to be reflected in the observation window of the point pattern as well (i.e., no point locations possible in these areas).

Value

rd_ras

References

Harms, K.E., Condit, R., Hubbell, S.P., Foster, R.B., 2001. Habitat associations of trees and shrubs in a 50-ha neotropical forest plot. Journal of Ecology 89, 947–959. <https://doi.org/10.1111/j.1365-2745.2001.00615.x>

See Also

translate_raster

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
landscape_random <- randomize_raster(landscape_classified, n_random = 19)

## End(Not run)
```

---

random_walk *Random walk*

Description

Randomization of the landscape using the habitat randomization algorithm.

Usage

```
random_walk
```

Format

rd_ras object.

---

reconstruction *Reconstruction*

Description

Randomized data for species b using pattern reconstruction.

Usage

```
reconstruction
```

Format

rd_pat object.

reconstruct_pattern *reconstruct_pattern*

Description

Pattern reconstruction

Usage

```
reconstruct_pattern(
  pattern,
  method = "homo",
  n_random = 1,
  e_threshold = 0.01,
  max_runs = 1000,
  no_change = Inf,
  annealing = 0.01,
  comp_fast = 1000,
  n_points = NULL,
  window = NULL,
  weights = c(0.5, 0.5),
  r_length = 250,
  return_input = TRUE,
  simplify = FALSE,
  verbose = TRUE,
  plot = FALSE
)
```

Arguments

pattern	ppp object with pattern.
method	Character with specifying the method. Either "homo", "cluster" or "hetero".
n_random	Integer with number of randomizations.
e_threshold	Double with minimum energy to stop reconstruction.
max_runs	Integer with maximum number of iterations if e_threshold is not reached.
no_change	Integer with number of iterations at which the reconstruction will stop if the energy does not decrease.
annealing	Double with probability to keep relocated point even if energy did not decrease.
comp_fast	Integer with threshold at which summary functions are estimated in a computational fast way.
n_points	Integer with number of points to be simulated.
window	owin object with window of simulated pattern.
weights	Vector with weights used to calculate energy. The first number refers to Gest(r), the second number to pcf(r).

r_length	Integer with number of intervals from r=0 to r=rmax for which the summary functions are evaluated.
return_input	Logical if the original input data is returned.
simplify	Logical if only pattern will be returned if n_random=1 and return_input=FALSE.
verbose	Logical if progress report is printed.
plot	Logical if pcf(r) function is plotted and updated during optimization.

Details

The functions randomizes the observed pattern by using pattern reconstruction as described in Tscheschel & Stoyan (2006) and Wiegand & Moloney (2014). The algorithm shifts a point to a new location and keeps the change only, if the deviation between the observed and the reconstructed pattern decreases. The pair correlation function and the nearest neighbour distance function are used to describe the patterns.

For large patterns (n > comp_fast) the pair correlation function can be estimated from Ripley's K-function without edge correction. This decreases the computational time. For more information see estimate_pcf_fast.

The reconstruction can be stopped automatically if for n steps the energy does not decrease. The number of steps can be controlled by no_change and is set to no_change = Inf as default to never stop automatically.

The weights must be 0 < sum(weights) <= 1. To weight both summary functions identical, use weights = c(0.5, 0.5).

spatstat sets r_length to 513 by default. However, a lower value decreases the computational time, while increasing the "bumpiness" of the summary function.

The arguments n_points and window are used for method="homo" only.

method="homo":: The algorithm starts with a random pattern.

method="cluster":: The algorithm starts with a random but clustered pattern.

method="hetero":: The algorithm starts with a random but heterogeneous pattern.

Value

rd_pat

References

Kirkpatrick, S., Gelatt, C.D.Jr., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. <https://doi.org/10.1126/science.220.4598.671>

Tscheschel, A., Stoyan, D., 2006. Statistical reconstruction of random point patterns. Computational Statistics and Data Analysis 51, 859–871. <https://doi.org/10.1016/j.csda.2005.09.007>

Wiegand, T., Moloney, K.A., 2014. Handbook of spatial point-pattern analysis in ecology. Chapman and Hall/CRC Press, Boca Raton. ISBN 978-1-4200-8254-8

See Also

calculate_energy
reconstruct_pattern_marks

Examples

```
## Not run:
pattern_recon <- reconstruct_pattern(species_b, n_random = 19, max_runs = 1000)

## End(Not run)
```

---

reconstruct_pattern_marks
reconstruct_pattern_marks

Description

Pattern reconstruction of marked pattern

Usage

```
reconstruct_pattern_marks(
  pattern,
  marked_pattern,
  n_random = 1,
  e_threshold = 0.01,
  max_runs = 10000,
  no_change = Inf,
  annealing = 0.01,
  r_length = 250,
  return_input = TRUE,
  simplify = FALSE,
  verbose = TRUE,
  plot = FALSE
)
```

Arguments

pattern	ppp object with pattern.
marked_pattern	ppp object with marked pattern. See Details section for more information.
n_random	Integer with number of randomizations.
e_threshold	Double with minimum energy to stop reconstruction.
max_runs	Integer with maximum number of iterations if e_threshold is not reached.
no_change	Integer with number of iterations at which the reconstruction will stop if the energy does not decrease.

annealing	Double with probability to keep relocated point even if energy did not decrease.
r_length	Integer with number of intervals from r = 0 to r = rmax for which the summary functions are evaluated.
return_input	Logical if the original input data is returned.
simplify	Logical if only pattern will be returned if n_random = 1 and return_input = FALSE.
verbose	Logical if progress report is printed.
plot	Logical if pcf(r) function is plotted and updated during optimization.

Details

The function randomizes the numeric marks of a point pattern using pattern reconstruction as described in Tscheschel & Stoyan (2006) and Wiegand & Moloney (2014). Therefore, an unmarked as well as a marked pattern must be provided. The unmarked pattern must have the spatial characteristics and the same observation window and number of points as the marked one (see reconstruct_pattern_* or fit_point_process). Marks must be numeric because the mark-correlation function is used as summary function. Two randomly chosen marks are switch each iterations and changes only kept if the deviation between the observed and the reconstructed pattern decreases.

spatstat sets r_length to 513 by default. However, a lower value decreases the computational time while increasing the "bumpiness" of the summary function.

Value

rd_mar

References

Kirkpatrick, S., Gelatt, C.D.Jr., Vecchi, M.P., 1983. Optimization by simulated annealing. Science 220, 671–680. <https://doi.org/10.1126/science.220.4598.671>

Tscheschel, A., Stoyan, D., 2006. Statistical reconstruction of random point patterns. Computational Statistics and Data Analysis 51, 859–871. <https://doi.org/10.1016/j.csda.2005.09.007>

Wiegand, T., Moloney, K.A., 2014. Handbook of spatial point-pattern analysis in ecology. Chapman and Hall/CRC Press, Boca Raton. ISBN 978-1-4200-8254-8

See Also

fit_point_process
reconstruct_pattern

Examples

```
## Not run:
pattern_recon <- reconstruct_pattern(species_a, n_random = 1, max_runs = 1000,
simplify = TRUE, return_input = FALSE)
marks_sub <- spatstat.geom::subset.ppp(species_a, select = dbh)
marks_recon <- reconstruct_pattern_marks(pattern_recon, marks_sub,
n_random = 19, max_runs = 1000)
```

```
## End(Not run)
```

`results_habitat_association` *results_habitat_association*

Description

Results habitat association

Usage

```
results_habitat_association(
  pattern,
  raster,
  significance_level = 0.05,
  breaks = NULL,
  digits = NULL,
  verbose = TRUE
)
```

Arguments

`pattern`	ppp object with original point pattern data or rd_pat or rd_mar object with randomized point pattern.
`raster`	SpatRaster with original discrete habitat data or rd_ras object with randomized environmental data.
`significance_level`	Double with significance level.
`breaks`	Vector with breaks of habitat classes.
`digits`	Integer with digits used during rounding.
`verbose`	Logical if messages should be printed.

Details

The functions shows significant habitat associations by comparing the number of points within a habitat between the observed data and randomized data as described in Plotkin et al. (2000) and Harms et al. (2001). Significant positive or associations are present if the observed count in a habitat is above or below a certain threshold of the randomized count, respectively.

In case the SpatRaster contains NA cells, this needs to be reflected in the observation window of the point pattern as well (i.e., no point locations possible in these areas).

If `breaks = NULL` (default), only habitat labels (but not breaks) will be returned. If a vector with `breaks` is provided (same order as increasing habitat values), the breaks will be included as well.

Value

data.frame

References

Harms, K.E., Condit, R., Hubbell, S.P., Foster, R.B., 2001. Habitat associations of trees and shrubs in a 50-ha neotropical forest plot. Journal of Ecology 89, 947–959. <https://doi.org/10.1111/j.1365-2745.2001.00615.x>

Plotkin, J.B., Potts, M.D., Leslie, N., Manokaran, N., LaFrankie, J.V., Ashton, P.S., 2000. Species-area curves, spatial aggregation, and habitat specialization in tropical forests. Journal of Theoretical Biology 207, 81–99. <https://doi.org/10.1006/jtbi.2000.2158>

See Also

`reconstruct_pattern`
`fit_point_process`

Examples

```
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
species_a_random <- fit_point_process(species_a, n_random = 199)
results_habitat_association(pattern = species_a_random, raster = landscape_classified)
```

---

species_a *Species a*

Description

A species with negative associations to habitat 4 of `landscape`. Please be aware that a negative association to one habitat will inevitable lead to positive associations to other habitats (Yamada et al. 2006).

Usage

```
species_a
```

Format

A spatstat ppp object.

References

Yamada, T., Tomita, A., Itoh, A., Yamakura, T., Ohkubo, T., Kanzaki, M., Tan, S., Ashton, P.S., 2006. Habitat associations of Sterculiaceae trees in a Bornean rain forest plot. Journal of Vegetation Science 17, 559–566.

species_b *Species b*

Description

A species with positive associations to habitat 5 of `landscape`. Please be aware that a positive association to one habitat will inevitable lead to negative associations to other habitats (Yamada et al. 2006)

Usage

```
species_b
```

Format

A spatstat ppp object.

References

Yamada, T., Tomita, A., Itoh, A., Yamakura, T., Ohkubo, T., Kanzaki, M., Tan, S., Ashton, P.S., 2006. Habitat associations of Sterculiaceae trees in a Bornean rain forest plot. Journal of Vegetation Science 17, 559–566.

torus_trans *Torus trans*

Description

Torus translation of the classified `landscape`.

Usage

```
torus_trans
```

Format

rd_ras object.

translate_raster *translate_raster*

Description

Torus translation

Usage

```
translate_raster(
  raster,
  steps_x = NULL,
  steps_y = NULL,
  return_input = TRUE,
  simplify = FALSE,
  verbose = TRUE
)
```

Arguments

raster	SpatRaster with discrete habitat classes.
steps_x, steps_y	Integer with number of steps (cells) the raster is translated into the corresponding direction. If both are null, all possible combinations are used resulting in n = ((50 + 1) * (50 + 1)) - 4 rasters.
return_input	Logical if the original input data is returned.
simplify	Logical if only the raster will be returned if n_random = 1 and return_input = FALSE.
verbose	Logical if progress report is printed.

Details

Torus translation test as described in Harms et al. (2001). The raster is shifted in all four cardinal directions by steps equal to the raster resolution. If a cell exits the extent on one side, it enters the extent on the opposite side.

The method does not allow any NA values to be present in the SpatRaster.

Value

rd_ras

References

Harms, K.E., Condit, R., Hubbell, S.P., Foster, R.B., 2001. Habitat associations of trees and shrubs in a 50-ha neotropical forest plot. Journal of Ecology 89, 947–959. <https://doi.org/10.1111/j.1365-2745.2001.00615.x>

See Also

randomize_raster

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")

landscape_random <- translate_raster(landscape_classified)
landscape_random_sub <- translate_raster(landscape_classified,
steps_x = 1:10, steps_y = 1:5)

## End(Not run)
```

---

unpack_randomized *unpack_randomized*

Description

Load randomized raster object

Usage

```
unpack_randomized(raster)
```

Arguments

raster	rd_ras object with randomized raster.

Details

Because of how SpatRaster are saved (need to be packed), this function allows to unpack previously packed raster objects that were saved using pack_randomized. For further details, see wrap.

Value

rd_ras

See Also

pack_randomized wrap

Examples

```
## Not run:
landscape_classified <- classify_habitats(terra::rast(landscape), n = 5, style = "fisher")
landscape_random <- randomize_raster(landscape_classified, n_random = 3)
x <- pack_randomized(raster = landscape_random)
y <- unpack_randomized(raster = y)

## End(Not run)
```

Index