Package: rassta (via r-universe)

August 26, 2024

Title Raster-Based Spatial Stratification Algorithms

Version 1.0.5

Description Algorithms for the spatial stratification of landscapes, sampling and modeling of spatially-varying phenomena. These algorithms offer a simple framework for the stratification of geographic space based on raster layers representing landscape factors and/or factor scales. The stratification process follows a hierarchical approach, which is based on first level units (i.e., classification units) and second-level units (i.e., stratification units). Nonparametric techniques allow to measure the correspondence between the geographic space and the landscape configuration represented by the units. These correspondence metrics are useful to define sampling schemes and to model the spatial variability of environmental phenomena. The theoretical background of the algorithms and code examples are presented in Fuentes, Dorantes, and Tipton (2021). <doi:10.31223/X50S57>.

Maintainer Bryan A. Fuentes <bryandrep@gmail.com>

License AGPL (>= 3)

Encoding UTF-8

URL https://bafuentes.github.io/rassta/

BugReports https://github.com/bafuentes/rassta/issues/

LazyData true

Roxygen list(markdown = TRUE)

RoxygenNote 7.2.0

Imports cluster (>= 2.1.2), data.table (>= 1.14.0), dplyr (>= 1.0.7), DT (>= 0.18), foreach (>= 1.5.1), GGally (>= 2.1.2), ggplot2 (>= 3.3.5), grDevices, histogram (>= 0.0.25), KernSmooth (>= 2.23.18), kohonen (>= 3.0.10), plotly (>= 4.9.4.1), rlang (>= 0.4.11), scales (>= 1.1.1), shiny (>= 1.6.0), stats, stringdist (>= 0.9.6.3), stringi (>= 1.7.2), terra (>= 1.3.4), utils

Suggests testthat (>= 3.0.0), tinytest (>= 1.3.1), doParallel (>= 1.0.16), mgcv (>= 1.8.40), knitr, rmarkdown

Config/testthat/edition 3

VignetteBuilder knitr

Repository https://bafuentes.r-universe.dev

RemoteUrl https://github.com/bafuentes/rassta

RemoteRef HEAD

RemoteSha 63068334c7bc551d60639c6243e1353024805c68

Contents

`dummies` *Create Dummy Layers from Categorical Raster Layers*

Description

Given a SpatRaster whose cell values represent categories (e.g., soil types, land use/cover classes), a dummy layer indicating the presence/absence of each category is created, and optionally written on disk. Each category in the raster layer must be represented by a unique integer cell value. Output values indicating the presence or absence of categories in the dummy layers may be set using *preval* and *absval* arguments, respectively.

Usage

```
dummies(
  ca.rast,
  vals = NULL,
  preval = 100,
  absval = 0,
```

```
  to.disk = FALSE,
  outdir = ".",
  extension = ".tif",
  ...
)
```

Arguments

ca.rast	SpatRaster, as in rast. Single-layer SpatRaster whose (integer) cell values represent numeric IDs for categories.
vals	Integer. Vector or sequence of values representing the categories for which dummy layers will be created. If NULL, all categories will be processed. Default: NULL
preval	Integer. Value indicating presence of category. Default: 100
absval	Integer. Value indicating absence of category. Default: 0
to.disk	Boolean. Write output dummy layers to disk? Default: FALSE
outdir	Character. If *to.disk* = *TRUE*, string specifying the path for the output dummy raster layers. Default: "."
extension	Character. If *to.disk* = *TRUE*, String specifying the extension for the output raster layers (see **Details**). Default: ".tif"
...	If *to.disk* = *TRUE*, additional arguments as for writeRaster.

Details

This function mainly differs from segregate in that presence and absence values can be set arbitrarily, thus allowing for values other than those traditionally used in one-hot/dummy encoding (i.e., 1 and 0).

Please note that the argument *extension* does not correspond to the argument *filetype* in writeRaster. However, writeRaster should recognize the appropriate extension for the output raster layers from the *extension* argument. For instance, by setting *extension* = *".tif"*, writeRaster will recognize the extension as *GeoTiff*, which is the GDAL driver name.

Value

Single-layer or multi-layer SpatRaster with dummy layer(s).

See Also

segregate

Other Miscellaneous Functions: figure(), plot3D()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Single-layer SpatRaster of geologic units
f <- list.files(path = p, pattern = "geology2.tif", full.names = TRUE)
```

```
geol <- terra::rast(f)
# Dummy layer from geologic unit 1
dums <- dummies(ca.rast = geol, vals = 1, preval = 100, absval = 0)
```

`engine` *Predictive Modeling Engine*

Description

Modeling of spatially varying phenomena based on landscape similarity to stratification units. If each stratification unit across geographic space represents a distinct landscape configuration (in terms of multiple landscape factors and/or factor scales), and if each landscape configuration influences a phenomenon in a distinct way, then the spatial variability of that phenomenon can be assessed across a landscape by relating each geographic location to each distinct landscape configuration. Therefore, the more similar a geographic location is to the landscape configuration represented by a given stratification unit, then also the more similar the response of a phenomenon will be at that location to the typical response for conditions within the given stratification unit. Both continuous and categorical response variables are supported. For categorical responses, each category must be identified by an integer value.

Usage

```
engine(
  res.type = "cont",
  ls.rast,
  n.win = 3,
  su.repobs,
  tiles,
  parallel = FALSE,
  outdir = ".",
  tile.rm = TRUE,
  extension = ".tif",
  verbose = FALSE,
  ...
)
```

Arguments

`res.type`	Character. Type of response to model. Options are "cont" for continuous, and "cat" for categorical response. Default: "cont"
`ls.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster representing landscape similarities to stratification units. Only similarities for units with a representative observation are allowed. Character prefix in the file name of similarities is allowed.
`n.win`	Integer. Positive number indicating how many winning stratification units should be considered. See **Details**. Default: 3

su.repobs	Data frame. The first column of this data frame must contain only the numeric code for the stratification units (without prefix). Each additional column must contain the value of the representative response observation for each stratification unit. Multiple response variables are allowed (one per column). Note that all response variables in the data frame must share the same type (*res.type*). See example on issues related to **non-explicit column names**.
tiles	SpatVector, as in vect. Spatial vector of polygon geometry with the boundaries of the area of interest. This vector can be subdivided in regions (i.e., tiles) to balance memory allocation and processing speed (see **Details**). If this vector is tiled, then its attribute table must only contain an ID column with a unique identifier for each tile (1,2,...,n). Additionally, This vector must have the same coordinate reference system as *ls.rast*.
parallel	Boolean. Perform parallel processing? A parallel backend needs to be registered beforehand with registerDoParallel. Moreover, a tiled spatial vector should be supplied for *tiles*. Default: FALSE
outdir	Character. String specifying the path for the output raster tiles/layer(s) of modeled response(s). Default: "."
tile.rm	Boolean. Should the tiles of modeled response(s) be removed from disk after the tile merging process? Default: TRUE
extension	Character. String specifying the extension for the output raster layer(s) of modeled response(s). Default: ".tif"
verbose	Boolean. Show warning messages in the console? Default: FALSE
...	Additional arguments as for writeRaster.

Details

The predictive modeling process is cell-wise, which means that it operates on a cell-by-cell basis. For a given cell occurring in the geographic space supported by a raster layer, the predictive modeling engine first identifies the *n* stratification units to which the given cell is most similar (i.e., 'winning stratification units'). The engine is able to identify the winning stratification units thanks to the user-provided set of landscape similarity layers *ls.rast*. Subsequently, the response value from the representative observation for each winning stratification unit is identified. In the case of a continuous response, a weighted average of representative response values is performed. For each representative response value, the weight is proportional to the corresponding stratification unit's landscape similarity value in the given cell. The result of the weighted average is assigned as the response value in the given cell. In the case of a categorical response, the modal value from the representative response values of the *n* winning stratification units is assigned to the given cell.

Note that the name for each raster layer in *ls.rast* should match the numeric code of the corresponding stratification unit, which is obtained from the column of numeric codes in *su.repobs*. Nevertheless, raster layer names in *ls.rast* with a character prefix in the numeric code and/or file extension should work fine (e.g., "su_1101.tif" instead of "1101"). If the landscape similarity layers in *ls.rast* were created with similarity, then raster layer names will not have any prefix nor extension as part of the numeric code.

When dealing with large geographic spaces, high raster resolutions (i.e., small cell sizes), or both, a considerable amount of memory is required to perform the modeling process. To reduce memory usage, the predictive modeling engine performs tile-based processing of landscape similarity layers

and **writes results directly on disk**. Tile-based processing increases the computational time, thus parallelization is allowed by setting up a parallel backend. If parallelization is enabled, then care must be taken with the size of the tiles since larger sizes will have a greater impact on memory usage. Consequently, the parallel, tile-based processing will be less useful.

Value

Multi-layer or single-layer SpatRaster with modeled response(s).

See Also

similarity, observation

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRaster of landscape similarities
fls <- list.files(path = p, pattern = "su_", full.names = TRUE)
ls <- terra::rast(fls)
# Numeric code and representative response value for stratification units
fro <-list.files(path = p, pattern = "repobs.csv", full.names = TRUE)
ro <- read.csv(fro)
# Extract only those stratification units with representative value
ls <- ls[[as.character(ro$SU)]]
# SpatVector with processing tiles
fti <- list.files(path = p, pattern = "tiles.shp", full.names = TRUE)
ti <- terra::vect(fti)
# Directory for temporary files
o <- tempdir()
# Perform predictive modeling of continuous response
r <- engine(res.type = "cont", ls.rast = ls, n.win = 2, su.repobs = ro,
            tiles = ti, outdir = o, overwrite = TRUE
           )
# Plot modeled response
if(interactive()){plot(r)}
# Clean temporary files
file.remove(list.files(path = o, pattern = "soc.tif", full.names = TRUE))
#
#-------
# A note on non-explicit response's names (obtained from su.repobs):

## This will result in incorrectly modeled response values
x <- c("SOM", "SOM_30cm", "SOM_45cm")    # SOM = soil organic matter
grep(x[1], x)                            # Non explicit
grep(x[2], x)                            # Explicit
grep(x[3], x)                            # Explicit

## This will result in correct values
x <- c("SOM_15cm", "SOM_30cm", "SOM_45cm")
grep(x[1], x)                            # Explicit
grep(x[2], x)                            # Explicit
```

```
grep(x[3], x)                          # Explicit
```

---

figure *Reproduce Figures from Fuentes et al. (n.d.)*

Description

This function is intended to reproduce the figures presented in *rassta: Raster-based Spatial Stratification Algorithms* (Fuentes et al., 2021). Note that this function assumes that all the necessary inputs for each figure are loaded in the working environment. For the creation of each input, please refer to the data and examples presented in the aforementioned work. Also, please note that the use of this function is not intended for RStudio.

Usage

```
figure(x, d, scaling = 100, to.disk = FALSE, verbose = FALSE)
```

Arguments

x	Integer. Number identifying the figure to reproduce.
d	List. List with the data required for the figure to reproduce.
scaling	Integer. This number scales (i.e., resizes) the R's plotting device, such that *width* = *x/scaling* & *height* = *x/scaling*, with *x* = pixels. The default pixel size (not adjustable) and scaling value should work fine. Default = 100
to.disk	Boolean. Save figure to disk? Default = FALSE
verbose	Boolean. Show warning messages in the console? Default: FALSE

Value

None

References

B.A. Fuentes, M.J. Dorantes, and J.R. Tipton. rassta: Raster-based Spatial Stratification Algorithms. EarthArXiv, 2021. doi:10.31223/X50S57

See Also

Other Miscellaneous Functions: `dummies()`, `plot3D()`

Examples

```
if(interactive()){
require(terra)
p <- system.file("exdat", package = "rassta")
# Single-layer SpatRaster of geologic units
f <- list.files(path = p, pattern = "geology.tif", full.names = TRUE)
geol <- terra::rast(f)
# Dummy layers from geologic units
mat.sig <- dummies(ca.rast = geol, preval = 100, absval = 0)
figure(17, d = mat.sig)
}
```

`locations` *Select Representative Sampling Locations for Stratification Units*

Description

Selection of the *representative sampling locations* based on landscape similarity values. For a give stratification unit, the representative sampling location is the XY position where the highest landscape similarity value occurs. This location is assumed to best reflect the influence that the landscape configuration of a given stratification unit exerts on response phenomena. Currently, two selection methods are supported: (1) *maximum similarity within buffer zones* ("buffer"), and (2) *absolute maximum similarity* ("absolute"). For the buffer method, the *n* largest zones enclosing landscape similarity values above a certain threshold are first identified. Then, for each zone, one sample is placed at the *XY* location where the landscape similarity value is maximized. For the absolute method, a sample is placed at the *XY* locations with the *n* maximum landscape similarity values. In both methods, it is possible to constrain the sampling process to the boundaries of the stratification unit. Constraining the process ensures that the sampling locations determined for a given unit are placed within the boundaries of that unit. See **Details** for some guidance in the use of this function for classification units.

Usage

```
locations(
  ls.rast,
  su.rast,
  method = "buffer",
  constrained = TRUE,
  buf.quant = 0.9,
  buf.n = 1,
  abs.n = 1,
  tol = 1,
  parallel = FALSE,
  to.disk = FALSE,
  outdir = ".",
  verbose = FALSE,
  ...
)
```

Arguments

`ls.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster representing landscape similarities to stratification units.
`su.rast`	SpatRaster. Single-layer SpatRaster representing the stratification units occurring across geographic space. Integer values are expected as cell values (i.e., numeric codes) of stratification units.
`method`	Character. String denoting the sampling method. Current options are "buffer" for the maximum similarity within buffer zones method, and "absolute" for the absolute maximum similarity method. Default: "buffer"
`constrained`	Boolean. Should the sampling process be constrained to the boundaries of each stratification unit? See **Details**. Default: TRUE
`buf.quant`	Numeric. Number expressed in quantile notation (0-1) indicating the similarity threshold for the creation of buffer zones. Only zones enclosing raster cells with *landscape similarity* >= *buf.quant* will be created and thus, considered for sampling. See **Details**. Default: 0.9
`buf.n`	Integer. Positive integer indicating the *n* largest buffer zones for which sampling locations will be selected (*n* buffer zones per stratification unit, one sampling location per buffer zone). Default: 1
`abs.n`	Integer. When *method* = *"absolute"*, Positive integer indicating the number of sampling locations to select for each stratification unit. See **Details**. Default: 1
`tol`	Numeric. When *method* = *"absolute"*, this number will be subtracted from the sampled maximum value of a landscape similarity layer to ensure that the requested number of sampling locations will be found (see **Details**). The default assumes that landscape similarity values are on a scale of 1 to 100. If these values are on a different scale (e.g., decimal), then, *tol* needs to be adjusted accordingly. Default: 1
`parallel`	Boolean. Perform parallel processing? A parallel backend must be registered beforehand with `registerDoParallel`. Keep in mind that the amount of RAM to allocate when performing parallel processing can result prohibitive for large data sets. Default: FALSE
`to.disk`	Boolean. Should output SpatVector(s) (as in `vect`) be written to disk? Default: FALSE
`outdir`	Character. If *to.disk* = *TRUE*, string specifying the path for the output SpatVector(s). Default: "."
`verbose`	Boolean. Show warning messages in the console? Default: FALSE
`...`	Additional arguments, as for `writeVector`.

Details

Except when *buf.n* = *1* or *abs.n* = *1*, the number of returned sampling locations per stratification unit may be smaller than requested, especially when *constrained* = *TRUE*. For the constrained buffer method, reducing the landscape similarity threshold value *buf.quant* will not always result in more buffer zones; i.e., more sampling locations. The reason for this is that reducing the threshold value for the creation of buffer zones may actually promote the spatial contiguity of zones. For instance, two buffer zones created at *buf.quant* = *0.9*, may be merged into a single buffer zone when

buf.quant = 0.80. This will occur if the raster cells between the two buffer zones satisfy: *landscape similarity >= quantile(landscape similarity, 0.8)*. For the absolute method, increasing the value of the *tol* argument will ensure a safer search for *n* sampling locations and thus, greater chances of getting the total number of requested sampling locations per stratification unit.

Note that this sampling scheme can be applied for classification units. In order to do this, one should replace the multi-layer SpatRaster of landscape similarities with a multi-layer SpatRaster of spatial signatures. One should also replace the raster layer of stratification units with that of classification units.

Value

If *method = "buffer"* and *constrained = TRUE*, a list with the following components:

locations: SpatVector of point geometry. Each point in this vector represents the sampling location placed at the maximum landscape similarity value within a stratification unit's buffer zone. Tabular attributes in this SpatVector are (1) *SU* = stratification unit's numeric code, (2) *land_sim* = landscape similarity value, (3) *x* = X coordinate, and (4) *y* = Y coordinate.

buffer: SpatVector of polygon geometry. Each polygon in this vector represents the buffer zone of an stratification unit.

If *method = "buffer"* and *constrained = FALSE*:

locations: Same as **locations** from *method = "buffer"* and *constrained = TRUE*.

If *method = "absolute"*:

locations: SpatVector of point geometry. Each point in this vector represents the sampling location placed at the maximum landscape similarity value for an stratification unit. Tabular attributes in this SpatVector are (1) *SU* = stratification unit's numeric code, (2) *land_sim* = landscape similarity value, (3) *x* = X coordinate, and (4) *y* = Y coordinate.

See Also

similarity, strata

Other Functions for Stratified Sampling: observation()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRaster of landscape similarities
fls <- list.files(path = p, pattern = "su_", full.names = TRUE)
ls <- terra::rast(fls)
# Single-layer SpatRaster of stratification units
fsu <- list.files(path = p, pattern = "strata.tif", full.names = TRUE)
su <- terra::rast(fsu)
# Get 1 representative sampling location per stratification unit
rl <- locations(ls.rast = ls, su.rast = su)
# Plot representative locations (including buffer areas)
if(interactive()){
  plot(su, type = "classes", fun = function() c(points(rl$locations),
                                                polys(rl$buffers))
       )}
```

`observation` *Select the Representative Response Observation for Stratification Units*

Description

Selection of the *representative response observation* for each stratification unit occurring across geographic space. One observation from a set of *n* observations of a response variable sampled/measured within the spatial boundaries of a given stratification unit is selected according to the following criteria: (1) *maximum landscape similarity*, (2) *median response value*, and (3) *random selection*. The maximum landscape similarity (mls) selects the observation at the *XY* spatial location where the landscape similarity value is maximized for a given stratification unit. The median response value (mrv) selects the observation whose response value is (closest to) the median of all response values, as measured from the observations spatially enclosed by a given stratification unit. The random selection, as implied by the name, randomly selects one observation from the set of observations spatially enclosed by a given stratification unit. See **Details** for some guidance in the use of this function for classification units.

Usage

```
observation(
  su.rast,
  obs,
  col.id,
  col.resp,
  method = "mls",
  ls.rast,
  verbose = FALSE
)
```

Arguments

`su.rast`	SpatRaster, as in `rast`. Single-layer SpatRaster representing the stratification units occurring across geographic space. Integer values are expected as cell values (i.e., numeric codes) of stratification units.
`obs`	SpatVector, as in `vect`. Vector of point geometry whose tabular attributes should contain an ID column (1,2,...,n) and a column of the response' values.
`col.id`	Integer. Index of the ID column in the tabular attributes of *obs*.
`col.resp`	Integer. Index of the response' values column in the tabular attributes of *obs*.
`method`	Character. String specifying the selection method for the response representative observation. Options are "mls" for the maximum landscape similarity value, "mrv" for the median response value, and "random" for random selection. Default: "mls"
`ls.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster representing landscape similarities to stratification units. Only required if *method = "mls"*.
`verbose`	Boolean. Show warning messages in the console? Default: FALSE

Details

This selection scheme can be applied to classification units. For classification units, one should replace the multi-layer SpatRaster of landscape similarities with a multi-layer SpatRaster of spatial signatures. One should also replace the raster layer of stratification units with that of classification units.

Value

A list with the following components:

su.repobs: Data table with the following attributes: (1) Original IDs of the selected observation, (2) representative response value, (3) stratification unit's numeric code, and (4) landscape similarity value at the *XY* location of the selected observation (only if *method* = *"mls"*).

su.norepobs: List of the numeric codes of stratification units without observations.

su.repobs.sp: SpatVector of point geometry with the representative response observation for each stratification unit represented in *su.rast*.

See Also

strata, similarity

Other Functions for Stratified Sampling: locations()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Single-layer SpatRaster of stratification units
fsu <- list.files(path = p, pattern = "strata.tif", full.names = TRUE)
su <- terra::rast(fsu)
# Observations with response values.
## For this example, soil organic carbon (SOC) collected at 15 cm soil depth
fob <- list.files(path = p, pattern = "soc.shp", full.names = TRUE)
ob <- terra::vect(fob)
# Column indices for ID and measured response value
id <- 1
re <- 2
# Multi-layer SpatRaster of landscape similarities
fls <- list.files(path = p, pattern = "su_", full.names = TRUE)
ls <- terra::rast(fls)
# Selection of representative response observations for stratification units
ro <- observation(su.rast = su, obs = ob, col.id = id, col.resp = re,
                  ls.rast = ls
                 )
# Plot representative observations
if(interactive()){plot(su, type = "classes",
                       fun = function() points(ro$su_repobs.sp)
                      )}
```

plot3D *Interactive Maps of 3D surfaces*

Description

Interactive maps showing the 3-dimensional (*XYZ*) variability in raster layers representing continuous variables. The *XYZ* reference positions will be obtained from an elevation layer and the values of the continuous variables will be used as a surface color gradient. For this function to work, there must be a raster layer of elevation (e.g., digital terrain model) and at least one continuous variable among the raster layers to map. The maps produced are interactive, meaning that manual axis rotation and zoom are possible. Special consideration must be taken with large raster layers (large spatial coverage and/or high spatial resolution). This function can aggregates the spatial resolution (i.e., cell size) in order to handle large raster layers. This is achieved by internally calling `aggregate`. An aggregation factor will determine the final cell size, where *final cell size = cell size*aggregation factor*. In addition, a spatial extent can be provided to reduce the total mapping area and thus, to further reduce processing time. This function uses the **plotly** library. See **Details** for current limitations.

Usage

```
plot3D(
  var.rast,
  z,
  ex = 0.1,
  agg = FALSE,
  fact = NULL,
  spext = NULL,
  pals = NA,
  rev = NA
)
```

Arguments

`var.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster of *n* continuous variables and one layer representing the surface/terrain elevation.
`z`	Integer. Position (index) of the raster layer of elevation in *var.rast*.
`ex`	Numeric. Value indicating the *exaggeration* factor for the *Z* axis. This can be useful to enhance the visualization of subtle topographic variability. Default: 0.1
`agg`	Boolean. Should the spatial resolution be aggregated to reduce processing time? Default: FALSE
`fact`	Numeric. If *agg = TRUE*, value indicating the aggregation factor. Default: NULL
`spext`	Numeric. List with the coordinates of the bounding box for spatial subset (xmin, xmax, ymin, ymax). SpatRaster or SpatVector from which a spatial extent can be calculated are also an acceptable input. Default: NULL

pals	Character. List of strings with the names of the *n* color ramps (one per continuous variable). See hcl.colors. Default: NA
rev	Character. List of *n* Booleans indicating whether or not to reverse the color ramp for each continuous variable. Default: NA

Details

Currently, this function does not allow to adjust the labels for *XY* axes so that actual coordinates are shown. Instead, the relative position values are shown on these axes.

Value

List with **plotly-htmlwidget** objects. Each object calls the 3D map for a continuous variable in *var.rast*.

See Also

Other Miscellaneous Functions: dummies(), figure()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRaster of topographic variables
ft <- list.files(path = p, pattern = "^height|^slope|^wetness",
                 full.names = TRUE
                )
tvars <- terra::rast(ft)
# Single-layer SpatRaster of terrain elevation
fe <- list.files(path = p, pattern = "^elevation", full.names = TRUE)
e <- terra::rast(fe)
# Add elevation to the SpatRaster of topographic variables
etvars <- c(e, tvars)
# Interactive 3D maps
maps <- plot3D(var.rast = etvars, z = 1, ex = 0.2,
               pals = c("Zissou", "Plasma", "Spectral")
              )
if(interactive()){maps}
```

---

predict_functions *Predict Distribution Functions Across Geographic Space*

Description

Predicts constrained, univariate distribution functions across the geographic space supported by raster layers. For a given continuous variable used to create a classification unit, this function first calculates a user-defined distribution function for that variable using only observations selected from within the classification unit. In this way, the distribution function is univariate and

constrained. Subsequently, a *locally-estimated scatterplot smoothing* (LOESS) or a *generalized additive model* (GAM) is fitted. This model is fitted using the variable's observations as explanatory values and the values from the distribution function as the response values. Finally, the fitted model is predicted on the complete geographic space supported by the raster layer of the given variable. This process is iterated for all of the continuous variables and classification units. Each resulting layer can be thought of as a landscape correspondence measurement between an *XY* location in geographic space and the landscape configuration represented by a given classification unit in terms of a specific variable. The following distribution functions are currently supported: the probability density function (PDF), the empirical cumulative density function (ECDF), and the inverse of the empirical cumulative density function (iECDF). Please refer to **Details** for more information about how each distribution function is calculated. Also, see details on parallel processing.

Usage

```
predict_functions(
  cuvar.rast,
  cu.ind,
  cu,
  vars,
  dif,
  hist.type = "regular",
  hist.pen = "default",
  grid.mult = 1,
  kern = "normal",
  quant.sep = 0.01,
  method = "loess",
  span = 0.6,
  k = 20,
  discrete = TRUE,
  to.disk = FALSE,
  outdir = ".",
  prefix = "",
  extension = ".tif",
  overwrite = FALSE,
  verbose = FALSE,
  ...
)
```

Arguments

`cuvar.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster containing *n* continuous raster layers (i.e., variables) and one raster layer of classification units with integer cell values (i.e., Numeric identifiers).
`cu.ind`	Integer. Position (index) of the raster layer of classification units in *cuvar.rast*.
`cu`	Integer. Vector of integer values that correspond to the numeric identifiers of the units in the raster layer of classification units.
`vars`	Character. Vector of strings containing the names of the *n* continuous variables in *cuvar.rast*. These names have to be sequentially repeated according to the

number of classification units (See **Examples**).

`dif` Character. Vector of strings containing the distribution function to calculate for each continuous variable within each classification unit. The function will match the position of the name of the distribution function with that of the name of the continuous variable in *vars*.

`hist.type` Character. Type of histogram to calculate. Options are "regular", "irregular" (unequally-sized bins, very slow), and "combined" (the one with greater penalized likelihood is returned). See `histogram`. Default: "regular"

`hist.pen` Character. Penalty to apply when calculating the histogram (see `histogram`). Default: "default"

`grid.mult` Numeric. Multiplying factor to increase/decrease the size of the "optimal" grid size for the *Kernel Density Estimate* (KDE). Default: 1

`kern` Character. Type of kernel to use for the KDE. Default: "normal"

`quant.sep` Numeric. Spacing between quantiles for the calculation of the ECDF and iECDF. Quantiles are in the range of 0-1 thus spacing must be a decimal. Default: 0.01

`method` Character. Model to fit. Current options are "loess" for locally-estimated scatterplot smoothing (see `loess`), and "gam" for generalized additive model with support for large datasets (see `bam`). Default: "loess"

`span` Numeric. If *method* = *"loess"*, degree of smoothing for LOESS fitting. Default: 0.6

`k` Numeric. If *method* = *"gam"*, Number of knots for the *cubic regression splines*. Default: 20

`discrete` Boolean. If *method* = *"gam"*, discretize variables for storage and efficiency reasons? Can reduce processing time significantly. Default: TRUE

`to.disk` Boolean. Write the output raster layers of predicted distribution function to disk? See details an example about parallel processing. Default: FALSE

`outdir` Character. If *to.disk* = *TRUE*, string specifying the path for the output raster layers of predicted distribution function. Default: "."

`prefix` Character. If *to.disk* = *TRUE*, string specifying a prefix for the file names of the output raster layers of predicted distribution function. Default: ""

`extension` Character. If *to.disk* = *TRUE*, string specifying the extension for the output raster layers of predicted distribution function. Default: ".tif"

`overwrite` Boolean. If *to.disk* = *TRUE*, should raster layers in disk and with same name as the output raster layer(s) of predicted distribution function be overwritten? Default: FALSE

`verbose` Boolean. Show warning messages in the console? Default: FALSE

`...` If *to.disk* = *TRUE*, additional arguments as for `writeRaster`.

Details

To calculate the PDF, this function uses the binned KDE for observations drawn from the breaks of a regular/irregular histogram. The "optimal" number of bins for the histogram is defined by calling the function `histogram` (Mildenberger et al., 2019) with the user-defined penalty *hist.pen*.

Subsequently, the optimal number of bins is treated as equivalent to the "optimal" grid size for the binned KDE. The grid size can be adjusted by specifying the multiplying factor *grid.mult*. Lastly, the "optimal" bandwidth for the binned KDE is calculated by applying the *direct plugin* method of Sheather and Jones (1991). For the calculation of optimal bandwidth and for the binned KDE, the package **KernSmooth** is called. To calculate both the ECDF and the iECDF, this function calls the `ecdf` function on equally-spaced quantiles. The spacing between quantiles can be manually adjusted via *quant.sep*. In the case of iECDF, the ECDF is inverted by applying the formula: *iECDF = ((x - max(ECDF)) * -1) + min(ECDF)*; where *x* corresponds to each value of the ECDF.

The "cu", "vars", and "dif" parameters of this function are configured such that the output table from `select_functions` can be used directly as input. (see **Examples**).

When writing output raster layer to disk, multiple distribution functions can be predicted in parallel if a parallel backend is registered beforehand with `registerDoParallel`. Keep in mind that the function may require a large amount of memory when using a parallel backend with large raster layers (i.e., high resolution and/or large spatial coverage).

Some issues have been reported when manually creating cluster objects using the **parallel** package. To overcome this issue, a cluster object can be registered directly through `registerDoParallel` without passing it first through `makeCluster`. See examples.

Value

Single-layer or multi-layer SpatRaster with the predicted distribution function for each variable and for each classification unit.

References

T. Mildenberger, Y. Rozenholc, and D. Zasada. histogram: Construction of Regular and Irregular Histograms with Different Options for Automatic Choice of Bins, 2019. https://CRAN.R-project.org/package=histogram

S. Sheather and M. Jones. A reliable data-based bandwidth selection method for kernel density estimation. Journal of the Royal Statistical Society. Series B. Methodological, 53:683–690, 1991.

See Also

Other Landscape Correspondence Metrics: `select_functions()`, `signature`, `similarity()`

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRaster of topographic variables
## 3 continuous variables
ftva <- list.files(path = p, pattern = "^height|^slope|^wetness",
                   full.names = TRUE
                  )
tva <- terra::rast(ftva)
# Single-layer SpatRaster of topographic classification units
## Five classification units
ftcu <- list.files(path = p, pattern = "topography.tif", full.names = TRUE)
tcu <- terra::rast(ftcu)
```

```
# Add the classification units to the SpatRaster of topographic variables
tcuvars <- c(tcu, tva)
# Data frame with source for "cu", "vars", and "dif"
ftdif <- list.files(path = p, pattern = "topodif.csv", full.names = TRUE)
tdif <- read.csv(ftdif)
# Check structure of source data frame
head(tdif)
# Predict distribution functions
## 1 distribution function per variable and classification unit = 1
tpdif <- predict_functions(cuvar.rast = tcuvars,
                           cu.ind = 1,
                           cu = tdif$Class.Unit[1:3],
                           vars = tdif$Variable[1:3],
                           dif = tdif$Dist.Func[1:3],
                           grid.mult = 3,
                           span = 0.9
                          )
# Plot predicted distribution functions
if(interactive()){plot(tpdif, col = hcl.colors(100, "Oslo", rev = TRUE))}

#--------
# Writing results to disk and parallel processing

if(interactive()){
  # Directory for temporary files
  o <- tempdir()
  # Register parallel backend
  require(doParallel)
  registerDoParallel(4)
  # Predict distribution functions
  tpdif <- predict_functions(cuvar.rast = tcuvars,
                             cu.ind = 1,
                             cu = tdif$Class.Unit[1:3],
                             vars = tdif$Variable[1:3],
                             dif = tdif$Dist.Func[1:3],
                             grid.mult = 3, span = 0.9,
                             to.disk = TRUE,
                             outdir = o
                            )
   # Stop cluster
   stopImplicitCluster()
   # Clean temporary files
   file.remove(sources(tpdif))
 }
```

`select_functions` *Select Constrained Univariate Distribution Functions*

Description

Selection of distribution functions for continuous raster layers that were used to create a raster layer of classification units. The distribution functions currently supported are the probability density function (PDF), the empirical cumulative density function (ECDF), and the inverse of the empirical cumulative density function (iECDF). Please note that `select_functions` DOES NOT calculate the aforementioned distribution functions. The sole purpose of `select_functions` is to assist in the knowledge-driven selection of the most appropriate distribution function for each continuous variable used to create a given classification unit (see **Details**).

Usage

```
select_functions(
  cu.rast,
  var.rast,
  fun = mean,
  varscale = "uniminmax",
  mode = "auto",
  verbose = TRUE,
  ...
)
```

Arguments

`cu.rast`	SpatRaster, as in `rast`. Single-layer SpatRaster representing the classification units occurring across geographic space. The cell values (i.e., numeric IDs) for classification units must be integer values.
`var.rast`	SpatRaster. Multi-layer SpatRaster containing the *n* continuous raster layers of the variables used to create the classification units.
`fun`	Character. Descriptive statistical measurement (e.g., mean, max). See `zonal`. Default: mean
`varscale`	Character. Variable scaling method. See *scale* argument in `ggparcoord`. Default: "uniminmax"
`mode`	Character. String specifying the selection mode for univariate distribution functions. Possible values are "inter" for interactive selection, and "auto" for automatic selection (see Details). Default: "auto"
`verbose`	Boolean. Show warning messages in the console? Default: FALSE
`...`	Additional arguments as for `ggparcoord`.

Details

The selection of distribution functions is univariate, that is, for each variable, and it is constrained, meaning that the selection has to be made for each classification unit. Overall, the distribution functions are used to characterize typical values of a given continuous variable within a given classification unit. When the PDF is selected, values closer to, or at the peak of the PDF will be considered as the most typical. Contrarily, values at the tails of the PDF will be considered as the less typical. When the ECDF or the iECDF are selected, values toward (+)infinity and (-)infinity will be considered as the most typical values, respectively.

In order to assist the selection process, when *mode* = *"inter"*, this function displays an interactive parallel coordinates plot (see ggplotly) and a writable table (built in Shiny). For each variable, the parallel coordinates plot shows a trend of a descriptive statistical measurement (argument *fun*) across all of the classification units. Using this trend, one can then select the most appropriate distribution function for each variable based on the occurrence/absence of **"peaks"** and **"pits"** in the observed trend. For instance, a peak (highest point in the trend) would indicate that the given classification unit contains on average, the highest values of that variable. Conversely, a pit (lowest point in the trend) would indicate that the given classification unit contains on average, the lowest values of that variable. Thus, an ECDF and an iECDF can be selected for the peak and the pit, respectively. The PDF can be selected for classification units whose trend does not show either a peak or a pit. Please consider that peaks and pits are only reference points and thus, one should validate the selection of distribution functions based on domain knowledge.

When *mode* = *"auto"*, the criteria for the selection of distribution functions will be based on peaks and pits in the parallel coordinates plot.

The output table (**distfun**) is intended to be used as input in the predict_functions function.

The selection of distribution functions is similar to the selection of membership functions in fuzzy logic. For example, if one wants to describe a phenomenon through distribution functions of continuous variables, then the functions can be considered to be membership curves. Accordingly, the PDF, ECDF, and iECDF will be equivalent to the Gaussian, S, and Z membership functions, respectively.

Value

If *mode* = *"inter"*:

distfun: A DT table (DataTables library) with the following attributes: (1) *Class.Unit* = numeric ID for classification units, (2) *Variable* = each of the *n* continuous raster layers of a classification unit, and (3) *Dist.Func* = Empty column whose cells can be filled with the following strings: "PDF, "ECDF", and "iECDF" (unquoted). This table can be saved on disk through the Shiny interface.

parcoord: A plotly-based parallel coordinate plot which can be saved on disk using the R package htmlwidgets.

If *mode* = *"auto"*:

distfun: Same as **distfun** when *mode* = *"inter"*, except for column "Dist.Func" whose cells were automatically filled.

parcoord: Same as **parcoord** when *mode* = *"inter"*.

See Also

Other Landscape Correspondence Metrics: predict_functions(), signature, similarity()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRaster of topographic variables
## 3 topographic variables
tf <- list.files(path = p, pattern = "^height|^slope|^wetness",
                 full.names = TRUE
```

```
                )
tvars <- terra::rast(tf)
# Single-layer SpatRaster of topographic classification units
## 5 classification units
tcf <- list.files(path = p, pattern = "topography.tif", full.names = TRUE)
tcu <- terra::rast(tcf)
# Automatic selection of distribution functions
tdif <- select_functions(cu.rast = tcu, var.rast = tvars, fun = mean)
# Parallel coordinates plot
if(interactive()){tdif$parcoord}
```

`signature` *Calculate the Spatial Signature of Classification Units*

Description

Using a mathematical function, a raster layer is created from the cell-wise aggregation of a set of predicted distribution functions for a classification unit (see `predict_functions`). Each element in this set can be thought of as a correspondence measurement between an *XY* location in the geographic space and the landscape configuration represented by a given classification unit in terms of a specific variable. Therefore, aggregating the set of predicted distribution functions into a single layer produces an overall (multivariate) measurement of correspondence. This multivariate landscape correspondence metric is considered to be the *spatial signature* of the classification unit.

Usage

```
signature(
  pdif.rast,
  inprex,
  outname,
  fun = mean,
  to.disk = FALSE,
  outdir = ".",
  extension = ".tif",
  overwrite = FALSE,
  ...
)
```

Arguments

`pdif.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster whose raster layers represent predicted distribution functions for continuous variables and for one or more classification units. All predicted distribution functions for a particular classification unit are considered as part of the same set. There must be a matching pattern in the names of predicted distribution functions from the same set (see *inprex*).

inprex	Character. Prefix in the name of raster layers representing predicted distribution functions belonging to the same set (i.e.,same classification unit). If spatial signatures for multiple sets are to be calculated, a vector of strings must be provided. See **Details** and **Examples**.
outname	Character. Output layer/file name for the raster layer(s) of spatial signature. If the spatial signatures for multiple classification units are to be calculated, then a vector of strings must be provided.
fun	Function. The mathematical function must take a vector of values and return a single value (e.g., mean, max, sum, etc.). See app. Default: mean
to.disk	Boolean. Write the output raster layer(s) of spatial signature to disk? See details about parallel processing. Default: FALSE
outdir	Character. If *to.disk* = *TRUE*, string specifying the path for the output raster layer(s) of spatial signature. Default: "."
extension	Character. If *to.disk* = *TRUE*, String specifying the extension for the output raster layer(s) of spatial signature. Default: ".tif"
overwrite	Boolean. If *to.disk* = *TRUE*, should raster layers in disk and with same name as the output raster layer(s) of spatial signature be overwritten? Default: FALSE
...	If *to.disk* = *TRUE*, additional arguments as for writeRaster.

Details

Raster layers of predicted distribution functions belonging to the same classification unit must be identified by a unique prefix in their layer names (argument *inprex*). This prefix is used as a string pattern to find all the predicted distribution functions belonging to one classification unit. Consequently, a unique prefix must be defined for each additional classification unit to distinguish between predicted distribution functions for different classification units. Similarly, an additional string (or vector of strings) of file/layer name(s) must be provided to distinguish between the resulting spatial signatures for different classification units (argument *outname*). The length of *outname* must match that from *inprex*.

When writing the output raster layers of spatial signature to disk, a parallel backend can be registered before running this function with registerDoParallel to speed-up computation. Note that this is only helpful when calculating spatial signatures for many classification units.

From a spatial analysis standpoint, the aggregation of predicted distribution functions into spatial signature is similar to the application of fuzzy aggregation operators commonly used in GIS-based multi-criteria decision analysis. Moreover, The use of descriptive statistics to calculate *signatures* for landscape-related classification units can be traced back to the works of Pike and Rozema (1975), and Pike (1988).

Value

Single-layer or multi-layer SpatRaster with the spatial signature(s) calculated from the set(s) of predicted distribution functions.

References

R. Pike. The geometric signature: quantifying landslide-terrain types from digital elevation models. Mathematical geology, 20(5):491–511, 1988. doi:10.1007/BF00890333

R. Pike and W. Rozema. Spectral analysis of landforms. Annals of the Association of American Geographers,65(4):499–516, 1975. doi:10.1111/j.14678306.1975.tb01058.x

See Also

Other Landscape Correspondence Metrics: predict_functions(), select_functions(), similarity()

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRast with predicted distribution functions
## 3 continuous variables and 5 classification units, = 15 functions
ft <- list.files(path = p, pattern = "topo_", full.names = TRUE)
t <- terra::rast(ft)
# Vector with the prefix for each set of predicted distribution functions
## 5 classification units = 5 sets
it <- paste("topo_", seq(1, 5), "_", sep = "")
# Vector of names for output raster layers of spatial signature
## 5 spatial signatures, one per classification unit
ot <- paste("topography_", seq(1, 5), sep = "")
# Calculate spatial signatures
tsig <- signature(pdif.rast = t, inprex = it, outname = ot)
# Plot spatial signatures
if(interactive()){plot(tsig, col = hcl.colors(100, "Oslo", rev = TRUE))}
```

---

similarity *Calculate the Landscape Similarity to Stratification Units*

Description

For each stratification unit present in a single-layer SpatRaster, a raster layer of landscape similarity is created by aggregating the stratification unit's corresponding set of spatial signatures (see signature). For a stratification unit *x*, the corresponding set of spatial signatures consists of one spatial signature for each of the *n* classification units that are present in the numeric code of *x* (one classification unit per landscape factor/factor scale). The aggregation process is performed cell-wise, and by using a mathematical function which takes multiple values but return a single value (e.g., mean, sum, min, max). The resulting raster layer represents the correspondence between an *XY* location in geographic space and the landscape configuration represented by a given stratification unit.

Usage

```
similarity(
  su.rast,
  su.code,
  sig.rast,
  fun = mean,
```

```
  to.disk = FALSE,
  outdir = ".",
  prefix = "su_",
  extension = ".tif",
  overwrite = FALSE,
  ...
)
```

Arguments

su.rast	SpatRaster, as in rast. Single-layer SpatRaster representing the stratification units occurring across geographic space. Integer values are expected as cell values (i.e., numeric codes) of stratification units.
su.code	List. The structure of the stratification units' numeric code. This (nested) list should indicate the names of the landscape factors/factor scales used to create the stratification units, and the position (start, end) of their corresponding classification units' ID in the numeric code. See **Examples**.
sig.rast	SpatRaster. Multi-layer SpatRaster with the spatial signatures of all the classification units that were used to create the stratification units. The spatial signatures should follow this name convention: *x_n*; where *x* is the landscape factor/factor scale, and *n* is the numeric ID of the classification unit to which the spatial signature belongs.
fun	Function. The mathematical function must accept a vector of values and return a single value (e.g., mean, max, sum, etc.). See app. Default: mean
to.disk	Boolean. Write the output raster layers of landscape similarity to disk? See note about parallel processing. Default: FALSE
outdir	Character. If *to.disk* = *TRUE*, string specifying the path for the output raster layers of landscape similarity. Default: "."
prefix	Character. If *to.disk* = *TRUE*, prefix for the file name of the output raster layers of landscape similarity. Default: "su_"
extension	Character. If *to.disk* = *TRUE*, string specifying the extension for the output raster layers of landscape signature. Default: ".tif"
overwrite	Boolean. When *to.disk* = *TRUE*, should raster layers in disk and with same name as the output landscape similarities be overwritten? Default: FALSE
...	Additional arguments as for writeRaster (if *to.disk* = *TRUE*).

Details

The landscape similarity is a landscape correspondence metric. The aggregation of multiple spatial signatures into a single landscape similarity layer is somewhat similar to the application of fuzzy logic and aggregation operators in GIS-based multi-criteria decision analysis. Furthermore, the aggregation of raster layers indicating relative optimality for spatially-varying phenomena, like spatial signatures, can be guided by physical/ecological principles like Sprengel-Liebig's law of the minimum. In such case, one could select the *min* function when aggregating the spatial signatures into landscape similarities.

When writing the output raster layers of landscape similarity to disk, a parallel backend can be registered before running this function with registerDoParallel to speed-up computation.

Value

A list with the following components:

landsim: Multi-layer SpatRaster with the landscape similarity to each stratification unit present in *su.rast*.

codes: A data frame with the numeric code for each stratification unit and the corresponding classification units' numeric ID for each landscape factor/factor scale.

See Also

strata

Other Landscape Correspondence Metrics: predict_functions(), select_functions(), signature

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Single-layer SpatRaster of stratification units
fsu <- list.files(path = p, pattern = "strata2.tif", full.names = TRUE)
su <- terra::rast(fsu)
# Define the structure of the stratification units' numeric code
code <- list(geology = c(1,1), climate = c(2,2), topography = c(3,3))
# Multi-layer SpatRaster of spatial signatures of classification units
fsig <- list.files(path = p, pattern = "geology_|climate_|topography_",
                   full.names = TRUE
                  )
sig <- terra::rast(fsig)
# Calculate landscape similarity to stratification units
landsim <- similarity(su.rast = su, su.code = code, sig.rast = sig)

# Plot some landscape similarities
# if(interactive()){plot(landsim$landsim[[c(1,10,12,14)]],
#                        col = hcl.colors(100, "Oslo", rev = TRUE)
#                       )}

#-------
# A note on the numeric code of stratification units

# For a given stratification unit, the structure of its corresponding numeric
# code indicates: (1) the landscape factors and/or factor scales that were
# accounted for when creating the stratification unit, and (2) the numeric id
# of the classification unit from each landscape factor/factor scale.
# Consider the following numeric code structure:

su.code <- list(geology = c(1,1), climate = c(2,2), topography = c(3,4))

# The stratification units are composed of classification units from...
# ...three landscape factors: geology, climate, and topography
names(su.code)

# For geology, the classification units are represented by the first...
```

```
# ...digit in the numeric code
su.code$geology

# For climate, the classification units are represented by the second...
# ...digit in the numeric code
su.code$climate
# For topography, the classification units are represented by the third...
# ...and fourth digit in the numeric code
su.code$topography

# Thus, the numeric code of the stratification units 1101 and 2410 means:
su <- c(1101, 2410)
su[1]   # 'geology' = 1, 'climate' = 1, and 'topography' = 1
su[2]   # 'geology' = 2, 'climate' = 4, and 'topography' = 10
```

---

som_gap *Self-Organizing Map and Selection of k*

Description

Produces a low-dimensional representation of the input feature space for subsequent estimation of the "optimal" number of clusters (k) in a multivariate dataset. The dimension reduction is based on the self-organizing map technique (SOM) of Kohonen (1982; 1990), and implemented in R by the function supersom of Wehrens and Kruisselbrink (2018). To estimate the optimal k, the partitioning around medoids (PAM) of Kaufman and Rousseeuw (1990), coupled with the gap statistic of Tibshirani et al. (2001), is performed on the SOM's codebook vectors. This is achieved by internally calling pam and clusGap (Maechler et al., 2021). See *Details* for a brief theoretical background.

Usage

```
som_gap(
  var.rast,
  xdim = 12,
  ydim = 12,
  topo = "hexagonal",
  neighbourhood.fct = "gaussian",
  rlen = 600,
  dist.fcts = c("sumofsquares", "manhattan"),
  mode = "pbatch",
  K.max,
  stand = FALSE,
  B = 500,
  d.power = 2,
  spaceH0 = "original",
  method = "globalSEmax",
  SE.factor = 1,
  ...
)
```

Arguments

var.rast	SpatRaster, as in rast. This Multi-layer SpatRaster must contain *n* continuous variables from which the SOM will be created.
xdim	Integer. Horizontal dimension of the SOM's grid. Default: 12
ydim	Integer. Vertical dimension of the SOM's grid. Default: 12
topo	Character. Topology of the SOM's grid. Options = "rectangular", "hexagonal". Default: "hexagonal"
neighbourhood.fct	Character. Neighborhood of the SOM's grid. Options = "bubble", "gaussian". Default: "gaussian"
rlen	Integer. Number of times the complete dataset will be presented to the SOM's network. Default: 600
dist.fcts	Character. Vector of length 2 containing the distance functions to use for SOM (First element, options = "sumofsquares", "euclidean", "manhattan") and for PAM (second element, options = "euclidean", "manhattan"). Default: c("sumofsquares", "manhattan")
mode	Character. Type of learning algorithm. Options are "online", "batch", and "pbatch". Default: "pbatch"
K.max	Integer. Maximum number of clusters to consider, must be at least two (2).
stand	Boolean. For PAM function, does SOM's codebook vectors need to be standardized? Default: FALSE
B	Integer. Number of bootstrap samples for the gap statistic. Default: 500
d.power	Integer. Positive Power applied to euclidean distances for the gap statistic. Default: 2
spaceH0	Character. Space of the reference distribution for the gap statistic. Options = "scaledPCA", "original" (See **Details**). Default: "original"
method	Character. Optimal k selection criterion for the gap statistic. Options = "globalmax", "firstmax", "Tibs2001SEmax", "firstSEmax", "globalSEmax". See clusGap for more details. Default: "globalSEmax"
SE.factor	Numeric. Factor to feed into the standard error rule for the gap statistic. Only applicable for methods based on standard error (SE). See clusGap for more details. Default: 1
...	Additional arguments as for supersom.

Details

The clustering of SOM's codebook vectors has been proposed in several works, notably in that from Vesanto and Alhoniemi (2000). These authors proposed a two-stage clustering routine as an efficient method to reduce computational load, while obtaining satisfactory correspondence between the clustered codebook vectors and the clustered original feature space.

The main purpose of this function is to allow the use of clustering and k-selection algorithms that may result prohibitive for large datasets, such as matrices derived from raster layers commonly used during geocomputational routines. Thus, the SOM's codebook vectors can be subsequently used

for the calculation of distance matrices, which given the large size of their input feature space, may otherwise be impossible to create due to insufficient memory allocation capacity. Similarly, robust clustering algorithms that require full pairwise distance matrices (e.g., hierarchical clustering, PAM) and/or eigenvalues (e.g., spectral clustering) may also be performed on SOM's codebook vectors.

Note that `supersom` will internally equalize the importance (i.e., weights) of variables such that differences in scale will not affect distance calculations. This behavior can be prevented by setting *normalizeDataLayers* = *FALSE* in additional arguments passed to `supersom`. Moreover, custom weights can also be passed through the additional argument *user.weights*. In such case, user weights are applied on top of the internal weights.

When working with large matrices, the additional SOM argument *keep.data* may be set to FALSE. However, note that by doing so, the suggested follow-up function for raster products `som_pam` will not work since it requires both original data and winning units.

For the gap statistic, *method* = *"scaledPCA"* has resulted in errors for R sessions with BLAS/LAPACK supported by the Intel Math Kernel Library (MKL).

Value

SOM: An object of class **kohonen** (see `supersom`). The components of class kohonen returned by this function are: (1) *data* = original input matrix, (2) *unit.classif* = winning units for all observations, (3) *distances* = distance between each observation and its corresponding winning unit, (4) *grid* = object of class **somgrid** (see `somgrid`), (5) *codes* = matrix of codebook vectors, (6) *changes* = matrix of mean average deviations from codebook vectors, (7) *dist.fcts* = selected distance function, and other arguments passed to `supersom` (e.g., *radius*, *distance.weights*, etc.). Note that components 1, 2, and 3 will only be returned if *keep.data* = *TRUE*, which is the default.

SOMdist: Object of class **dist**. Matrix of pairwise distances calculated from the SOM's codebook vectors.

SOMgap: Object of class **clusGap**. The main component of class clusGap returned by this function is *Tab*, which is a matrix of the gap statistic results (see `clusGap`). Additional components are the arguments passed to the function (i.e., *spaceH0*, *B*), the PAM function, *n* (number of observations) and *call* (the clusGap call-type object).

Kopt: Optimal *k*, as selected by arguments *method* and (possibly) *SE.factor*.

References

L. Kaufman and P. Rousseeuw. Finding groups in data: an introduction to cluster analysis. John Wiley & Sons, 1990. doi:10.1002/9780470316801

T. Kohonen. Self-organized formation of topologically correct feature maps. Biological cybernetics, 43 (1):59–69, 1982. doi:10.1007/bf00337288

T. Kohonen. The self-organizing map. Proceedings of the IEEE, 78(9):1464–1480, 1990. doi:10.1016/s09252312(98)000307

M. Maechler, P. Rousseeuw, A. Struyf, M. Hubert, and K. Hornik. cluster: Cluster Analysis Basics and Extensions, 2021. https://CRAN.R-project.org/package=cluster

R. Tibshirani, G. Walther, and T. Hastie. Estimating the number of clusters in a data set via the gap statistic. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 63(2):411–423, 2001. doi:10.1111/14679868.00293

J. Vesanto and E. Alhoniemi. Clustering of the self-organizing map. IEEE Transactions on Neural Networks, 11(3):586–600, 2000. doi:10.1109/72.846731

R. Wehrens and J. Kruisselbrink. Flexible self-organizing maps in kohonen 3.0. Journal of Statistical Software, 87(1):1–18, 2018. doi:10.18637/jss.v087.i07

See Also

Other Functions for Landscape Stratification: `som_pam()`, `strata()`

Examples

```
require(terra)
# Multi-layer SpatRaster with topographic variables
p <- system.file("exdat", package = "rassta")
tf <- list.files(path = p, pattern = "^height|^slope|^wetness",
                 full.names = TRUE
                )
t <- rast(tf)
# Scale topographic variables (mean = 0, StDev = 1)
ts <- scale(t)
# Self-organizing map and gap statistic for optimum k
set.seed(963)
tsom <- som_gap(var.rast = ts, xdim = 8, ydim = 8, rlen = 150,
                mode = "online", K.max = 6, B = 300, spaceH0 = "original",
                method = "globalSEmax"
               )
# Optimum k
tsom$Kopt
```

---

som_pam *Rasterization of Self-Organizing Map and Partitioning Around Medoids*

Description

Creation of a rasterized representation of the outputs from the self-organizing map (SOM) and partitioning around medoids (PAM). Given a reference raster layer, each winning unit of the SOM and clustering value from the PAM will be mapped on the corresponding cell in the reference layer and across the geographic space supported by such layer. Note that this function is a follow-up of the `som_gap` function.

Usage

```
som_pam(ref.rast, kohsom, k, metric = "manhattan", stand = FALSE, ...)
```

Arguments

ref.rast	SpatRaster, as in rast. This raster layer will serve as a reference of the index of valid cells and the geographic support for the rasterized representation of SOM's winning units and PAM's clustering. See *Details* for some notes about efficiency.
kohsom	SOM Object of class **kohonen**, see supersom. The following components must be present in the SOM object (1) *unit.classif* = winning units for all observations, and (2) *codes* = matrix of codebook vectors.
k	Integer (positive value). Number of clusters to form from the codebook vectors of the SOM, where $k <$ *SOM's codebook vectors*.
metric	Character. Distance function for PAM. Options are "euclidean", and "manhattan". Default: "manhattan"
stand	Boolean. For the PAM function, do SOM's codebook vectors need to be standardized? Default: FALSE
...	Additional arguments as for pam. See **Details**.

Details

As in som_gap, this function calls pam to perform the clustering of SOM's codebook vectors. The SOM object must belong to the class **kohonen**, as in supersom.

Note that in order for som_pam to perform efficiently, the reference SpatRaster *ref.rast* must be a single-layer SpatRaster with the same cell size, number of rows, number of columns, and index of valid cells as those in the multi-layer SpatRaster object used in som_gap. If a multi-layer SpatRaster (with each layer possibly having a different index of valid cells) is used as the *ref.rast*, the efficiency of som_pam (i.e., running time and/or memory allocation) may be degraded when handling large SpatRaster objects.

For this function to work as intended, the additional argument *cluster.only* in pam must remain as FALSE, which is the default.

Value

sompam: Object of class **pam**. See ?pam.object for details.

sompam.rast: Multi-layer SpatRaster, as in rast. The first raster layer corresponds to the SOM's winning units. The second raster layer corresponds to the clustered SOM's codebook vectors by PAM.

See Also

Other Functions for Landscape Stratification: som_gap(), strata()

Examples

```
require(terra)
# Multi-layer SpatRaster with topographic variables
p <- system.file("exdat", package = "rassta")
ft <- list.files(path = p, pattern = "^height|^slope|^wetness",
                 full.names = TRUE
```

```
                )
t <- rast(ft)
# Scale topographic variables (mean = 0, StDev = 1)
ts <- scale(t)
# Self-organizing map and gap statistic for optimum k
set.seed(963)
tsom <- som_gap(var.rast = ts, xdim = 8, ydim = 8, rlen = 150,
                mode = "online", K.max = 6, B = 300, spaceH0 = "original",
                method = "globalSEmax"
               )
# Optimum k
tsom$Kopt
# PAM clustering of topographic SOM's codebook vectors
tpam <- som_pam(ref.rast = t[[1]], kohsom = tsom$SOM, k = tsom$Kopt)
# Plot topographic variables, SOM grid and PAM clustering
if(interactive()){plot(c(t, tpam$sompam.rast))}
```

`strata` *Create Stratification Units*

Description

Stratification units are created from the spatial intersection of raster layers representing different sets of classification units. Each set of classification units is related to a particular landscape factor (e.g., topography, climate) or to a particular spatial scale for a single landscape factor (e.g., micro-climate, macro-topography). Each resulting stratification unit is considered to represent a distinct landscape configuration in terms of multiple landscape factors/factor scales (represented by the classification units). This function automatically assigns a unique numeric code to each stratification unit. For *x* stratification unit, the numeric code represents the unique combination of classification units whose spatial intersection resulted in *x*. See **Examples** to get a better idea of the logic behind the code assignment process.

Usage

```
strata(cu.rast, to.disk = FALSE, outdir = ".", su.name, ...)
```

Arguments

`cu.rast`	SpatRaster, as in `rast`. Multi-layer SpatRaster for which each layer represents a set of classification units for a particular landscape factor or factor scale. Integer cell values (i.e., numeric identifiers) are expected.
`to.disk`	Boolean. Write the resulting raster layer of stratification units to disk? Default: FALSE
`outdir`	Character. If *to.disk* = *TRUE*, string specifying the path for the output raster layer of stratification units. Default: "."
`su.name`	Character. If *to.disk* = *TRUE*, file name (including extension) for the output raster layer of stratification units.
`...`	Additional arguments as for `writeRaster`.

Details

When printing *su.rast$code.mult*, the output shows the multiplier used for each landscape factor/factor scale. From this output, one can manually replicate the creation of stratification units through simple raster algebra. To do so, a weighted sum of the SpatRasters containing the classification units for each landscape factor/factor scale should be performed using the multipliers as weights. Note that the weights do not imply relative importance. The weights are required only to preserve a logical structure of the landscape factors/factor scales in the resulting numeric code.

Value

su.rast: Single-layer SpatRaster representing the stratification units occurring across geographic space. The cell values in this raster layer represents the numeric codes of stratification units.

code.mult: Multipliers used for the creation of the numeric codes. See **Details**.

See Also

Other Functions for Landscape Stratification: `som_gap()`, `som_pam()`

Examples

```
require(terra)
p <- system.file("exdat", package = "rassta")
# Multi-layer SpatRast with classification units (Cus)
## Three sets (i.e., landscape factors): geology, climate and topography
fcu <- list.files(path = p,
                  pattern = "geology.tif|climate.tif|topography.tif",
                  full.names = TRUE
                 )
cu <- terra::rast(fcu)
# Stratification units (SUs)
su <- strata(cu.rast = cu)
# Plot the stratification units
if(interactive()){plot(su$su.rast, type = "classes")}
#
# Note code structure from SUs and corresponding values from CUs
z <- c(su$su.rast, cu)[46,61]   # Example of one cell (row = 45, column = 45)
su$code.mult                    # Multipliers
z[c("SU", names(su$code.mult))] # Code structure

# Note what happens when some landscape factors have cell values greater...
#... than 1 digit (i.e., more than 9 distinct classification units)
cu <- c(cu[[1]], cu[[2]]^4, cu[[3]]^2)
su <- strata(cu.rast = cu)
su$code.mult
c(su$su.rast, cu[[names(su$code.mult)]])[46,61]
```

Index