

Package ‘nhSDM’

May 1, 2019

Title Tools for Natural Heritage SDMs

Date 2019-05-01

Version 0.0.4-3

Description Miscellaneous tools for working with Species Distribution Modelling (SDM) input and output data.

Depends R (>= 3.5.0), raster, sf

Imports methods, sp, gdalUtils, stringi, RSQLite

Suggests dplyr, lwgeom

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 6.1.1

R topics documented:

nh_best	1
nh_burn	3
nh_group	4
nh_patchdrop	5
nh_rasterize	6
nh_sample	7
nh_stack	8
nh_stack_resample	9

Index	11
--------------	-----------

nh_best	<i>Extract areas with best (highest value) as polygons from an SDM prediction raster, with optional feature mask</i>
---------	--

Description

If `spf` is given, areas intersecting these features (plus a buffer, if `min.dist` is specified) will not be included in the returned polygons.

Usage

```
nh_best(rast, spf = NULL, top.percent = NULL, min.size = NULL,
        min.dist = NULL, num.patches = NULL, rank.by = "area")
```

Arguments

<code>rast</code>	input raster model output with continuous values
<code>spf</code>	input spatial features (sp or sf spatial object); if given, it can be used to modify areas selected from <code>rast</code>
<code>top.percent</code>	numeric; percent (e.g.; 0.01 = 0.01%) of highest cell values in raster to extract
<code>min.size</code>	numeric; optional minimal area of an extracted polygon
<code>min.dist</code>	numeric; optional minimal distance from <code>spf</code> for patches. Default is 0 (can be adjacent to 'spf')
<code>num.patches</code>	numeric; optional number of patches to return, using <code>rank.by</code> criteria
<code>rank.by</code>	character; used for ranking patches when <code>num.patches</code> is not null; either 'area' (default) or 'value'

Details

`top.percent` and `min.size` will both be derived from `spf` if they are null and `spf` is given. In this case, `top.percent` will be set to be equal to `spf`'s cell coverage (prevalence relative to non-NA areas in raster). `min.size` will be set to the area of the smallest feature in `spf`. If `spf` are line or point features, this will be zero.

`rank.by` only subsets the outputs if `num.patches` is set. `rank.by` must be set to "value" to return the mean prediction value of the patch.

For better performance, crop the raster prior to running, and/or use a very low `top.percent` value, to return a smaller proportion of the cells as polygons.

Value

sp or sf object (polygons)

Author(s)

David Bucklin

Examples

```
## Not run:
spf <- sf::st_read("inputs/species/ambymabe/polygon_data/ambymabe.shp")
rast <- raster::raster("outputs/ambymabe/grids/ambymabe_20171018_130837.tif")
rast <- raster::crop(rast, spf)
best_polys <- nh_best(rast, spf)

## End(Not run)
```

nh_burn	<i>Add areas represented by features ('burn-in') to a binary raster SDM output</i>
---------	--

Description

Takes spatial features, a continuous raster (values between 0 and 1), and returns a classified binary (0/1) raster. Areas greater than `orig.thresh` are set to 1, all others 0. Areas intersecting `spf` features are assigned a value of 1 in the returned classified raster. See details for default calculation of `orig.thresh` and usage of `buffer`.

Usage

```
nh_burn(spf, rast, orig.thresh = NULL, buffer = 0,
        return.thresh = FALSE, ...)
```

Arguments

<code>spf</code>	input spatial features (sp or sf spatial object)
<code>rast</code>	input raster model output (values between 0 and 1)
<code>orig.thresh</code>	numeric between 0 and 1; 'global' threshold value to apply to raster
<code>buffer</code>	numeric; spatial buffer around <code>spf</code> to include in burn-in
<code>return.thresh</code>	logical; whether to return thresholds along with raster in a list
<code>...</code>	Other arguments as to <code>raster::writeRaster</code>

Details

When `buffer` is used or `orig.thresh` is not provided, a minimum cell value (`min.cell`) across all `spf` is calculated. The `min.cell` value will be used as a threshold for the full raster, or just areas within `buffer` distance of `spf`, if `buffer` is greater than 0.

If `buffer` is greater than 0 and `orig.thresh` is not `NULL`, `orig.thresh` is used as the threshold for the full raster. Additionally, cells within `buffer` distance of `spf` which are greater than the calculated `min.cell` value are also set to 1. If a given `orig.thresh` is lower than the calculated `min.cell` value, `buffer` will have no impact on the output.

The default is to return the raster only. If `return.thresh = TRUE`, the function will return a list with 3 named objects: `rast`, the output raster; `orig.thresh`, the global threshold (if used); `min.cell`, the minimum cell value threshold (if used).

Value

RasterLayer

Author(s)

David Bucklin

Examples

```
## Not run:
spf <- st_read("ambymabe/polygon_data/ambymabe.shp")
rast <- raster("ambymabe/grids/ambymabe_20171018_130837.tif")
class_burn <- nh_burn(spf, rast, 0.75, 250, filename = "model_classified.tif", datatype = "INT2U")

## End(Not run)
```

nh_group

Group spatial features using a defined separation distance

Description

Input features are allocated into new groups depending on whether they lie within the separation distance of another input feature.

Usage

```
nh_group(spf, sep.dist = 0, union = FALSE)
```

Arguments

spf	input spatial features (sp or sf spatial object)
sep.dist	separation distance with which to define groups (see description)
union	whether to union output groups into multi-features

Details

The grouping is done by input ID, so multi-features are considered one input feature (they will not be be ungrouped).

The separation distance `sep.dist` is numeric and in the units of `spf`'s coordinate system, unless the coordinate system uses latitude/longitude as the unit (e.g. WGS 84). In these cases, geodesic distances will be used and `sep.dist` should be specified in meters.

A column 'group' will be added to the output features. Specifying `union = TRUE` will output one (multi)feature per group, meaning original attributes will be discarded - only 'group' and 'count' (the number of original features in the group) will be returned. This feature requires the package `dplyr` to be installed.

Value

sp or sf object (same as input)

Author(s)

David Bucklin

Examples

```
## Not run:
spf <- rgdal::readOGR("ambymabe/polygon_data", "ambymabe")
spg <- nh_group(spf, 1000)

## End(Not run)
```

nh_patchdrop	<i>Remove contiguous patches smaller than a given patch size from binary output</i>
--------------	---

Description

Takes a binary/thresholded raster (values either NA, 0, or 1), and returns a binary raster. Clumps of contiguous cells with the value (1) that are smaller than the `min.patch` size are given a value of 0 (if 0 values are present in the input 'rast'); otherwise, they are given an NA value.

Usage

```
nh_patchdrop(spf = NULL, rast, min.patch = NULL, directions = 8, ...)
```

Arguments

<code>spf</code>	input spatial features (sp or sf spatial object)
<code>rast</code>	input binary raster output (values either NA/0 or 1)
<code>min.patch</code>	area of minimum patch size, in area units used in <code>rast</code>
<code>directions</code>	Integer. Which cells are considered adjacent? Should be 8 (default; Queen's case) or 4 (Rook's case). From <code>raster::clump</code>
<code>...</code>	Other arguments as to <code>raster::writeRaster</code>

Details

If 'spf' is given, the smallest feature's area will be used to derive a `min.patch` value, and any given 'min.patch' is ignored. If 'spf' is not given, a 'min.patch' value must be given, in area units of the input raster.

Value

RasterLayer

Author(s)

David Bucklin

Examples

```
## Not run:
spf <- st_read("ambymabe/polygon_data/ambymabe.shp")
rast <- raster("screen_lpsh/thumb/ambymabe.tif")

# use minimum patch size from presence features
rast_contig_minpres <- nh_patchdrop(spf, rast)

# use a minimum patch size of 10000 (in units from 'rast')
rast_contig_10km <- nh_patchdrop(rast = rast, min.patch = 10000,
  filename = "rast_contig_10km.tif", datatype = "INT2U")

## End(Not run)
```

nh_rasterize	<i>Convert vector format SDM prections to raster format</i>
--------------	---

Description

Requires spatial features, a template raster, and prediction values vector. The buffer is optional; when provided, the buffer distance can be provided as a single numeric value or a vector matching length of spf, for a variable buffer by feature.

Usage

```
nh_rasterize(spf, rast, pred.vals, buffer = 0, priority = NULL,
             rast.out = NULL, ...)
```

Arguments

spf	input spatial features (model predictions; sp or sf spatial object)
rast	input raster template
pred.vals	prediction values for spf features
buffer	numeric; spatial buffer around spf to include in burn-in
priority	numeric; vector of priority values for feature rasterization (higher values have priority)
rast.out	Output raster file name (with file extension)
...	Additional arguments to writeRaster (e.g. overwrite)

Details

Buffer units should be given in the units of rast. Make sure to keep in mind the resolution of the raster when choosing a buffer. If rasterizing lines and a one-cell width is desired, do not use a buffer.

A vector can be given to 'priority' for sorting prior to rasterization, where higher values have priority. When priority = NULL (default), priority will be defined as the pred.vals.

If rast.out is not specified, the raster will remain in temp folder.

Value

RasterLayer

Author(s)

David Bucklin

Examples

```
## Not run:
spf <- st_read("acipoxyr/shapefiles/acipoxyr_20180105_133929_results.shp")
rast <- raster("template.tif")

# rasterize
bla <- nh_rasterize(spf, rast, pred.vals = spf$prbblty, buffer = spf$strord*15)

## End(Not run)
```

nh_sample*Create points in features in reference raster cells*

Description

For each spatial feature, a given number (`num.samps`) of points are created in cells that the feature intersects.

Usage

```
nh_sample(spf, rast, num.samps = NULL, replace = FALSE,  
          force.min = FALSE)
```

Arguments

<code>spf</code>	input spatial features (sp or sf spatial object)
<code>rast</code>	raster dataset with extent overlapping <code>spf</code>
<code>num.samps</code>	number of samples to create in each feature (see details)
<code>replace</code>	whether to sample with or without replacement
<code>force.min</code>	whether to force <code>num.samps</code> points in features, even if they are duplicates

Details

`num.samps` can be a proportion (a decimal value < 1), single integer, or vector of integers equal to length of `spf` indicated the number of samples to take from each feature. If left `NULL`, `num.samps` will be set to the number of cells `[n]` intersecting the feature. If a proportion is given (e.g., 0.5), then `[n * num.samps]` will be returned. If a single integer is given, `num.samps` points will be sampled in each feature.

When `replace = FALSE` and `force.min = FALSE` (defaults), each cell can only contain one point (across each feature, and the entire returned set of points). In this case, when the number of samples points to create exceeds the number of unique cells intersected by a given feature, the number of samples for that feature equals the number of cells. If `replace = TRUE`, sampling is done with replacement and duplicates may be taken. The special case `replace = FALSE` and `force.min = TRUE` will always return `num.samps` per feature. It only produces duplicates if `num.samps` exceeds the number of cells intersecting the feature, in which case it will replicate the samples until `num.samps` is reached.

If CRS do not match, the features will be transformed to the CRS of the raster.

A column ‘`feat.id`’ is added to the output point features to indicate the row number of the feature that the point was generated within.

Value

sp or sf object (points)

Author(s)

David Bucklin

Examples

```
## Not run:
r<-raster::raster("AnnMnTemp.tif")
spf <- rgdal::readOGR("ambymabe/polygon_data", "ambymabe")
# can also use sf: spf <- sf::st_read("ambymabe/polygon_data/ambymabe.shp")
spf.samps <- nh_sample(spf, r, num.samps)

## End(Not run)
```

nh_stack

Stack multiple binary SDM rasters into one raster layer

Description

Takes a rastfiles of binary raster's filenames (values 0/1), a template raster covering the extent desired for the stack, and optionally the codes (names) to use for the rasters. Returns a single raster layer (factor type with an attribute table).

Usage

```
nh_stack(rastfiles, rast, codes = NULL, return.table = TRUE)
```

Arguments

<code>rastfiles</code>	raster file names; character vector
<code>rast</code>	raster template dataset
<code>codes</code>	species codes; character vector. If given, must match length of <code>rastfiles</code>
<code>return.table</code>	Whether to return a table with <code>nh_stack</code> unique values, species codes, and file-names

Details

The raster attribute table (accessed using `levels()`) has four columns: 'ID': the unique integer value in the raster; 'VALUE': the internal `nh_stack` unique value for that 'ID', 'ALLCODES': the identity of species codes, pasted in a character vector seperated by ';', 'ALLCODES_CT': the number of unique codes for that value.

If `codes` is not given, the raster layer name will be used as the layer's code. If these are not unique, the internal `nh_stack` unique value will be pasted to the end of the original code, and a message will be printed.

All rasters must have the same projection and resolution, though they can have different extents - the processing extent is defined by `rast`. To summarize all rasters across their entire extents, template should essentially be the union (mosaic) of all raster extents.

When `return.table` is TRUE, a list with two objects (1), the stack raster, and (2) a summary table of included rasters is returned (with internal `nh_stack` unique values, species codes, and file names). This table is required for resampling the stack (i.e. with `nh_stack_resample`).

Value

RasterLayer

Author(s)

David Bucklin

Examples

```
## Not run:
rast<-raster::raster("project_mask.tif")
list <- sort(list.files(paste0(rastout, t, "/thumb"), full.names = T,
  pattern = "^[:lower:]{5,12}\\\\.tif$"))

stack <- nh_stack(list, rast)
# view raster attribute table
levels(stack)[[1]]

## End(Not run)
```

nh_stack_resample	<i>Resample a raster from nh_stack to a lower (coarser) resolution</i>
-------------------	--

Description

Takes an output raster from `nh_stack`, and returns a lower-resolution version, with recalculated species assemblages for the larger cells. New values are "aggregated" by `fact`, the number of cells to aggregate in the x/y dimensions (see `?raster::aggregate`).

Usage

```
nh_stack_resample(rast, lookup, fact = 10, spf = NULL)
```

Arguments

<code>rast</code>	raster output from <code>nh_stack</code>
<code>lookup</code>	lookup table from <code>nh_stack</code>
<code>fact</code>	aggregation factor, in number of cells (see <code>?raster::aggregate</code>)
<code>spf</code>	Optional vector spatial features to use for aggregation (sp or sf-class polygons). If supplied, <code>fact</code> will be ignored

Details

You can also provide polygons (sp or sf-class) to `spf`, over which to aggregate species assemblages. The polygons intersecting areas with data in `rast` are returned, with columns identifying species codes and counts. This method will fail with large rasters (see `raster::zonal`), in which case processing subsets of the stack raster is advised. Polygons will be returned in their original projection, but processing internally is done in the raster's projection.

Value

RasterLayer

Author(s)

David Bucklin

Examples

```
## Not run:
stack <- nh_stack(list, rast, return.table = TRUE)

# resample from 30m to 990m (~1km) resolution
stack1km <- nh_stack_resample(stack[[1]], stack[[2]], fact = 33)

# view species count raster
ct <- deratify(r2, att = "ALLCODES_CT")
plot(ct)

## End(Not run)
```

Index

nh_best, [1](#)
nh_burn, [3](#)
nh_group, [4](#)
nh_patchdrop, [5](#)
nh_rasterize, [6](#)
nh_sample, [7](#)
nh_stack, [8](#)
nh_stack_resample, [9](#)