Package 'rangemap'

October 14, 2022

Type Package

Title Simple Tools for Defining Species Ranges

Version 0.1.18

Maintainer Marlon E. Cobos <manubio13@gmail.com>

Date 2021-09-03

Description A collection of tools to create species range maps based on
occurrence data, statistics, and spatial objects. Other tools in this
collection can be used to analyze the environmental characteristics of
the species ranges. Plotting options to represent results in various
manners are also available. Results obtained using these tools can be
used to explore the distribution of species and define areas of occupancy
and extent of occurrence of species. Other packages help to explore species
distributions using distinct methods, but options presented in this set of
tools (e.g., using trend surface analysis and concave hull polygons) are
exclusive. Description of methods, approaches, and comments for some of the
tools implemented here can be found in:
IUCN (2001) <https://portals.iucn.org/library/node/10315>,
Peterson et al. (2011) <https://www.degruyter.com/princetonup/view/title/506966>,
and Graham and Hijmans (2006) <doi:10.1111/j.1466-8238.2006.00257.x>.

URL https://github.com/marlonecobos/rangemap

BugReports https://github.com/marlonecobos/rangemap/issues

Imports concaveman (>= 1.0), dplyr (>= 0.8), maps (>= 3.3), maptools
(>= 0.9), methods, rnaturalearthdata (>= 0.1), raster (>= 3.0),
rgdal (>= 1.4), rgeos (>= 0.5), rgl (>= 0.100), scales (>=
1.1), sf (>= 0.6), spatial (>= 7.3), sp (>= 1.3)

Depends R (>= 3.5.0)

License GPL-3

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Suggests knitr, rmarkdown

VignetteBuilder knitr

NeedsCompilation no

Author Marlon E. Cobos [aut, cre] (<https://orcid.org/0000-0002-2611-1767>),
Vijay Barve [aut] (<https://orcid.org/0000-0002-4852-2567>),
Narayani Barve [aut] (<https://orcid.org/0000-0002-7893-8774>),
Alberto Jimenez-Valverde [aut]
(<https://orcid.org/0000-0001-9962-2106>),
Claudia Nuñez-Penichet [aut] (<https://orcid.org/0000-0001-7442-8593>)

Repository CRAN

Date/Publication 2021-09-03 21:10:02 UTC

R topics documented:

adm_area_names 3
adm_boundaries 4
aoo 4
buffer_range 5
clusters 6
country_codes 7
cvehull_range 8
cxhull_range 8
eoo 9
GADM_spoly 10
geobuffer_points 11
hull_polygon 12
keep_big_polygons 13
LAEA_projection 13
north_arrow 14
occ_d 15
occ_f 16
occ_p 16
occ_train 17
plot_ranges 18
rangemap 19
rangemap_boundaries 19
rangemap_buffer 22
rangemap_enm 23
rangemap_explore 26
rangemap_hull 27
rangemap_plot 29
rangemap_tsa 31
ranges_emaps 33
ranges_espace 35
simple_wmap 36
spdf_range 37
sp_model 37

sp_range 38
summary 38
variables 39

Index **40**

`adm_area_names` *List of names of administrative areas form the GADM data base*

Description

A dataset containing names of all the available administrative areas from the GADM data base. Names describe distinct administrative areas in five levels.

Usage

```
adm_area_names
```

Format

A data frame with 41535 rows and 4 columns.

ISO3 character, country code.

NAME_0 character, name of administrative areas at level 0.

NAME_1 character, name of administrative areas at level 1.

NAME_2 character, name of administrative areas at level 2.

Source

https://gadm.org/

Examples

```
data("adm_area_names", package = "rangemap")
adm_area_names[1:10, 1:4]

# Country names: Level 0
unique(adm_area_names$NAME_0)

# Provinces of a country: Level 1
as.character(unique(adm_area_names[adm_area_names$NAME_0 == "Ecuador", "NAME_1"]))

# Some administrative areas at level 2 of India
l2_India <- as.character(unique(adm_area_names[adm_area_names$NAME_0 == "India",
                                               "NAME_2"]))
```

`adm_boundaries` *Example SpatialPolygonsDataFrame of country boundaries*

Description

A SpatialPolygonsDataFrame of 9 countries from South America.

Usage

```
adm_boundaries
```

Format

SpatialPolygonsDataFrame with 9 features.

features SpatialPolygons, 9.

data.frame 9 rows, 11 columns.

Examples

```
data("adm_boundaries", package = "rangemap")
adm_boundaries
```

`aoo` *Area of occupancy of a species as defined by the IUCN*

Description

Area of occupancy of a species as defined by the IUCN

Usage

```
aoo(occ_pr, species)
```

Arguments

`occ_pr`	SpatialPointsDataFrame of occurrence records. Projection must be one that allows safe calculation of areas (e.g., Lambert Azimuthal Equal Area).
`species`	(character) scientific name of the species.

Value

A list containing a SpatialPolygonsDataFrame of the area of occupancy, and a vector with the areas in km2 of the spatial polygons resulted.

Examples

```
# data
data("occ_p", package = "rangemap")
occ <- unique(occ_p)
WGS84 <- sp::CRS("+init=epsg:4326")
occ_sp <- sp::SpatialPointsDataFrame(coords = occ[, 2:3], data = occ,
                                     proj4string = WGS84)

LAEA <- LAEA_projection(spatial_object = occ_sp)
occ_pr <- sp::spTransform(occ_sp, LAEA)

sp <- as.character(occ[1, 1])

# AOO
aoo_pe <- aoo(occ_pr = occ_pr, species = sp)
```

`buffer_range` *Example of sp_range* object based on buffers*

Description

A sp_range_iucn object containing the results of the function `rangemap_buffer`.

Usage

```
buffer_range
```

Format

sp_range_iucn with 6 slots.

name character, name identifying the origin of the object.

summary data.frame, summary of results.

species_range SpatialPolygonsDataFrame of species range.

species_unique_records SpatialPointsDataFrame of species occurrences.

extent_of_occurrence SpatialPolygonsDataFrame of species extent of occurrence.

area_of_occupancy SpatialPolygonsDataFrame of species area of occupancy.

Examples

```
data("buffer_range", package = "rangemap")
summary(buffer_range)
```

clusters *Finds clusters for SpatialPointsDataFrame based on distinct methods*

Description

Finds clusters for SpatialPointsDataFrame based on distinct methods

Usage

```
clusters(occ_pr, cluster_method = "hierarchical", split_distance,
         n_k_means, set_seed = 1, verbose = TRUE)
```

Arguments

`occ_pr`	SpatialPointsDataFrame of occurrence records. Projection must be one that allows safe calculation of distances (e.g., Azimuthal equidistant)
`cluster_method`	(character) name of the method to be used for clustering the occurrences. Options are "hierarchical" and "k-means"; default = "hierarchical". See details for more information on the two available methods.
`split_distance`	(numeric) distance in meters that will limit connectivity among hull polygons created with chunks of points separated by long distances. This parameter is used when `cluster_method` = "hierarchical".
`n_k_means`	(numeric) number of clusters in which the species occurrences will be grouped when using the "k-means" `cluster_method`.
`set_seed`	(numeric) integer value to specify a seed. Default = 1.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

`cluster_method` must be chosen based on the spatial configuration of the species occurrences. Both methods make distinct assumptions and one of them may perform better than the other depending on the spatial pattern of the data.

The k-means method, for example, performs better when the following assumptions are fulfilled: Clusters are spatially grouped—or “spherical” and Clusters are of a similar size. Owing to the nature of the hierarchical clustering algorithm it may take more time than the k-means method. Both methods make assumptions and they may work well on some data sets, and fail on others.

Another important factor to consider is that the k-means method always starts with a random choice of cluster centers, thus it may end in different results on different runs. That may be problematic when trying to replicate your methods. With hierarchical clustering, most likely the same clusters can be obtained if the process is repeated.

For more information on these clustering methods see Aggarwal and Reddy (2014) https://goo.gl/RQ2ebd.

Value

A SpatialPointsDataFrame with an extra column in data defining clusters.

Examples

```
# data
data("occ_p", package = "rangemap")

# preparing spatial points
occ <- as.data.frame(unique(occ_p))
WGS84 <- sp::CRS("+init=epsg:4326")
occ_sp <- sp::SpatialPointsDataFrame(coords = occ[, 2:3], data = occ,
                                     proj4string = WGS84)

# reprojecting for measuring distances
LAEA <- LAEA_projection(spatial_object = occ_sp)
occ_pr <- sp::spTransform(occ_sp, LAEA)

# clustering
occ_clus <- clusters(occ_pr, cluster_method = "k-means", n_k_means = 2)
```

`country_codes` *List of country names and ISO codes*

Description

A dataset containing codes for identifying countries according to ISO norms.

Usage

```
country_codes
```

Format

A data frame with 247 rows and 4 columns.

Country_or_Area_Name character, names.

ISO_ALPHA.2_Code character, country code.

ISO_ALPHA.3_Code character, country code.

ISO_Numeric_Code_UN_M49_Numerical_Code numeric, country numeric codes.

Source

https://www.nationsonline.org/oneworld/country_code_list.htm

Examples

```
data("country_codes", package = "rangemap")
head(country_codes)
```

`cvehull_range` *Example of sp_range* object based on concave hulls*

Description

A sp_range_iucn object containing the results of the function `rangemap_hull`.

Usage

```
cvehull_range
```

Format

sp_range_iucn with 6 slots.

name character, name identifying the origin of the object.

summary data.frame, summary of results.

species_range SpatialPolygonsDataFrame of species range.

species_unique_records SpatialPointsDataFrame of species occurrences.

extent_of_occurrence SpatialPolygonsDataFrame of species extent of occurrence.

area_of_occupancy SpatialPolygonsDataFrame of species area of occupancy.

Examples

```
data("cvehull_range", package = "rangemap")
summary(cvehull_range)
```

`cxhull_range` *Example of sp_range* object based on convex hulls*

Description

A sp_range_iucn object containing the results of the function `rangemap_hull`.

Usage

```
cxhull_range
```

Format

sp_range_iucn with 6 slots.

name character, name identifying the origin of the object.

summary data.frame, summary of results.

species_range SpatialPolygonsDataFrame of species range.

species_unique_records SpatialPointsDataFrame of species occurrences.

extent_of_occurrence SpatialPolygonsDataFrame of species extent of occurrence.

area_of_occupancy SpatialPolygonsDataFrame of species area of occupancy.

Examples

```
data("cxhull_range", package = "rangemap")
summary(cxhull_range)
```

---

eoo — *Extent of occurrence of a species based on convex hull polygons*

Description

Extent of occurrence of a species based on convex hull polygons

Usage

```
eoo(occurrences, polygons)
```

Arguments

`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84).
`polygons`	SpatialPolygons object to clip convex hulls to these limits. Projection must be WGS84 (EPSG:4326).

Details

Areas are calculated in square kilometers using the Lambert Azimuthal Equal Area projection, centered on the centroid of occurrence points given as inputs.

Value

A list containing a SpatialPolygonsDataFrame of the extent of occurrence, and a vector with the areas in km2 of the spatial polygons resulted. Projection of resulting spatial object is Lambert Azimuthal Equal Area.

Examples

```
# occurrences
data("occ_f", package = "rangemap")
occ <- unique(occ_f)

# polygons
poly <- simple_wmap("simple", "Cuba")
LAEA <- LAEA_projection(occ[, 2:3])
poly_pr <- sp::spTransform(poly, LAEA)

# to fix topology problems
poly_pr <- rgeos::gBuffer(poly_pr, width = 0)

# EOO
eoo_pe <- eoo(occurrences = occ, polygons = poly_pr)
```

`GADM_spoly` *Get SpatialPolygons of countries at distinct administrative levels*

Description

Get SpatialPolygons of countries at distinct administrative levels

Usage

```
GADM_spoly(country_code, boundary_level, keep_data = FALSE)
```

Arguments

`country_code`	(character) code of country or countries of interest.
`boundary_level`	(numeric) level of administrative division to be considered.
`keep_data`	(logical) whether or not to keep downloaded files. Default = `FALSE`.

Value

A SpatialPolygonsDataFrame from the GADM database at the level selected. If an error occurs when downloading any of the spatial objects based on `country_code`, the result is NULL.

Examples

```
map_gadm <- GADM_spoly(country_code = "UY", boundary_level = 0)
```

geobuffer_points *Geodesic buffer for unprojected points*

Description

geobuffer_points helps in creating geodesic buffers of points represented by longitude and latitude coordinates.

Usage

```
geobuffer_points(data, radius, by_point = FALSE, n_segments = 100,
                 wrap_antimeridian = FALSE)
```

Arguments

data	matrix of geographic unprojected coordinates (i.e., in EPSG:4326). Columns must be "longitude" and "latitude", in that order.
radius	(numeric) radius of buffer in meters.
by_point	(logical) whether or not to do buffers by point. If FALSE, the default, buffer polygons that overlap will be dissolved to obtain an only feature. Default = 100.
n_segments	(numeric) number of segments to approximate a circle.
wrap_antimeridian	(logical) whether or not to wrap buffers in the antimeridian when they overpass it.

Details

The process is done using an algorithm that calculates the buffer in the North Pole and then rotates this buffer into the actual location. The rotation is effected by converting the original buffer to geocentric Cartesian (XYZ) coordinates. A matrix multiplication helps to rotate those coordinates along the Prime Meridian to the target latitude, converting the coordinates back to Geographic (WGS84). Then the buffer is spun around the Earth's axis by adding the target longitude to each second coordinate.

The algorithm was developed by a moderator of the Geographic Information Systems Stack Exchange (online community). More details are available available at the following site.

Value

A SpatialPolygons object of buffered points. Final projection is WGS84 (EPSG:4326).

Examples

```
#data
data("occ_p", package = "rangemap")
coords <- occ_p[, 2:3]
```

```
# buffers
bufferp <- geobuffer_points(data = coords, radius = 25000)

sp::plot(bufferp, axes = TRUE)
```

---

hull_polygon *Convex or concave hull polygons from spatial points*

Description

Convex or concave hull polygons from spatial points

Usage

```
hull_polygon(occ_pr, hull_type = "convex", concave_distance_lim = 5000,
             verbose = TRUE)
```

Arguments

`occ_pr`	SpatialPoints* object containing geographic points to be used to create hull polygons. This spatial object must be projected to a system with the argument "+units=m".
`hull_type`	(character) type of hull polygons to be created. Available options are: "convex" and "concave". Default = "convex".
`concave_distance_lim`	(numeric) distance, in meters, to be passed to the length_threshold parameter of the `concaveman` function. Default = 5000. Ignored if `hull_type` is not "concave".
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Value

A SpatialPolygons object with the hull polygon. If the number of points in occ_pr is 1 or 2 a SpatialPointsDataFrame object is returned.

Examples

```
# data
data("occ_p", package = "rangemap")

# preparing spatial points
occ <- as.data.frame(unique(occ_p))
WGS84 <- sp::CRS("+init=epsg:4326")
occ_sp <- sp::SpatialPointsDataFrame(coords = occ[, 2:3], data = occ,
                                     proj4string = WGS84)

# reprojecting
LAEA <- LAEA_projection(spatial_object = occ_sp)
```

```
occ_pr <- sp::spTransform(occ_sp, LAEA)

# convex hull polygon
cvx_hull <- hull_polygon(occ_pr, hull_type = "convex")
```

`keep_big_polygons` *Exclude small polygons from SpatialPolygons object*

Description

Exclude small polygons from SpatialPolygons object

Usage

```
keep_big_polygons(polygons, threshold_size)
```

Arguments

`polygons`	SpatialPolygonsDataFrame object.
`threshold_size`	(numeric) threshold value of area to determine whether polygons are big or not. Areas must be according to projection of `polygons`.

Value

A SpatialPolygonsDataFrame with polygons with areas above `threshold_size`.

Examples

```
data("spdf_range", package = "rangemap")
sp::plot(spdf_range)

big_polys <- keep_big_polygons(polygons = spdf_range, threshold_size = 0.2)
sp::plot(big_polys)
```

`LAEA_projection` *Prepare Equal-Area or Equidistant Projection*

Description

Prepare Equal-Area or Equidistant Projection

Usage

```
LAEA_projection(occurrences = NULL, spatial_object = NULL)

AED_projection(occurrences = NULL, spatial_object = NULL)
```

Arguments

`occurrences`	matrix or data.frame containing coordinates to serve as a reference for the center of the projection. Columns must be: "longitude" and "latitude", in that order.
`spatial_object`	Spatial* objects, Points or Polygons, to be used to calculate a reference for the center of the projection. Projection must be WGS84 (EPSG:4326).

Details

If arguments are not defined projection is centered in 0, 0 for longitude and latitude.

Value

An object of class CRS.

Examples

```
LAEA_projection()

data("occ_p", package = "rangemap")
occ <- unique(occ_p)[, 2:3]
AED_projection(occc)
```

`north_arrow` *Helper to add north arrow to map plots*

Description

north_arrow plots a North arrow in user defined places in a map.

Usage

```
north_arrow(position = "topright", xlim = NULL, ylim = NULL)
```

Arguments

`position`	(character or numeric) position of the North arrow. If character, options are: "topright", "topleft", "bottomleft", or "bottomright". Default = "topright".
`xlim`	(numeric) vector of two numbers indicating the x limits of the plotting area. Default = `NULL`.
`ylim`	(numeric) vector of two numbers indicating the y limits of the plotting area. Default = `NULL`.

Value

Plot of a simple North arrow located in the `position` of the plot specified.

Examples

```
# simple plot
plot(1:10, 1:10, col = "transparent")

# north arrows
north_arrow(position = "topright")
north_arrow(position = "bottomright")
```

`occ_d` *Occurrence records for the species Dasypus kappleri*

Description

A dataset containing geographic coordinates of a South American armadillo.

Usage

```
occ_d
```

Format

A data frame with 306 rows and 3 columns.

name character, species scientific name.

longitude numeric, longitude values.

latitude numeric, latitude values.

Source

https://www.gbif.org/

Examples

```
data("occ_d", package = "rangemap")
head(occ_d)
```

occ_f — *Occurrence records for the species Peltophryne fustiger*

Description

A dataset containing geographic coordinates of the Giant Cuban Toad.

Usage

```
occ_f
```

Format

A data frame with 73 rows and 3 columns.

name character, species scientific name.

longitude numeric, longitude values.

latitude numeric, latitude values.

Source

https://www.gbif.org/

Examples

```
data("occ_f", package = "rangemap")
head(occ_f)
```

occ_p — *Occurrence records for the species Peltophryne empusa*

Description

A dataset containing geographic coordinates of a Caribbean toad.

Usage

```
occ_p
```

Format

A data frame with 182 rows and 3 columns.

name character, species scientific name.

longitude numeric, longitude values.

latitude numeric, latitude values.

Source

https://www.gbif.org/

Examples

```
data("occ_p", package = "rangemap")
head(occ_p)
```

occ_train *Occurrence records for the species Amblyomma americanum*

Description

A dataset containing geographic coordinates of a North American tick.

Usage

```
occ_train
```

Format

A data frame with 89 rows and 3 columns.

Species character, species scientific name.

Longitude numeric, longitude values.

Latitude numeric, latitude values.

Source

https://www.gbif.org/

Examples

```
data("occ_train", package = "rangemap")
head(occ_train)
```

plot_ranges *Helper to plot multiple ranges on top of environmental layers*

Description

Helper to plot multiple ranges on top of environmental layers

Usage

```
plot_ranges(sp_ranges, sp_records = NULL, variable, range_colors = NULL,
            color_variable = NULL, xlim = NULL, ylim = NULL)
```

Arguments

`sp_ranges`	list of SpatialPolygonsDataFrame objects representing species ranges.
`sp_records`	a SpatialPointsDataFrame of species occurrences.
`variable`	a RasterLayer representing an environmental variable.
`range_colors`	vector of colors for borders of species ranges. If `NULL`, the default, distinct levels of gray will be used.
`color_variable`	a color palette (a vector of continuous colors generated by functions like heat.colors). If `NULL`, the default, rev(terrain.colors(255)) will be used.
`xlim`	the x limits (x1, x2) of the plot. `NULL` indicates that the range of values to be plotted will define limits.
`ylim`	the y limits (x1, x2) of the plot.

Value

A plot showing species ranges on top of a environmental variable.

Examples

```
# example data
data("buffer_range", package = "rangemap")
data("cxhull_range", package = "rangemap")

ranges <- list(buffer_range@species_range, cxhull_range@species_range)

var <- raster::stack(system.file("extdata", "variables.tif",
                                 package = "rangemap"))[[1]]

# plotting
plot_ranges(ranges, variable = var)
```

`rangemap` *rangemap: Simple Tools for Defining Species Ranges*

Description

rangemap contains a collection of tools to create species range maps based on occurrence data, statistics, and spatial objects. Other tools in this collection can be used to analyze the environmental characteristics of the species ranges. Plotting options to represent results in various manners are also available. Results obtained using these tools can be used to explore the distribution of species and define areas of occupancy and extent of occurrence of species.

rangemap main functions

`rangemap_boundaries`, `rangemap_buffer`, `rangemap_enm`, `rangemap_explore`, `rangemap_plot`, `rangemap_hull`, `rangemap_tsa`, `ranges_emaps`, `ranges_espace`

`rangemap_boundaries` *Species distributional ranges based on administrative boundaries*

Description

rangemap_boundaries generates a distributional range for a given species by considering all the polygons of administrative entities in which the species has been detected. Optionally, representations of the species extent of occurrence (using convex hulls) and the area of occupancy according to the IUCN criteria can also be generated. Shapefiles can be saved in the working directory if it is needed.

Usage

```
rangemap_boundaries(occurrences, adm_areas, country_code, boundary_level = 0,
                    polygons, extent_of_occurrence = TRUE,
                    area_of_occupancy = TRUE, keep_data = FALSE,
                    dissolve = FALSE, final_projection, save_shp = FALSE,
                    name, overwrite = FALSE, verbose = TRUE)
```

Arguments

`occurrences`	(optional) a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84). If not defined, `adm_areas` must be defined and these areas will be used as part of the species range.
`adm_areas`	(optional, character) a vector of names of administrative areas known to be occupied by the species, names depend on the `boundary_level` selected. Check `adm_area_names` for an idea of how to define names in this parameter. If not defined, `occurrences` must exist.

`country_code`	(optional, character) vector of country codes that will be considered when creating the species range. Including neighbor countries may be necessary to obtain better results. Use `rangemap_explore` for a preview of all potential countries involved in the analysis. Codes follow the ISO-3166-1 norm as in function `getData`. If not defined, `polygons` must be included. Ignored if `polygons` is provided.
`boundary_level`	(numeric) level of the administrative boundaries (from 0 to 2; 0 is the country level and higher values indicate finer divisions). Default = 0. Ignored if `polygons` is defined.
`polygons`	(optional) a SpatialPolygonsDataFrame object that will be used instead of boundaries specified in `country_code` to create species ranges based on overlapping of species records with these layer, as well as names defined in `adm_areas`. Projection must be WGS84 (EPSG:4326). If `adm_areas` is defined, `polygons` must have, as part of its data, a field (column) named "adm_names" for selecting extra areas based on names. If `polygons` is defined, arguments `country_code` and `boundary_level` will be ignored.
`extent_of_occurrence`	(logical) whether to obtain the extent of occurrence of the species based on a simple convex hull polygon; default = `TRUE`.
`area_of_occupancy`	(logical) whether to obtain the area of occupancy of the species based on a simple grid of 4 km^2 resolution; default = `TRUE`.
`keep_data`	(logical) if `TRUE` and `polygons` is not defined, data downloaded from the GADM data base will be kept in the working directory. Useful if all or part of the downloaded files will be used in posterior analyses since those files will not be downloaded again and time will be saved. Default = `FALSE`.
`dissolve`	(logical) if `TRUE`, distinct polygons selected as part of the species range will be dissolved for creating simpler shapes, default = `FALSE`. Owing to the high resolution in the GADM data the dissolving process may be time consuming, specially if the species has a broad distribution.
`final_projection`	(character) string of projection arguments for resulting Spatial objects. Arguments must be as in the PROJ.4 documentation. See `CRS-class` for details. If `NULL`, the default, projection used is WGS84 (EPSG:4326).
`save_shp`	(logical) if `TRUE`, shapefiles of the species range, occurrences, extent of occurrence, and area of occupancy will be written in the working directory. Default = `FALSE`.
`name`	(character) valid if `save_shp` = `TRUE`. The name of the shapefile to be exported. A suffix will be added to `name` depending on the object, as follows: species extent of occurrence = "_extent_occ", area of occupancy = "_area_occ", and occurrences = "_unique_records".
`overwrite`	(logical) whether or not to overwrite previous results with the same name. Default = `FALSE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

Data for countries defined in `country_code` are downloaded and loaded using the function `getData`. Information about country codes and names of administrative areas, at distinct levels, can be consulted using: `country_codes` and `adm_area_names`.

Value

A sp_range object (S4) containing: (1) a data.frame with information about the species range, and SpatialPolygons objects of (2) unique occurrences, (3) species range, (4) extent of occurrence, and (5) area of occupancy.

If only `adm_areas` are defined, the result will be a sp_range object (S4) with two elements: (1) a data.frame with information about the species range, and (2) a SpatialPolygons object of the species range.

If `extent_of_occurrence` and/or `area_of_occupancy` = `FALSE`, the corresponding spatial objects in the resulting sp_range object will be empty, an areas will have a value of 0.

If downloading data based on `country_code` fails, the result is `NULL`.

Examples

```
# getting the data
data("occ_d", package = "rangemap")

# checking which countries may be involved in the analysis
rangemap_explore(occurrences = occ_d)

# preparing arguments
level <- 0
adm <- "Ecuador" # Athough no record is on this country, we know it is in Ecuador
countries <- c("PER", "BRA", "COL", "VEN", "ECU", "GUF", "GUY", "SUR", "BOL")

# running using occurrence data
b_range <- rangemap_boundaries(occurrences = occ_d, adm_areas = adm,
                               country_code = countries, boundary_level = level)

summary(b_range)

# running using only names of areas
adm1 <- c("Brazil", "Ecuador", "Peru", "Bolivia", "Colombia", "Venezuela")
b_range1 <- rangemap_boundaries(adm_areas = adm1, country_code = countries,
                                boundary_level = level)

summary(b_range1)
```

rangemap_buffer *Species distributional ranges based on buffered occurrences*

Description

rangemap_buffer generates a distributional range for a given species by buffering provided occurrences using a defined distance. Optionally, representations of the species extent of occurrence (using convex hulls) and the area of occupancy according to the IUCN criteria can also be generated. Shapefiles can be saved in the working directory if it is needed.

Usage

```
rangemap_buffer(occurrences, buffer_distance = 100000, polygons = NULL,
                extent_of_occurrence = TRUE, area_of_occupancy = TRUE,
                final_projection = NULL, save_shp = FALSE, name,
                overwrite = FALSE, verbose = TRUE)
```

Arguments

`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84).
`buffer_distance`	(numeric) distance, in meters, to be used for creating the buffer areas around occurrences, default = 100000.
`polygons`	(optional) a SpatialPolygons* object to clip buffer areas and adjust the species range and other polygons to these limits. Projection must be WGS84 (EPSG:4326). If `NULL`, the default, a simplified world map will be used.
`extent_of_occurrence`	(logical) whether to obtain the extent of occurrence of the species based on a simple convex hull polygon; default = `TRUE`.
`area_of_occupancy`	(logical) whether to obtain the area of occupancy of the species based on a simple grid of 4 km^2 resolution; default = `TRUE`.
`final_projection`	(character) string of projection arguments for resulting Spatial objects. Arguments must be as in the PROJ.4 documentation. See `CRS-class` for details. If `NULL`, the default, projection used is WGS84 (EPSG:4326).
`save_shp`	(logical) if `TRUE`, shapefiles of the species range, occurrences, extent of occurrence, and area of occupancy will be written in the working directory. Default = `FALSE`.
`name`	(character) valid if `save_shp` = `TRUE`. The name of the shapefile to be exported. A suffix will be added to `name` depending on the object, as follows: species extent of occurrence = "_extent_occ", area of occupancy = "_area_occ", and occurrences = "_unique_records".

`overwrite`	(logical) whether or not to overwrite previous results with the same name. Default = `FALSE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

All resulting Spatial objects in the results will be projected to the `final_projection`. Areas are calculated in square kilometers using the Lambert Azimuthal Equal Area projection, centered on the centroid of occurrence points given as inputs.

Value

A sp_range object (S4) containing: (1) a data.frame with information about the species range, and Spatial objects of (2) unique occurrences, (3) species range, (4) extent of occurrence, and (5) area of occupancy.

If `extent_of_occurrence` and/or `area_of_occupancy` = `FALSE`, the corresponding spatial objects in the resulting sp_range object will be empty, an areas will have a value of 0.

Examples

```
# getting the data
data("occ_p", package = "rangemap")

# buffer distance
dist <- 100000

buff_range <- rangemap_buffer(occurrences = occ_p, buffer_distance = dist)

summary(buff_range)
```

`rangemap_enm` *Species distributional ranges based on ENMs/SDMs outputs*

Description

rangemap_enm generates a distributional range for a given species using a continuous raster layer produced using ecological niche modeling or species distribution modeling tools. This function binarizes the model in suitable and unsuitable areas using a user specified level of omission or a given threshold value. Optionally, representations of the species extent of occurrence (using convex hulls) and the area of occupancy according to the IUCN criteria can also be generated. Shapefiles can be saved in the working directory if it is needed.

Usage

```
rangemap_enm(model_output, occurrences = NULL, threshold_value = NULL,
             threshold_omission = NULL, min_polygon_area = 0,
             simplify = FALSE, simplify_level = 0, polygons = NULL,
             extent_of_occurrence = TRUE, area_of_occupancy = TRUE,
             final_projection = NULL, save_shp = FALSE, name,
             overwrite = FALSE, verbose = TRUE)
```

Arguments

`model_output`	a RasterLayer of suitability for the species of interest generated using a ENM or SDM algorithm, that will be binarized using the a user-defined `threshold_value` or a value calculated based on a percentage of omission (0 - 100) defined in `threshold_omission`. If the layer is projected, this projection must be WGS84 (EPSG:4326); if not projected, WGS84 projection will be assigned for the analysis.
`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees. `occurrences` may not be defined, but if so, `threshold_value` must be defined. Default = `NULL`.
`threshold_value`	(numeric) value used for reclassifying the `model_output`. This value will be the lowest considered as suitable for the species and must be inside the range of values present in `model_output`. If defined, `threshold_omission` will be ignored. If `occurrences` is not defined, this parameter is required. Default = `NULL`.
`threshold_omission`	(numeric) percentage of occurrence records to be excluded from suitable areas considering their values of suitability in the continuous model (e.g., 0, 5, or 10). Ignored if `threshold_value` is provided. Default = `NULL`.
`min_polygon_area`	(numeric) minimum area of polygons that will be kept as part of the species ranges after defining suitable areas and convert raster layer to polygon. Default = 0. A value of 0 will keep all polygons.
`simplify`	(logical) if `TRUE`, polygons of suitable areas will be simplified at a tolerance defined in `simplify_level`. Default = `FALSE`.
`simplify_level`	(numeric) tolerance to consider when simplifying polygons created from suitable areas in `model_output`. Lower values will produce polygons more similar to the original geometry. Default = 0. If simplifying is needed, try numbers 0-1 first. Ignored if `simplify` = `FALSE`.
`polygons`	(optional) a SpatialPolygons* object to clip polygons and adjust extent of occurrence to these limits. Projection must be WGS84 (EPSG:4326). If `NULL`, the default, a simplified world map will be used.
`extent_of_occurrence`	(logical) whether to obtain the extent of occurrence of the species based on a simple convex hull polygon; default = `TRUE`.

`area_of_occupancy`	(logical) whether to obtain the area of occupancy of the species based on a simple grid of 4 km^2 resolution; default = `TRUE`.
`final_projection`	(character) string of projection arguments for resulting Spatial objects. Arguments must be as in the PROJ.4 documentation. See `CRS-class` for details. If `NULL`, the default, projection used is WGS84 (EPSG:4326).
`save_shp`	(logical) if `TRUE`, shapefiles of the species range, occurrences, extent of occurrence, and area of occupancy will be written in the working directory. Default = `FALSE`.
`name`	(character) valid if `save_shp` = `TRUE`. The name of the shapefile to be exported. A suffix will be added to `name` depending on the object, as follows: species extent of occurrence = "_extent_occ", area of occupancy = "_area_occ", and occurrences = "_unique_records".
`overwrite`	(logical) whether or not to overwrite previous results with the same name. Default = `FALSE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

All resulting Spatial objects in the list of results will be projected to the `final_projection`. Areas are calculated in square kilometers using the Lambert Azimuthal Equal Area projection, centered on the centroid of occurrence points given as inputs or, if points are not provided, the resulting range.

Value

If `occurrences` and `threshold_omission` are defined, a sp_range object (S4) containing: (1) a data.frame with information about the species range, and Spatial objects of (2) unique occurrences, (3) species range, (4) extent of occurrence, and (5) area of occupancy.

If instead of `occurrences` and `threshold_omission`, `threshold_value` is provided, the result will be a sp_range object (S4) of two elements: (1) a data.frame with information about the species range, and (2) a SpatialPolygons object of the species range.

If `extent_of_occurrence` and/or `area_of_occupancy` = `FALSE`, the corresponding spatial objects in the resulting sp_range object will be empty, an areas will have a value of 0.

Examples

```
# parameters
sp_mod <- raster::raster(list.files(system.file("extdata", package = "rangemap"),
                                    pattern = "sp_model", full.names = TRUE))
data("occ_train", package = "rangemap")

thres <- 5
save <- TRUE
name <- "test"
```

```
enm_range <- rangemap_enm(model_output = sp_mod, occurrences = occ_train,
                          threshold_omission = thres)

summary(enm_range)
```

`rangemap_explore` *Exploring occurrences before creating range maps*

Description

rangemap_explore generates simple figures to visualize species occurrence data in the geography.

Usage

```
rangemap_explore(occurrences, show_countries = FALSE, graphic_device = FALSE,
                 xlim = NULL, ylim = NULL)
```

Arguments

`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84).
`show_countries`	(logical) if `TRUE`, ISO 3 country codes will label country polygons. Default = `FALSE`.
`graphic_device`	(logical) if `TRUE`, a new graphic device is opened to plot the figure. Default = `FALSE`.
`xlim`	(numeric) vector of length = 2 with the limits in longitude for the plot. The default, NULL, uses the limits of countries with `occurrences`.
`ylim`	(numeric) vector of length = 2 with the limits in latitude for the plot. The default, NULL, uses the limits of countries with `occurrences`.

Details

Base map of countries of the world is a SpatialPolygonsDataFrame downloaded from the Natural Earth database (scale = 50).

Value

A simple figure of species occurrences in a geographical context.

Examples

```
# getting the data
data("occ_f", package = "rangemap")

# simple figure of the species occurrence data
rangemap_explore(occurrences = occ_f, show_countries = TRUE)
```

`rangemap_hull` *Species distributional ranges based on convex or concave hull polygons*

Description

rangemap_hull generates a distributional range for a given species by creating convex or concave hull polygons based on occurrence data. Optionally, representations of the species extent of occurrence (using convex hulls) and the area of occupancy according to the IUCN criteria can also be generated. Shapefiles can be saved in the working directory if it is needed.

Usage

```
rangemap_hull(occurrences, hull_type = "convex", concave_distance_lim = 5000,
              buffer_distance = 50000, split = FALSE,
              cluster_method = "hierarchical", split_distance = NULL,
              n_k_means = NULL, polygons = NULL, extent_of_occurrence = TRUE,
              area_of_occupancy = TRUE, final_projection = NULL,
              save_shp = FALSE, name, overwrite = FALSE, verbose = TRUE)
```

Arguments

`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84).
`hull_type`	(character) type of hull polygons to be created. Available options are: "convex" and "concave". Default = "convex".
`concave_distance_lim`	(numeric) distance, in meters, to be passed to the length_threshold parameter of the `concaveman` function. Default = 5000. Ignored if `hull_type` is not "concave".
`buffer_distance`	(numeric) distance, in meters, to be used for creating a buffer around resulting hull polygons; default = 50000.
`split`	(logical) if `TRUE`, a distance (for hierarchical clustering) or a number (for K-means clustering) is used to separate distinct chunks of occurrences. Recommended when the species of interest has a disjunct distribution. Default = `FALSE`.
`cluster_method`	(character) name of the method to be used for clustering the occurrences. Options are "hierarchical" and "k-means"; default = "hierarchical". Note that this parameter is ignored when `split` = `FALSE`. See details for more information on the two available methods.
`split_distance`	(numeric) distance in meters that will limit connectivity among hull polygons created with chunks of points separated by long distances. This parameter is used when `cluster_method` = "hierarchical" and `split` = `TRUE`. Default = `NULL`.
`n_k_means`	(numeric) if `split` = `TRUE`, number of clusters in which the species occurrences will be grouped when using the "k-means" `cluster_method`. Default = `NULL`.

`polygons`	(optional) a SpatialPolygons* object to clip polygons and adjust the species range and other polygons to these limits. Projection must be WGS84 (EPSG:4326). If `NULL`, the default, a simplified world map will be used.
`extent_of_occurrence`	(logical) whether to obtain the extent of occurrence of the species based on a simple convex hull polygon; default = `TRUE`.
`area_of_occupancy`	(logical) whether to obtain the area of occupancy of the species based on a simple grid of 4 km^2 resolution; default = `TRUE`.
`final_projection`	(character) string of projection arguments for resulting Spatial objects. Arguments must be as in the PROJ.4 documentation. See `CRS-class` for details. If `NULL`, the default, projection used is WGS84 (EPSG:4326).
`save_shp`	(logical) if `TRUE`, shapefiles of the species range, occurrences, extent of occurrence, and area of occupancy will be written in the working directory. Default = `FALSE`.
`name`	(character) valid if `save_shp` = `TRUE`. The name of the shapefile to be exported. A suffix will be added to `name` depending on the object, as follows: species extent of occurrence = "_extent_occ", area of occupancy = "_area_occ", and occurrences = "_unique_records".
`overwrite`	(logical) whether or not to overwrite previous results with the same name. Default = `FALSE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

All resulting Spatial objects in the results will be projected to the `final_projection`. Areas are calculated in square kilometers using the Lambert Azimuthal Equal Area projection, centered on the centroid of occurrence points given as inputs.

If `split = TRUE`, some point clusters may end up having less than three points, in which cases creating hull polygons is not possible. Such point clusters will be discarded if `buffer_distance` = 0. To keep them as part of the final result, use a `buffer_distance` > 0.

The `cluster_method` must be chosen based on the spatial configuration of the species occurrences. Both methods make distinct assumptions and one of them may perform better than the other depending on the spatial pattern of the data.

The k-means method, for example, performs better when the following assumptions are fulfilled: Clusters are spatially grouped—or "spherical" and Clusters are of a similar size. Owing to the nature of the hierarchical clustering algorithm it may take more time than the k-means method. Both methods make assumptions and they may work well on some data sets, and fail on others.

Another important factor to consider is that the k-means method always starts with a random choice of cluster centers, thus it may end in different results on different runs. That may be problematic when trying to replicate your methods. With hierarchical clustering, most likely the same clusters can be obtained if the process is repeated.

For more information on these clustering methods see Aggarwal and Reddy (2014), here.

Value

A sp_range object (S4) containing: (1) a data.frame with information about the species range, and Spatial objects of (2) unique occurrences, (3) species range, (4) extent of occurrence, and (5) area of occupancy.

If `extent_of_occurrence` and/or `area_of_occupancy` = `FALSE`, the corresponding spatial objects in the resulting sp_range object will be empty, an areas will have a value of 0.

Examples

```
# getting the data
data("occ_d", package = "rangemap")

# other info for running
dist <- 100000
hull <- "convex" # try also "concave"

hull_range <- rangemap_hull(occurrences = occ_d, hull_type = hull,
                            buffer_distance = dist)

summary(hull_range)
```

`rangemap_plot` *Plot of sp_range* objects*

Description

rangemap_plot generates customizable figures of species range maps using objects produced by other functions of this package.

Usage

```
rangemap_plot(sp_range, polygons, add_EOO = FALSE, add_occurrences = FALSE,
              basemap_color = "gray93", range_color = "darkgreen",
              extent_color = "blue", occurrences_color = "yellow",
              grid = FALSE, grid_sides = "bottomleft", ylabels_position = 1.3,
              legend = FALSE, legend_position = "bottomright",
              northarrow = FALSE, northarrow_position = "topright",
              scalebar = FALSE, scalebar_position = "bottomleft",
              scalebar_length = 100, zoom = 1)
```

Arguments

`sp_range`	a sp_range object produced with any of the following functions: `rangemap_buffer`, `rangemap_boundaries`, `rangemap_hull`, `rangemap_enm`, and `rangemap_tsa`.
`polygons`	(optional) a SpatialPolygons* object to be used as base for the map. If `NULL`, a simplified world map will be used.

`add_EOO`	(logical) if `TRUE`, the extent of occurrence of the species will be added to the figure. Ignored if the `sp_range` is product of the `rangemap_boundaries` function and administrative areas were selected only based on names. Default = `FALSE`.
`add_occurrences`	(logical) if `TRUE`, the species occurrence records will be added to the figure. Ignored if the `sp_range` is product of the `rangemap_boundaries` function and administrative areas were selected only based on names. Default = `FALSE`.
`basemap_color`	color for the basemap (`polygons`) to be plotted. Default = "gray93".
`range_color`	color for the species `sp_range` to be plotted. Default = "darkgreen".
`extent_color`	color for the species extent of occurrence to be plotted. Default = "blue".
`occurrences_color`	color for the species `occurrences` to be plotted. Default = "yellow".
`grid`	(logical) if `TRUE`, labels and grid division ticks will be inserted in `grid_sides`. Default = `FALSE`.
`grid_sides`	(character) sides in which the labels will be placed in the figure. Options are the same than for other position character indicators (see details). Default = "bottomleft".
`ylabels_position`	(numeric) if `grid` = `TRUE`, separation (in lines) of y axis labels from the axis. Bigger numbers will increase separation. Default = 1.3.
`legend`	(logical) if `TRUE`, a legend of the plotted features will be added to the figure at `legend_position`. Default = `FALSE`.
`legend_position`	(numeric or character) site in the figure where the legend will be placed. If numeric, vector of length two indicating x and y coordinates to be used to position the legend. See details for options of character indicators of position. Default = "bottomright".
`northarrow`	(logical) if `TRUE`, a simple north arrow will be placed in `northarrow_position`. Default = `FALSE`.
`northarrow_position`	(numeric or character) site in the figure where the north legend will be placed. If numeric, vector of length two indicating x and y coordinates to be used to position the north arrow. See details for options of character indicators of position. Default = "topright".
`scalebar`	(logical) if `TRUE`, a simple scale bar will be inserted in the figure at `scalebar_position` with a length of `scalebar_length`. Default = `FALSE`.
`scalebar_position`	(numeric or character) site in the figure where the scale bar will be placed. If numeric, vector of length two indicating x and y coordinates to be used to position the scale bar. See details for options of character indicators of position. Default = "bottomleft".
`scalebar_length`	(numeric) length of the scale bar in km. Using entire numbers divisible for two is recommended. Default = 100.
`zoom`	(numeric) zoom factor when plotting the species range in a map. Default = 1. Larger values will zoom in into the species range and smaller values will zoom out. A value of 0.5 will duplicate the area that the biggest range is covering.

Details

Position of distinct elements depend on the spatial configuration of the species range. Therefore, their position may need to be changed if the elements are needed. Position options are: "bottomright", "bottomleft", "topleft", and "topright". Numerical descriptions of positions are also allowed.

Value

A plot of the species range in a geographic context, with some map components defined by the user.

Examples

```
# example data
data("cvehull_range", package = "rangemap")

# arguments for the species range figure
extent <- TRUE
occ <- TRUE
legend <- TRUE

# creating the species range figure
rangemap_plot(cvehull_range, add_EOO = extent, add_occurrences = occ,
              legend = legend)
```

`rangemap_tsa` *Species distributional ranges based on trend surface analyses*

Description

rangemap_tsa generates a distributional range for a given species using a trend surface analysis. An approach to the species extent of occurrence (using convex hulls) and the area of occupancy according to the IUCN criteria is also generated. Shapefiles can be saved in the working directory if it is needed.

Usage

```
rangemap_tsa(occurrences, region_of_interest, cell_size = 5,
             threshold = 0, simplify = FALSE, simplify_level = 0,
             extent_of_occurrence = TRUE, area_of_occupancy = TRUE,
             final_projection = NULL, save_shp = FALSE,
             save_ts_layer = FALSE, name, overwrite = FALSE, verbose = TRUE)
```

Arguments

`occurrences`	a data.frame containing geographic coordinates of species occurrences, columns must be: Species, Longitude, and Latitude. Geographic coordinates must be in decimal degrees (WGS84).

`region_of_interest`	a SpatialPolygonsDataFrame object on which the trend surface analysis will be performed. For instance, a country, an ecoregion, or a biogeographic region. Projection must be WGS84 (EPSG:4326).
`cell_size`	(numeric) vector of length 1 or 2, defining the size of cells (km) at which the resultant trend surface will be created; default = 5. `cell_size` will depend on the extent of `region_of_interest`. Values lower than 1 are only recommended when the species is locally distributed.
`threshold`	(numeric) percentage of occurrence records to be excluded when deciding the minimum value trend surface output to be considered as part of the species range. Default = 0.
`simplify`	(logical) if `TRUE`, polygons of suitable areas will be simplified at a tolerance defined in `simplify_level`. Default = `FALSE`.
`simplify_level`	(numeric) tolerance at the moment of simplifying polygons created using the trend surface model. Lower values will produce polygons more similar to the original geometry. Default = 0. If simplifying is needed, try numbers between 0 and 1 first.
`extent_of_occurrence`	(logical) whether to obtain the extent of occurrence of the species based on a simple convex hull polygon; default = `TRUE`.
`area_of_occupancy`	(logical) whether to obtain the area of occupancy of the species based on a simple grid of 4 km^2 resolution; default = `TRUE`.
`final_projection`	(character) string of projection arguments for resulting Spatial objects. Arguments must be as in the PROJ.4 documentation. See `CRS-class` for details. If `NULL`, the default, projection used is WGS84 (EPSG:4326).
`save_shp`	(logical) if `TRUE`, shapefiles of the species range, occurrences, extent of occurrence, and area of occupancy will be written in the working directory. Default = `FALSE`.
`save_ts_layer`	(logical) if `TRUE`, the TSA layer will be included as part of the object returned. If `save_shp` = TRUE, the TSA layer will be written in GeoTiff format. Default = `FALSE`
`name`	(character) valid if `save_shp` = TRUE. The name of the geographic files to be exported. A suffix will be added to `name` depending on the object as follows: species extent of occurrence = "_extent_occ", area of occupancy = "_area_occ", occurrences = "_unique_records", and, if `save_ts_layer` = `TRUE`, trend surface layer "_tsa".
`overwrite`	(logical) whether or not to overwrite previous results with the same name. Default = `FALSE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

All resulting Spatial objects in the results will be projected to the `final_projection`. Areas are calculated in square kilometers using the Lambert Azimuthal Equal Area projection, centered on the centroid of occurrence points given as inputs.

Trend surface analysis is a method based on low-order polynomials of spatial coordinates for estimating a regular grid of points from scattered observations. This method assumes that all cells not occupied by occurrences are absences; hence its use depends on the quality of data and the certainty of having or not a complete sampling of the `regiong_of_interest`.

Value

A sp_range object (S4) containing: (1) a data.frame with information about the species range, and SpatialPolygons objects of (2) unique occurrences, (3) species range, (4) extent of occurrence, and (5) area of occupancy. If `save_ts_layer` = TRUE, a (6) TSA layer will be included as well.

If `extent_of_occurrence` and/or `area_of_occupancy` = `FALSE`, the corresponding spatial objects in the resulting sp_range object will be empty, an areas will have a value of 0.

Examples

```
# data
data("occ_f", package = "rangemap")

CU <- simple_wmap("simple", regions = "Cuba")

# running
tsa_range <- rangemap_tsa(occurrences = occ_f, region_of_interest = CU,
                          cell_size = 5)

summary(tsa_range)
```

`ranges_emaps` *Plots of species ranges on maps of environmental variables*

Description

ranges_emaps plots one or more ranges of the same species on various maps of environmental factors (e.g. climatic variables) to detect implications of using one or other type of range regarding the environmental conditions in the areas.

Usage

```
ranges_emaps(..., variables, add_occurrences = FALSE,
             range_colors = NULL, color_variables = NULL,
             ranges_legend = TRUE, legend_position = "bottomright",
             legend_cex = 0.7, zoom = 0.7, verbose = TRUE)
```

Arguments

`...`	one or more objects of class `sp_range` produced with any of the following functions: `rangemap_buffer`, `rangemap_boundaries`, `rangemap_hull`, `rangemap_enm`, and/or `rangemap_tsa`. Using up to three or four ranges is recommended for more precise comparisons.

`variables`	a RasterLayer or RasterStack object of environmental variables that will be used as the base for maps. Projection is assumed to be WGS84 (EPSG:4326). Consider that depending on the species range, using more than 9 variables creates a plot that may not fit in an A4 paper sheet. A maximum of 21 variables is allowed, if this limit is surpassed, other variables will be ignored.
`add_occurrences`	(logical) if `TRUE`, species occurrences contained in `sp_range` objects will be added to the figure. Default = `FALSE`. If the none of the objects contains occurrences, this argument will be ignored.
`range_colors`	vector of colors for borders of species ranges. If `NULL`, the default, distinct levels of gray will be used. If more than 3 `sp_range` objects are included, defining your own colors is recommended.
`color_variables`	a color palette (a vector of continuous colors generated by functions like heat.colors). If `NULL`, the default, rev(terrain.colors(255)) will be used.
`ranges_legend`	(logical) if `TRUE`, a legend of the plotted ranges will be added to the last panel of the plot at `legend_position`. Default = `TRUE`.
`legend_position`	(numeric or character) location where the legend will be placed in the plot. If numeric, vector of length = 2 indicating x and y coordinates to position the legend. See details in `legend` for character options of position. Default = "bottomright".
`legend_cex`	(numeric) size of the legend with respect to `cex` option in `par`. Default = 0.7.
`zoom`	(numeric) zoom factor when plotting the species range in a map (based on the largest range). Default = 1.3. Larger values will zoom in into the species range and smaller values will zoom out. A value of 0.5 will duplicate the area that the biggest range is covering.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Details

Position of distinct elements depend on the spatial configuration of the species range. Therefore, their position may need to be changed if such elements are needed (e.g., legend). Current character options available for position are: "bottomright", "bottomleft", "topleft", and "topright".

Value

A plot showing species ranges on top of maps of environmental variables.

Examples

```
# example data
data("buffer_range", package = "rangemap")
data("cxhull_range", package = "rangemap")
data("cvehull_range", package = "rangemap")

vars <- raster::stack(system.file("extdata", "variables.tif",
                                  package = "rangemap"))
names(vars) <- c("bio5", "bio6", "bio13", "bio14")
```

```
# plotting
ranges_emaps(buffer_range, cxhull_range, cvehull_range, variables = vars)
```

ranges_espace *Comparison of species ranges in environmental space*

Description

ranges_espace generates a three dimensional comparison of distributional ranges for a species created using distinct functions of `rangemap`, to visualize them in environmental conditions.

Usage

```
ranges_espace(..., add_occurrences = TRUE, variables, do_pca = FALSE,
              max_background = 10000, occurrence_color = "blue",
              range_colors = NULL, alpha = 0.6, legend = TRUE,
              verbose = TRUE)
```

Arguments

`...`	one or more objects of class `sp_range` produced with any of the following functions: `rangemap_buffer`, `rangemap_boundaries`, `rangemap_hull`, `rangemap_enm`, and/or `rangemap_tsa`. Using up to five ranges is allowed for more precise comparisons.
`add_occurrences`	(logical) if `TRUE`, species occurrences contained in one of the `sp_range` objects will be plotted. Default = `TRUE`. If none of the objects contains occurrences this argument will be ignored.
`variables`	a RasterStack object of at least 3 environmental variables that will be used to represent the environmental space. If more than 3 variables are provided, the first ones will be used, unless `do_pca` = TRUE, in which case the 3 first principal components will be used. Projection is assumed to be WGS84 (EPSG:4326).
`do_pca`	(logical) whether to summarize all variables using a principal component analysis. Default = FALSE.
`max_background`	(numeric) maximum number of data from variables to be used for representing the environmental space. Default = 10000.
`occurrence_color`	color for occurrence records in environmental space.
`range_colors`	vector of colors for the ranges to be represented. If `NULL`, the default, a set of colors will be used. Since transparency is used for representing ranges in the plot, colors may look different.
`alpha`	(numeric) degree of opacity for plotting species ranges. Default = 0.6.
`legend`	(logical) if `TRUE`, a simple legend will be added. Default = `TRUE`.
`verbose`	(logical) whether or not to print messages about the process. Default = TRUE.

Value

A figure showing distributional ranges of a species represented in environmental space (3 dimensions).

Examples

```
# example data
data("buffer_range", package = "rangemap")
data("cxhull_range", package = "rangemap")

vars <- raster::stack(system.file("extdata", "variables.tif",
                                  package = "rangemap"))
names(vars) <- c("bio5", "bio6", "bio13", "bio14")

## comparison
ranges_espace(buffer_range, cxhull_range, variables = vars,
              add_occurrences = TRUE)
```

---

simple_wmap *Get a simplified SpatialPolygonsDataFrame of the world*

Description

Get a simplified SpatialPolygonsDataFrame of the world

Usage

```
simple_wmap(which = "simplest", regions = ".")
```

Arguments

which	(character) name of type of SpatialPolygons to be obtained. Options are: "simple" and "simplest"; default = "simplest".
regions	(character) name of the country (or region if which = "simple") for which to obtain the SpatialPolygonsDataFrame.

Value

A simplified SpatialPolygonsDataFrame of the world in WGS84 projection.

Examples

```
map <- simple_wmap()
```

`spdf_range` *Example SpatialPolygonsDataFrame of a species range*

Description

A SpatialPolygonsDataFrame representing the distribution of a species from North America.

Usage

```
spdf_range
```

Format

SpatialPolygonsDataFrame with 1 feature.

features SpatialPolygons, 1.

data.frame ID = 1.

Examples

```
data("spdf_range", package = "rangemap")
spdf_range
```

`sp_model` *An ecological niche model created with Maxent*

Description

A RasterLayer containing an ecological niche model for the a tick (*Amblyomma americanum*).

Format

A RasterLayer with 150 rows, 249 columns, and 37350 cells:

Suitability suitability, in probability values.

Value

No return value, used with function `raster` to bring an example of ecological niche modeling output.

Source

`https://kuscholarworks.ku.edu/handle/1808/26376`

Examples

```
model <- raster::raster(system.file("extdata", "sp_model.tif",
                                    package = "rangemap"))

raster::plot(model)
```

---

sp_range — *An S4 class to organize data and results of sp_range* objects*

Description

A list of classes (some of them inherited) to contain information derived from analyses using `rangemap`. Available classes are: sp_range, sp_range_iucn, and sp_range_iucnextra.

Slots

`name` name depending on how species range was constructed.

`summary` a data.frame.

`species_unique_records` a SpatialPointsDataFrame, not in sp_range.

`species_range` a SpatialPolygonsDataFrame.

`extent_of_occurrence` a SpatialPolygonsDataFrame, not in sp_range.

`area_of_occupancy` a SpatialPolygonsDataFrame, not in sp_range.

`trend_surface_model` a RasterLayer, not in sp_range.

Examples

```
showClass("sp_range")
showClass("sp_range_iucn")
```

---

summary — *Summary of attributes and results*

Description

Summary of attributes and results

Usage

```
## S4 method for signature 'sp_range'
summary(object)

## S4 method for signature 'sp_range_iucn'
summary(object)

## S4 method for signature 'sp_range_iucnextra'
summary(object)
```

Arguments

`object`	object of class sp_range*.

Value

A written summary.

`variables` *A set of environmental variables for examples*

Description

A RasterStack containing four bioclimatic variables downloaded from the WorldClim database 1.4.

Format

A RasterStack with 180 rows, 218 columns, 39240 cells, and 4 layers:

variables.1 bio5.

variables.2 bio6.

variables.3 bio13.

variables.4 bio14.

Value

No return value, used with function `stack` to bring an example of a set of environmental variables.

Source

`https://www.worldclim.org/data/v1.4/worldclim14.html`

Examples

```
vars <- raster::stack(system.file("extdata", "variables.tif",
                                  package = "rangemap"))
names(vars) <- c("bio5", "bio6", "bio13", "bio14")

raster::plot(vars)
```

Index

∗ **datasets**

adm_area_names, 3
adm_boundaries, 4
buffer_range, 5
country_codes, 7
cvehull_range, 8
cxhull_range, 8
occ_d, 15
occ_f, 16
occ_p, 16
occ_train, 17
spdf_range, 37

adm_area_names, 3, *19*, *21*
adm_boundaries, 4
AED_projection *(*LAEA_projection*)*, 13
aoo, 4

buffer_range, 5

clusters, 6
concaveman, *12*, *27*
country_codes, 7, *21*
cvehull_range, 8
cxhull_range, 8

eoo, 9

GADM_spoly, 10
geobuffer_points, 11
getData, *20, 21*

hull_polygon, 12

keep_big_polygons, 13

LAEA_projection, 13
legend, *34*

north_arrow, 14

occ_d, 15
occ_f, 16
occ_p, 16
occ_train, 17

par, *34*
plot_ranges, 18

rangemap, 19, *35*, *38*
rangemap_boundaries, *19*, 19, *29, 30*, *33*, *35*
rangemap_buffer, *5*, *19*, 22, *29*, *33*, *35*
rangemap_enm, *19*, 23, *29*, *33*, *35*
rangemap_explore, *19, 20*, 26
rangemap_hull, *8*, *19*, 27, *29*, *33*, *35*
rangemap_plot, *19*, 29
rangemap_tsa, *19*, *29*, 31, *33*, *35*
ranges_emaps, *19*, 33
ranges_espace, *19*, 35
raster, *37*

simple_wmap, 36
sp_model, 37
sp_range, *33–35*, 38
sp_range-class *(*sp_range*)*, 38
sp_range_iucn *(*sp_range*)*, 38
sp_range_iucn-class *(*sp_range*)*, 38
sp_range_iucnextra *(*sp_range*)*, 38
sp_range_iucnextra-class *(*sp_range*)*, 38
spdf_range, 37
stack, *39*
summary, 38
summary,sp_range-method *(*summary*)*, 38
summary,sp_range_iucn-method *(*summary*)*, 38
summary,sp_range_iucnextra-method *(*summary*)*, 38

variables, 39