Package: eSDM (via r-universe)

December 7, 2024

Title Ensemble Tool for Predictions from Species Distribution Models

Description A tool which allows users to create and evaluate ensembles of species distribution model (SDM) predictions. Functionality is offered through R functions or a GUI (R Shiny app). This tool can assist users in identifying spatial uncertainties and making informed conservation and management decisions. The package is further described in Woodman et al (2019) <doi:10.1111/2041-210X.13283>.

Version 0.4.4

URL https://github.com/swfsc/eSDM/, https://swfsc.github.io/eSDM/

BugReports https://github.com/swfsc/eSDM/issues/

Depends R (>= 4.0.0)

Imports dplyr (>= 1.1), magrittr, methods, purrr, rlang, ROCR, sf (>= 1.0), shiny, stats, units

Suggests colorRamps, colourpicker, dichromat, DT, knitr, leafem, leaflet, maps, raster, RColorBrewer, rmarkdown, shinybusy, shinydashboard, shinyjs, testthat (>= 2.1.0), tmap (>= 2.3), viridis, zip

License Apache License (== 2)

Encoding UTF-8

LazyData true

RoxygenNote 7.3.2

VignetteBuilder knitr

NeedsCompilation no

Author Sam Woodman [aut, cre] (<https://orcid.org/0000-0001-6071-8186>)

Maintainer Sam Woodman <sam.woodman@noaa.gov>

Repository CRAN

Date/Publication 2024-10-07 22:20:02 UTC

Config/pak/sysreqs libgdal-dev gdal-bin libgeos-dev make libssl-dev libproj-dev libsqlite3-dev libudunits2-dev zlib1g-dev

Contents

`eSDM-package` *Ensemble tool for predictions from Species Distribution Models*

Description

eSDM: A tool for creating and exploring ensembles of predictions from Species Distribution Models

Details

eSDM provides functionality for overlaying SDM predictions onto a single base geometry and creating and evaluating ensemble predictions. This can be done manually in R, or using the eSDM GUI (an R Shiny app) opened through eSDM_GUI

eSDM allows users to overlay SDM predictions onto a single base geometry, create ensembles of these predictions via weighted or unweighted averages, calculate performance metrics for each set of predictions and for resulting ensembles, and visually compare ensemble predictions with original predictions. The information provided by this tool can assist users in understanding spatial uncertainties and making informed conservation decisions.

The GUI ensures that the tool is accessible to non-R users, while also providing a user-friendly environment for functionality such as loading other polygons to use and visualizing predictions. However, user choices are restricted to the workflow provided by the GUI.

Author(s)

Sam Woodman <sam.woodman@noaa.gov>

See Also

https://swfsc.github.io/eSDM/

`ensemble_create` *Create ensemble of SDM predictions*

Description

Create a weighted or unweighted ensemble of SDM predictions, including associated uncertainty values

Usage

```
ensemble_create(x, x.idx, w = NULL, x.var.idx = NULL, ...)

## S3 method for class 'sf'
ensemble_create(x, x.idx, w = NULL, x.var.idx = NULL, ...)

## S3 method for class 'data.frame'
ensemble_create(x, x.idx, w = NULL, x.var.idx = NULL, ...)
```

Arguments

`x`	object of class `sf` or class `data.frame`
`x.idx`	vector of column names or numerical indices; indicates which columns in `x` will be used to create the ensemble
`w`	weights for the ensemble; either a numeric vector the same length as `x` or a data frame (or tibble) with the same number of rows as `x` and `ncol(w) == length(x.idx)`. If w is a numeric vector, its values (i.e. the weights) must sum to 1. The default value is `1 / length(x.idx)`, i.e. an unweighted ensemble
`x.var.idx`	vector of column names or column indices; indicates columns in `x` with variance values with which to calculate uncertainty values for the ensemble. If `x.var.idx` is specified, it must be the same length as `x.idx`. Use `x.var.idx = NULL` (the default) if none of the predictions have associated uncertainty values; in this case the uncertainty values for the ensemble will be calculated using the among-model uncertainty. See the 'Details' section for more information
`...`	Arguments to be passed to methods; specifically designed for passing `na.rm` argument to `sum`

Details

`ensemble_create` is designed to be used after overlaying predictions with `overlay_sdm` and (if desired) rescaling the overlaid predictions with `ensemble_rescale`.

This function implements ensemble methods provided in eSDM_GUI. Note that it does not implement regional exclusion, which must be done manually if not using the GUI.

Ensemble uncertainty is calculated using either the within-model uncertainty (if `x.var.idx` is specified) or the among-model uncertainty (if `x.var.idx` is `NULL`). See the eSDM GUI manual for applicable formulas.

Value

An object of the same class as x with two columns appended to the data frame:

- 'Pred_ens' - The ensemble predictions
- 'Var_ens' - The variance of the ensemble predictions, calculated using either the within-model uncertainty (if x.var.idx is specified) or the among-model uncertainty (if x.var.idx is NULL)

Note that all other columns of x will be included in the returned object. Also, if x is of class sf then 1) the geometry list-column will be the last column of the returned object and 2) the agr attribute will be set as 'constant' for 'Pred_ens' and 'Var_ens'

Examples

```
ensemble_create(preds.1, c("Density", "Density2"), c(0.2, 0.8))
ensemble_create(preds.1, 1:2, c(0.2, 0.8), c("Var1", "Var2"))
ensemble_create(data.frame(a = 1:5, b = 3:7), c(1, 2))

weights.df <- data.frame(runif(325), c(rep(NA, 100), runif(225)))
ensemble_create(preds.1, c("Density", "Density2"), weights.df, na.rm = TRUE)
```

---

ensemble_rescale *Rescale SDM predictions*

Description

Rescale SDM predictions and (if applicable) associated uncertainties

Usage

```
ensemble_rescale(x, x.idx, y, y.abund = NULL, x.var.idx = NULL)
```

Arguments

x	object of class sf
x.idx	vector of column names or column indices; indicates columns in x with prediction values that will be rescaled
y	rescaling method; must be either "abundance" or "sumto1". See 'Details' section for descriptions of the rescaling methods
y.abund	numeric value; ignored if y is not "abundance"
x.var.idx	vector of column names or column indices; indicates columns in x with variance values that will be rescaled. If x.var.idx is specified, it must be the same length as x.idx. Use x.var.idx = NULL (the default) if none of the predictions have associated uncertainty values; see the 'Details' section for more information

Details

ensemble_rescale is intended to be used after overlaying predictions with overlay_sdm and before creating ensembles with ensemble_create. The provided rescaling methods are:

- 'abundance' - Rescale the density values so that the predicted abundance is y.abund
- 'sumto1' - Rescale the density values so their sum is 1

SDM uncertainty values must be rescaled differently than the prediction values. Columns specified in x.var.idx must contain variance values. These values will be rescaled using the formula var(c * x) = c^2 * var(x), where c is the rescaling factor for the associated predictions.

If x.var.idx is not NULL, then the function assumes x.var.idx[1] contains the variance values associated with the predictions in x.idx[1], x.var.idx[2] contains the variance values associated with the predictions in x.idx[2], etc. Use NA in x.var.idx to indicate a set of predictions that does not have associated uncertainty values (e.g., x.var.idx = c(4, NA, 5))

Value

The sf object x with the columns specified by x.idx and x.var.idx rescaled. The agr attributes of x will be conserved

Examples

```
ensemble_rescale(preds.1, c("Density", "Density2"), "abundance", 50)
ensemble_rescale(preds.1, c(1, 2), "sumto1")

ensemble_rescale(
  preds.1, c("Density", "Density2"), "abundance", 100, c(3,4)
)
```

---

eSDM_GUI *Open the eSDM GUI*

Description

Open the eSDM graphical user interface (GUI); an R Shiny app for creating ensemble predictions using SDM predictions.

Usage

```
eSDM_GUI(launch.browser = TRUE)
```

Arguments

launch.browser Logical with default of TRUE; passed to launch.browser argument of runApp

`evaluation_metrics` *Calculate SDM evaluation metrics*

Description

Calculate AUC, TSS, and RMSE for given density predictions and validation data

Usage

```
evaluation_metrics(x, x.idx, y, y.idx, count.flag = FALSE)
```

Arguments

`x`	object of class sf; SDM predictions
`x.idx`	name or index of column in `x` with prediction values
`y`	object of class sf; validation data
`y.idx`	name or index of column in `y` with validation data. This validation data column must have at least two unique values, e.g. 0 and 1
`count.flag`	logical; `TRUE` indicates that the data in column `y.idx` is count data, while `FALSE` indicates that the data is presence/absence. See details for differences in data processing based on this flag.

Details

If `count.flag == TRUE`, then `eSDM::model_abundance(x, x.idx, FALSE)` will be run to calculate predicted abundance and thus calculate RMSE. Note that this assumes the data in column `x.idx` of `x` are density values.

If `count.flag == FALSE`, then all of the values in column `y.idx` of `y` must be `0` or `1`.

All rows of `x` with a value of `NA` in column `x.idx` and all rows of `y` with a value of `NA` in column `y.idx` are removed before calculating metrics

Value

A numeric vector with AUC, TSS and RMSE values, respectively. If `count.flag == FALSE`, the RMSE value will be `NA`

Examples

```
evaluation_metrics(preds.1, 2, validation.data, "sight")

evaluation_metrics(preds.1, "Density2", validation.data, "count", TRUE)
```

gshhg.l.L16 *Low resolution GSHHG world map*

Description

Low resolution GSHHG world map, includes hierarchical levels L1 and L6. Processed using st_make_valid

Usage

```
gshhg.l.L16
```

Format

An object of class sfc

Source

http://www.soest.hawaii.edu/pwessel/gshhg/

model_abundance *Calculate predicted abundance*

Description

Calculates the predicted abundance by multiplying the density prediction values by prediction polygon areas

Usage

```
model_abundance(x, dens.idx, sum.abund = TRUE)
```

Arguments

x	object of class sf; SDM with density predictions. Must have a valid crs code
dens.idx	name or index of column(s) in x with density predictions. Can be a character vector (column names) or numeric vector (column indices)
sum.abund	logical; whether or not to sum all of the predicted abundances

Details

Multiplies the values in the specified column(s) (i.e. the density predictions) by the area in square kilometers of their corresponding prediction polygon. The area of each prediction polygon is calculated using st_area from geos_measures. x must have a valid crs code to calculate area for these abundance calculations.

Value

If sum.abund == TRUE, then a vector of the same length as dens.idx representing the predicted abundance for the density values in each column.

If sum.abund == FALSE and the length of dens.idx is 1, then a numeric vector with the predicted abundance of each prediction polygon of x.

If sum.abund == FALSE and the length of dens.idx is greater than 1, then a data frame with length(dens.idx) columns of the predicted abundance of prediction polygons

Examples

```
model_abundance(preds.1, "Density")
model_abundance(preds.1, c(1, 1))
model_abundance(preds.1, c(1, 1), FALSE)
```

---

overlay_sdm *Overlay SDM predictions onto base geometry*

Description

Overlay specified SDM predictions that meet the percent overlap threshold requirement onto base geometry

Usage

```
overlay_sdm(base.geom, sdm, sdm.idx, overlap.perc)
```

Arguments

base.geom	object of class sfc; base geometry
sdm	object of class sf; original SDM predictions
sdm.idx	names or indices of column(s) with data to be overlaid
overlap.perc	numeric; percent overlap threshold, i.e. percentage of each base geometry polygon must overlap with SDM prediction polygons for overlaid density value to be calculated and not set as NA

Details

See the eSDM GUI manual for specifics about the overlay process. This process is equivalent to areal interpolation (Goodchild and Lam 1980), where base.geom is the target, sdm is the source, and the data specified by sdm.idx are spatially intensive.

Note that overlay_sdm removes rows in sdm that have NA values in the first column specified in sdm.idx (i.e. sdm.idx[1]), before the overlay. Thus, for valid overlay results, all columns of sdm specified in sdm.idx must either have NA values in the same rows or contain only NAs.

Value

Object of class `sf` with the geometry of `base.geom` and the data in the `sdm.idx` columns of `sdm` overlaid onto that geometry. Note that this means all columns of `sdm` not in `sdm.idx` will not be in the returned object. Because the data are considered spatially intensive, the `agr` attribute will be set as 'constant' for all columns in the returned object.

Additionally, the output will match the class of `sdm`, with regards to the classes tbl_df, tbl, and data.frame. This means that, in addition to being an `sf` object, if `sdm` is a tibble then the output will also be a tibble, while if `sdm` is just a data frame then the output will not be a tibble.

References

Goodchild, M.F. & Lam, N.S.-N. (1980) Areal interpolation: a variant of the traditional spatial problem. Geo-Processing, 1, 297-312.

Examples

```
pol1.geom <- sf::st_sfc(
  sf::st_polygon(list(rbind(c(1,1), c(3,1), c(3,3), c(1,3), c(1,1)))),
  crs = sf::st_crs(4326)
)
pol2.geom <- sf::st_sfc(
  sf::st_polygon(list(rbind(c(0,0), c(2,0), c(2,2), c(0,2), c(0,0)))),
  crs = sf::st_crs(4326)
)
pol2.sf <- sf::st_sf(data.frame(Dens = 0.5), geometry = pol2.geom,
                     crs = sf::st_crs(4326))

overlay_sdm(pol1.geom, pol2.sf, 1, 25)

# Output 'Dens' value is NA because of higher overlap.perc value
overlay_sdm(pol1.geom, pol2.sf, 1, 50)


# These examples take longer to run
overlay_sdm(sf::st_geometry(preds.1), preds.2, 1, 50)
overlay_sdm(sf::st_geometry(preds.2), preds.1, "Density", 50)
```

---

preds *Sample SDM density predictions*

Description

`preds.1`, `preds.2`, and `preds.3` are objects of class `sf` that serve as sample sets of SDM density predictions for the eSDM package

Usage

```
preds.1

preds.2

preds.3
```

Format

Objects of class `sf` with a column of density predictions (name: `Density`) and a simple feature list column (name: `geometry`). `preds.1` also has a second column of sample density predictions (name: `Density2`), as well as `Var1` and `Var2`, representing the variance

`preds1`: An object of class sf (inherits from data.frame) with 325 rows and 5 columns.

`preds2`: An object of class sf (inherits from data.frame) with 1891 rows and 2 columns.

`preds3`: An object of class sf (inherits from data.frame) with 1445 rows and 2 columns.

An object of class `sf` (inherits from `data.frame`) with 1891 rows and 2 columns.

An object of class `sf` (inherits from `data.frame`) with 1445 rows and 2 columns.

Details

`preds.1` sample SDM density predictions created by importing Sample_predictions_2.csv into the eSDM GUI, exporting predictions, and then clipping them to the SoCal_bite.csv region. Also manually added two variance columns (numbers are randomly generated with a max of 0.01)

`preds.2` sample SDM density predictions created by importing Sample_predictions_1.csv into the eSDM GUI, exporting predictions, and then clipping them to the SoCal_bite.csv region

`preds.3` is a set of sample SDM density predictions created by importing Sample_predictions_4_gdb into the eSDM GUI, exporting predictions, and then clipping them to the SoCal_bite.csv region

---

`pts2poly_centroids` *Create polygons from centroid coordinates*

Description

Create polygon(s) from a data frame with coordinates of the polygon centroid(s)

Usage

```
pts2poly_centroids(x, y, ...)
```

Arguments

x	data frame with at least two columns; the first two columns must contain longitude and latitude coordinates, respectively. See 'Details' section for how additional columns are handled
y	numeric; the perpendicular distance from the polygon centroid (center) to its edge (i.e. half the length of one side of a polygon)
...	passed to st_sf or to st_sfc, e.g. for passing named arguments `crs` and `agr`

Details

This function was designed for someone who reads in a .csv file with a grid of coordinates representing SDM prediction points and needs to create prediction polygons with the .csv file coordinates as the polygon centroids. However, the function can be used to create square polygons of any size around the provided points, regardless of if those polygons touch or overlap. The created polygons are oriented so that, in a 2D plane, their edges are parallel to either the x or the y axis.

If `x` contains more than two column, then additional columns will be treated as simple feature attributes, i.e. passed along as the first argument to st_sf

If a `crs` is not specified in `...`, then the `crs` attribute of the polygon(s) will be `NULL`.

Value

Object of class `sfc` (if `x` has exactly two columns) or class `sf` (if `x` has exactly more than two columns). The object will have a geometry type of `POLYGON`. If the object is of class `sf`, the name of the geometry list-column will be "geometry"

Examples

```
# Create an sfc object from a data frame of two columns
x <- data.frame(
  lon = c(5, 10, 15, 20, 5, 10, 15, 20),
  lat = c(5, 5, 5, 5, 10, 10, 10, 10)
)
pts2poly_centroids(x, 2.5, crs = 4326)

# Create an sf object from a data frame of more than two columns
x <- data.frame(
  lon = c(5, 10, 15, 20, 5, 10, 15, 20),
  lat = c(5, 5, 5, 5, 10, 10, 10, 10),
  sdm.pred = runif(8),
  sdm.pred2 = runif(8)
)
pts2poly_centroids(x, 2.5, crs = 4326, agr = "constant")
```

`pts2poly_vertices` *Create polygons from vertex coordinates*

Description

Create polygon(s) from a data frame with the coordinates of the polygon vertices

Usage

```
pts2poly_vertices(x, ...)
```

Arguments

`x`	data frame with at least two columns; the first two columns must contain longitude and latitude coordinates, respectively. See 'Details' section for how additional columns are handled
`...`	passed to st_sfc, e.g. for passing named argument `crs`

Details

Vertices of different polygons must be demarcated by rows with values of `NA` in both the first and second columns (i.e. the longitude and latitude columns).

All columns in `x` besides the first two columns are ignored.

If a `crs` is not specified in `...`, then the `crs` attribute of the polygon(s) will be `NULL`.

Value

Object of class `sfc` with the geometry type `POLYGON`

Examples

```
x <- data.frame(
  lon = c(40, 40, 50, 50, 40),
  lat = c(0, 10, 10, 0, 0)
)
pts2poly_vertices(x, crs = 4326)

# Create an sf object
x <- data.frame(
  lon = c(40, 40, 50, 50, 40, NA, 20, 20, 30, 30, 20),
  lat = c(0, 10, 10, 0, 0, NA, 0, 10, 10, 0, 0)
)
sf::st_sf(Pred = 1:2, geometry = pts2poly_vertices(x, crs = 4326))
```

`validation.data` *Sample validation data*

Description

Sample validation data created by cropping Validation_data.csv to the SoCal_bite.csv region (.csv files from ...)

Usage

```
validation.data
```

Format

An object of class `sf` with 8 rows and 3 variables

sight 1's and 0's indicating species presence/absence

count number of individuals observed at each point

geometry simple feature list column representing validation data points

Index