Package: caretSDM (via r-universe)

June 8, 2026

Type Package

Title Build Species Distribution Modeling using 'caret'

Version 1.9.5

Maintainer Luíz Fernando Esser <luizesser@gmail.com>

Description Use machine learning algorithms and advanced geographic information system tools to build Species Distribution Modeling in a extensible and modern fashion.

License MIT + file LICENSE

BugReports https://github.com/luizesser/caretSDM/issues

Imports caret, checkCLI, checkmate, cli, CoordinateCleaner, data.table, dismo, dplyr, ECDFniche, ecospat, fs, ggplot2, ggspatial, glue, gtools, lwgeom, maxnet, methods, pROC, purrr, raster, sf, stars, stats, stringdist, stringr, terra, tidyr, utils

Encoding UTF-8

LazyData true

Suggests bench, biomod2, blockCV, gbm, cito, covr, e1071, earth, furrr, future, gam, here, httr2, kknn, knitr, mapview, mda, naivebayes, nnet, parallelly, pdp, progressr, R.utils, randomForest, rgbif, rmarkdown, roxyglobals, rpart, RSNNS, Rtsne, sdm, tibble, usdm, withr, xgboost, testthat (>= 3.0.0)

Config/testthat/edition 3

Depends R (>= 4.2.0)

URL https://luizesser.github.io/caretSDM/

Config/Needs/website rmarkdown

Roxygen list(roclets = c(``collate'', ``namespace'', ``rd'', ``roxyglobals::global_roclet''))

Config/roxyglobals/filename generated-globals.R

Config/roxyglobals/unique FALSE

Config/roxygen2/version 8.0.0

Config/pak/sysreqs libabsl-dev cmake libgdal-dev gdal-bin libgeos-dev libglpk-dev make default-jdk libicu-dev libjpeg-dev libpng-dev libuv1-dev libxml2-dev libssl-dev libproj-dev libsqlite3-dev libudunits2-dev

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

Date/Publication 2026-06-08 02:40:38 UTC

RemoteUrl https://github.com/luizesser/caretsdm

RemoteRef HEAD

RemoteSha cff577f8e0dbbe679a49e0da9dc1f8caefc9bcb6

Contents

`add_predictors` *Add predictors to* `sdm_area`

Description

This function includes new predictors to the `sdm_area` object.

Usage

```
add_predictors(sa, pred, variables_selected = NULL, gdal = TRUE,
                       lines_as_sdm_area = FALSE)

get_predictors(i)
```

Arguments

`sa`	A `sdm_area` object.
`pred`	`RasterStack`, `SpatRaster`, `stars` or `sf` object with predictors data.
`variables_selected`	`character` vector with variables names in `pred` to be used as predictors. If `NULL` adds all variables.
`gdal`	Boolean. Force the use or not of GDAL when available. See details.
`lines_as_sdm_area`	Boolean. If `x` is a `sf` with LINESTRING geometry, it can be used to model species distribution in lines and not grid cells.
`i`	`input_sdm` or `sdm_area` object to retrieve data from.

Details

add_predictors returns a sdm_area object with a grid built upon the x parameter. There are two ways to make the grid and resample the variables in sdm_area: with and without gdal. As standard, if gdal is available in you machine it will be used (gdal = TRUE), otherwise sf/stars will be used. lines_as_sdm_area and gdal parameters are passed to sdm_area function, so they will be used in the grid creation and resampling of predictors. They will be retrieved automatically from the sdm_area object.

Value

For add_predictors the same input sdm_area object is returned including the pred data binded to the previous grid. get_predictors retrieves the grid from the i object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) and Reginaldo Ré. https://luizfesser.wordpress.com

See Also

sdm_area get_predictor_names bioc

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc)

# Retrieve predictors data:
get_predictors(sa)
```

---

add_scenarios *Add scenarios to* sdm_area

Description

This function includes scenarios in the sdm_area object.

Usage

```
add_scenarios(sa, scen = NULL, scenarios_names = NULL, pred_as_scen = TRUE,
                  variables_selected = NULL, stationary = NULL, crop_area = NULL)

set_scenarios_names(i, scenarios_names = NULL)

scenarios_names(i)
```

```
get_scenarios_data(i)

select_scenarios(i, scenarios_names = NULL)
```

Arguments

sa	A sdm_area or input_sdm object.
scen	RasterStack, SpatRaster or stars object. If NULL adds predictors as a scenario.
scenarios_names	Character vector with names of scenarios.
pred_as_scen	Logical. If TRUE adds the current predictors as a scenario.
variables_selected	Character vector with variables names in scen to be used as variables. If NULL adds all variables.
stationary	Names of variables from sa that should be used in scenarios as stationary variables.
crop_area	A sf object to crop the scen object if necessary.
i	A sdm_area or input_sdm object.

Details

The function add_scenarios adds scenarios to the sdm_area or input_sdm object. If scen has variables that are not present as predictors the function will use only variables present in both objects. stationary variables are those that don't change through the scenarios. It is useful for hidrological variables in fish habitat modeling, for example (see examples below). When adding multiple scenarios in multiple runs, the function will always add a new "current" scenario. To avoid that, set pred_as_scen = FALSE.

Value

add_scenarios returns the input sdm_area or input_sdm object with a new slot called scenarios with scen data as a list, where each slot of the list holds a scenario and each scenario is a sf object. set_scenarios_names sets new names for scenarios in sdm_area/input_sdm object. scenarios_names returns scenarios' names. get_scenarios_data retrieves scenarios data as a list of sf objects. select_scenarios selects scenarios from sdm_area/input_sdm object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

sdm_area input_sdm

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc)

# Include scenarios:
sa <- add_scenarios(sa, scen[1:2]) |> select_predictors(c("bio1", "bio12"))

# Set scenarios names:
sa <- set_scenarios_names(sa, scenarios_names = c(
  "future_1", "future_2",
  "current"
))
scenarios_names(sa)

# Get scenarios data:
scenarios_grid <- get_scenarios_data(sa)
scenarios_grid

# Select scenarios:
sa <- select_scenarios(sa, scenarios_names = c("future_1"))

# Setting stationary variables in scenarios:
sa <- sdm_area(rivs[c(1:200), ], cell_size = 100000, output_crs = 6933, lines_as_sdm_area = TRUE) |>
  add_predictors(bioc) |>
  add_scenarios(scen, stationary = c("LENGTH_KM", "DIST_DN_KM"))
```

---

algorithms *Caret Algorithms*

Description

A `data.frame` with characteristics of each algorithm available in `caretSDM`. Each column is a different characteristics. This can be helpful for more experienced modelers select algorithms. See the source for a selection method using this data.

Usage

```
algorithms
```

Format

'algorithms' A `data.frame` with 230 rows and 60 columns:

X Algorithms names

Further columns Algorithms attributes

Source

<https://topepo.github.io/caret/models-clustered-by-tag-similarity.html>

`background` *Obtain Background data*

Description

This function obtains background data given a set of predictors.

Usage

```
background(occ,
           pred = NULL,
           method = "random",
           n_set = 1,
           n_bg = 10000,
           proportion = NULL)

n_background(i)

background_method(i)

background_data(i)
```

Arguments

`occ`	A `occurrences_sdm` or `input_sdm` object.
`pred`	A `sdm_area` object. If `NULL` and `occ` is a `input_sdm`, `pred` will be retrieved from `occ`.
`method`	Method to obtain the background data. One of: "random" or a custom function (see details).
`n_set`	`numeric`. Number of datasets of background data to create.
`n_bg`	`numeric`. Number of background records to be generated in each dataset created. If `NULL` then the function prevents imbalance by using the same number of presence records (`n_records(occ)`). If you want to address different sizes to each species, you must provide a named vector (as in `n_records(occ)`).
`proportion`	`numeric`. A number between 0 and 1 representing a proportion of the area to be mapped as background. E.g.: if the whole area has 5,000 cells and proportion is 0.1, then `n_bg` is set to 500. Standard is NULL. This argument overwrites `n_bg`.
`i`	A `input_sdm` object.

Details

background is used in the SDM workflow to obtain background data, a step necessary for MaxEnt algorithm to run. This function helps avoid the use of pseudoabsence data in background algorithms and the use of background data in pseudoabsence algorithms, a very common mistake. If user provides a custom function, it must have the arguments env_sf and occ_sf, which will consist of two "sf"s. The first has the predictor values for the whole study area, while the second has the presence records for the species. The function must return a vector with cell_ids of the background data.

n_background returns the number of background records obtained per species.

background_data returns a list of species names. Each species name will have a lists with background data from class sf.

Value

A occurrences_sdm or input_sdm object with background data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

link{input_sdm} pseudoabsences occurrences_sdm get_occurrences get_predictors

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Background generation:
i <- background(i, proportion = 1) # All available data is obtained as background data.
```

bioc *Bioclimatic Variables*

Description

A stars object with bioclimatic variables (bio1, bio4 and bio12) for the Parana state in Brazil. Data obtained from WorldClim 2.1 at 10 arc-min resolution.

Usage

```
bioc
```

Format

'bioc' A stars with 1 attribute and 3 bands:

bio1 Annual Mean Temperature

bio4 Temperature Seasonality

bio12 Annual Precipitation

Source

<https://www.worldclim.org/>

buffer_sdm *Create buffer around occurrences*

Description

Create buffer around records in occ_data to be used as study area

Usage

```
buffer_sdm(occ_data, size = NULL, occ_crs = NULL, mcp = FALSE)
```

Arguments

occ_data	A data.frame object with species, decimalLongitude and decimalLatitude columns. Usually the output from GBIF_data.
size	numeric. The distance between the record and the margin of the buffer (i.e. buffer radius).
occ_crs	numeric. Indicates which EPSG is the occ_data in.
mcp	boolean. Should the buffer be applied in each record (FALSE) or in a minimum convex polygon/convex hull (TRUE)? Standard is FALSE.

Value

A `sf` buffer around `occ_data` records.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

`GBIF_data`

Examples

```
# Create sdm_area object:
study_area <- buffer_sdm(occ, size = 50000, occ_crs = 6933)
plot(study_area)
```

---

correlate_sdm *Correlation between projections*

Description

This function aims to unveil the correlation of different algorithms outputs. For that, it uses the predictions on current scenario, but other scenarios can be tested.

Usage

```
correlate_sdm(i, scenario = "current")
```

Arguments

`i`	A `input_sdm` object containing predictions.
`scenario`	A `character` containing scenario to be tested. Standard is `"current"`. Value must match `scenarios_names(i)`.

Value

A `data.frame` with pearson correlation between projections.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
if (interactive()) {
  # Create sdm_area object:
  set.seed(1)
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

  # Include scenarios:
  sa <- add_scenarios(sa)

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)

  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
    repeats = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
  i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)

  # Check correlations:
  correlate_sdm(i)
}
```

`data_clean` *Presence data cleaning routine*

Description

Data cleaning wrapper using CoordinateCleaner package.

Usage

```
data_clean(occ, pred = NULL,
           species = NA, lon = NA, lat = NA,
           capitals = TRUE,
           centroids = TRUE,
           duplicated = TRUE,
           identical = TRUE,
           institutions = TRUE,
           invalid = TRUE,
           terrestrial = TRUE,
           independent_test = TRUE,
           fun = NULL)
```

Arguments

occ	A occurrences_sdm object or input_sdm.
pred	A sdm_area object. If occ is a input_sdm object with predictors data, than pred is obtained from it.
species	A character stating the name of the column with species names in occ (see details).
lon	A character stating the name of the column with longitude in occ (see details).
lat	A character stating the name of the column with latitude in occ (see details).
capitals	Boolean to turn on/off the exclusion from countries capitals coordinates (see ?cc_cap)
centroids	Boolean to turn on/off the exclusion from countries centroids coordinates (see ?cc_cen)
duplicated	Boolean to turn on/off the exclusion from duplicated records (see ?cc_dupl)
identical	Boolean to turn on/off the exclusion from records with identical lat/long values (see ?cc_equ)
institutions	Boolean to turn on/off the exclusion from biodiversity institutions coordinates (see ?cc_inst)
invalid	Boolean to turn on/off the exclusion from invalid coordinates (see ?cc_val)
terrestrial	Boolean to turn on/off the exclusion from coordinates falling on sea (see ?cc_sea)
independent_test	Boolean. If occ has independent test data, the data cleaning routine is also applied on it.
fun	Function. A custom function to apply to occurrence data. It must receive a df argument, which will be a data.frame with three columns: species, decimalLongitude and decimalLatitude; The function must return the same data.frame with the same three columns.

Details

If the user does not used GBIF_data function to obtain species records, the function may have problems to find which column from the presences table has species, longitude and latitude information.

In this regard, we implemented the parameters `species`, `lon` and `lat` so the use can explicitly inform which columns should be used. If they remain as NA (standard) the function will try to guess which columns are the correct one.

Value

A `occurrences_sdm` object or `input_sdm` with cleaned presence data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

`GBIF_data` `occurrences_sdm` `sdm_area` `input_sdm` `get_predictor_names`

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Clean coordinates (terrestrial is set to false to make the run quicker):
i <- data_clean(i, terrestrial = FALSE)
```

---

ensemble_sdm *Ensemble Species Distribution Models*

Description

Calculates ensemble predictions for species distribution models using custom or implemented methods.

Usage

```
ensemble_sdm(m,
             scen = NULL,
             method = "average",
             metric = NULL,
             fun = NULL
```

```
            )

get_ensembles(
  i,
  type = "matrix",
  spp_name = NULL,
  scenario = NULL,
  ensemble_type = NULL
)

add_ensembles(e1, e2)
```

Arguments

m	A input_sdm or a models object.
scen	A scenarios object or NULL. If NULL and m is a input_sdm with a scenarios slot, it will be used.
method	Character or a function. Which ensembles should be calculated? See details.
metric	Character. Used with method = "weighted_average": Which metric should be used to weight predictions? If NULL
fun	Function. If method = "committee_average", the function will be used to binarize the data. It will receive caret's train object and must return a numeric value (the threshold, see details).
i	A input_sdm or a predictions object.
type	Character. Output format desired. One of "matrix", "sf", "stars", "raster", or "rast". Defaults to "matrix".
spp_name	Character or NULL. Name of the species to retrieve ensembles for. Defaults to the first available species if NULL.
scenario	Character or NULL. Name of the scenario to retrieve ensembles for. Defaults to the first available scenario if NULL.
ensemble_type	Character or NULL. The ensemble method to use for retrieval. Must be a subset of the methods stored in i$ensembles$method. Defaults to the first method if NULL.
e1	A ensembles object.
e2	A ensembles object.

Details

ensembles could be set to three different strategies OR a custom function. The three implemented strategies are: average is the mean occurrence probability, which is a simple mean of predictions; weighted_average is the same average, but weighted by a metric, which needs to be set using argument metric (see mean_validation_metrics for the metrics available). committee_average is the committee average, as known as majority rule, where predictions are binarized and then a mean is obtained. To binarize predictions, user can set a custom function in the fun argument to calculate a threshold for each model. Standardly, the committee average uses the caret::thresholder function to find the threshold that maximizes the sum of sensitivity and specificity (through caretSDM:::.MaxSeSp).

Custom function (fun) must use the argument mod, which is the model output from caret package (see get_models) and must return a numeric value (see example). method can also be set to a custom function, which must receive the argument pred_mat, which is a matrix of predictions (columns are models and rows are cells) and return a vector of predictions (one value per cell). See the median example below for a custom function.

get_predictions returns the list of all predictions to all scenarios, all species, all algorithms and all repetitions. Useful for those who wish to implement their own ensemble methods.

get_ensembles returns a matrix of data.frames, where each column is a scenario and each row is a species.

scenarios_names returns the scenarios names in a sdm_area or input_sdm object.

get_scenarios_data returns the data from scenarios in a sdm_area or input_sdm object.

Value

A input_sdm or a predictions object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

sdm_area input_sdm mean_validation_metrics

Examples

```
if (interactive()) {
  # Create sdm_area object:
  set.seed(1)
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

  # Include scenarios:
  sa <- add_scenarios(sa)

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)

  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
```

```
    repeats = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
  i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)

  # Ensemble:
  i <- ensemble_sdm(i, method = "average")
  i
}

# Example from a custom function to obtain the threshold that maximizes
# the sensitivity plus specificity:
MaxSeSp <- function(mod) {
  th <- caret::thresholder(mod,
    threshold = seq(0, 1, by = 0.001),
    final = TRUE,
    statistics = c("Sensitivity", "Specificity")
  )
  th <- th$prob_threshold[which.max(th$Sensitivity + th$Specificity)]
  if (length(th) > 1) mean(th) else th
}

# Example from a custom function to obtain ensembles using the median instead of the mean:
median_ensemble <- function(pred_mat) {
  apply(pred_mat, 1, median, na.rm = TRUE)
}
```

GBIF_data *Retrieve Species data from GBIF*

Description

This function is a wrapper to get records from GBIF using `rgbif` and return a `data.frame` ready to be used in caretSDM.

Usage

```
GBIF_data(s, file = NULL, as_df = FALSE, ...)
```

Arguments

s	character vector of species names.
file	character with file to save the output. If not informed, data will not be saved on folder.
as_df	Should the output be a dataframe? Default is FALSE, returning a occurrences object.
...	Arguments to pass on rgbif::occ_data().

Value

A data.frame with species occurrences data, or an occurrences object if as_df = FALSE.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

References

https://www.gbif.org

Examples

```
## Select species names:
# s <- c("Araucaria angustifolia", "Salminus brasiliensis")

## Run function:
# oc <- GBIF_data(s)
```

---

gcms_ensembles *Ensemble GCMs into one scenario*

Description

An ensembling method to group different GCMs into one SSP scenario

Usage

```
gcms_ensembles(i, gcms = NULL)
```

Arguments

i	A input_sdm object.
gcms	GCM codes in scenarios_names(i) to group scenarios.

Value

A input_sdm object with grouped GCMs.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

GBIF_data occurrences_sdm sdm_area input_sdm get_predictor_names

Examples

```
if (interactive()) {
  # Create sdm_area object:
  set.seed(1)
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc)

  # Include scenarios:
  sa <- add_scenarios(sa, scen) |> select_predictors(c("bio1", "bio12"))

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)

  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
  i <- train_sdm(i,
    algo = c("naive_bayes"),
    ctrl = ctrl_sdm,
    variables_selected = c("bio1", "bio12")
  ) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)

  # Ensemble:
  i <- ensemble_sdm(i, method = "average")
```

```
  i

  # Ensemble GCMs:
  i <- gcms_ensembles(i, gcms = c("ca", "mi"))
  i
}
```

---

input_sdm input_sdm

Description

This function creates a new input_sdm object.

Usage

```
input_sdm(...)

add_input_sdm(i1, i2)
```

Arguments

...	Data to be used in SDMs. Can be a occurrences and/or a sdm_area object.
i1	A input_sdm object.
i2	A input_sdm object.

Details

If sdm_area is used, it can include predictors and scenarios. In this case, input_sdm will detect and include as scenarios and predictors in the input_sdm output. Objects can be included in any order, since the function will work by detecting their classes. The returned object is used throughout the whole workflow to apply functions.

Value

A input_sdm object containing:

grid	sf with POLYGON geometry representing the grid for the study area or LINESTRING if sdm_area was built with a LINESTRING sf.
bbox	Four corners for the bounding box (class bbox): minimum value of X, minimum value of Y, maximum value of X, maximum value of Y
cell_size	numeric information regarding the size of the cell used to rescale variables to the study area, representing also the cell size in the grid.
epsg	character information about the EPSG used in all slots from sdm_area.
predictors	character vector with predictors names included in sdm_area.

Author(s)

Luiz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

occurrences_sdm sdm_area

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa, scen)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)
```

---

is_input_sdm is_class *functions to check caretSDM data classes.*

Description

This functions returns a boolean to check caretSDM object classes.

Usage

```
is_input_sdm(x)

is_sdm_area(x)

is_occurrences(x)

is_models(x)

is_predictions(x)
```

Arguments

x	Object to be tested.

Value

Boolean.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

is_sdm_area(sa)

is_input_sdm(sa)
```

---

join_area *Join Area*

Description

Join cell_id data from sdm_area to a occurrences

Usage

```
join_area(occ, pred)
```

Arguments

occ	A occurrences object or input_sdm.
pred	A sdm_area object to retrieve cell_id from.

Details

This function is key in this SDM workflow. It attaches cell_id values to occ, deletes records outside pred and allows the use of pseudoabsences. This function also tests if CRS from both occ and pred are equal, otherwise the CRS of pred is used to convert occ.

Value

A occurrences object with cell_id to each record.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

occurrences_sdm sdm_area input_sdm pseudoabsences

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa, scen)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933) |> join_area(sa)
```

---

multicollinearity_sdm *Multicollinearity Analysis*

Description

Apply multicollinearity calculation on predictors.

Usage

```
multicollinearity_sdm(pred,
                      method = NULL,
                      variables_selected = NULL,
                      cumulative_proportion = 0.99,
                      th = 0.5,
                      ...)

selected_variables(i)
```

Arguments

pred	A input_sdm or predictors object.
method	Which method should be used to detect multicollinearity. Can be a character or a custom function.
variables_selected	A vector with pre-selected variables names to filter variables.
cumulative_proportion	A numeric with the threshold for cumulative proportion in PCA. Standard is 0.99, meaning that axes returned as predictors sum up more than 99 environmental variance.

th	Threshold to be applied in VIF routine. See ?usdm::vifcor.
...	Further arguments to be passed to the applied method.
i	A input_sdm object.

Details

multicollinearity_sdm is a wrapper function to run usdm::vifcor, usdm::vifstep or a pca in caretSDM, but also provides a way to implement custom functions to reduce multicollinearity. If user provides a custom function, it must have the arguments env_sf and occ_sf, which will consist of two sfs. The first has the predictor values for the whole study area, while the second has the presence records for the species. The function must return a vector with selected variables.

Value

A input_sdm or predictors object with VIF data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

vif_predictors pca_predictors get_predictor_names

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa, scen)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# VIF calculation:
i <- multicollinearity_sdm(i, method = "vifcor", th = 0.5)
i

# Retrieve information about vif:
vif_summary(i)
selected_variables(i)

# Example of custom function:
custom_function <- function(env_sf, occ_sf) {
```

```
  env_df <- dplyr::select(sf::st_drop_geometry(env_sf), -"cell_id")
  correlations <- cor(env_df)
  col <- caret::findCorrelation(correlations, cutoff = 0.7)
  selected <- colnames(correlations)[-col]
  return(selected)
}
```

---

occ *Araucaria angustifolia occurrence data*

Description

A `data.frame` object with Araucaria angustifolia occurrence data obtained from GBIF and filtered with Parana state `sf`.

Usage

```
occ
```

Format

'occ' A `data.frame` with 420 rows and 3 columns (EPSG:6933):

species Species name

decimalLongitude Longitude in meters

decimalLatitude Latitude in meters

Source

<https://www.gbif.org>

---

occurrences_sdm *Occurrences Managing*

Description

This function creates and manage `occurrences` objects.

Usage

```
occurrences_sdm(occ,
                independent_test = NULL,
                p = 0.1,
                occ_crs = NULL,
                independent_test_crs = NULL,
                crs = NULL,
                ...)

n_records(i)

species_names(i)

get_coords(i)

get_occurrences(i)

occurrences_as_df(i)

add_occurrences(oc1, oc2)
```

Arguments

occ	A data.frame, tibble or sf with species records.
independent_test	Boolean. If independet_test is TRUE, a fraction of the data is kept for independent testing. Otherwise, the whole dataset x is used. It can also be a data.frame or a sf, with species records to be used as independent test. Structure and names should be identical to those in x.
p	Numeric. Fraction of data to be used as independent test. Standard is 0.1.
occ_crs	Numeric. CRS of occ.
independent_test_crs	Numeric. CRS of independent_test if it is a data.frame.
crs	Deprecated. Use occ_crs instead.
...	A vector with column names addressing the columns with species names, longitude and latitude, respectively, in occ.
i	input_sdm or occurrences object.
oc1	A occurrences object to be summed with.
oc2	A occurrences object to be summed with.

Details

occ must have three columns: species, decimalLongitude and decimalLatitude. When sf it is only necessary a species column. n_records return the number of presence records to each species. species_names return the species names. get_coords return a data.frame with coordinates of species records. get_occurrences return a sf with coordinates of species records, species names

and cell_ids. add_occurrences return a occurrences. This function sums two occurrences objects. It can also sum a occurrences object with a data.frame object. occurrences_as_df returns a data.frame with species names and coordinates.

Value

A occurrences object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

input_sdm GBIF_data occ

Examples

```
# Create occurrences:
oc <- occurrences_sdm(occ, crs = 6933)
```

---

parana *Paraná State*

Description

A sf object with a polygon for the Paraná state in Brazil. This is a subset of the brazilian map provided by official government agency (IBGE)

Usage

```
parana
```

Format

'parana' A sf with 1 row and 5 columns:

GID0 State code

CODIGOIB1 State's phone code

NOMEUF2 Name of the state

SIGLAUF3 Abbreviation of the state's name

geom Geometry column of the sf

Source

<https://www.ibge.gov.br/geociencias/cartas-e-mapas/bases-cartograficas-continuas/15759-brasil.html>

pca_predictors *Predictors as PCA-axes*

Description

Transform predictors data into PCA-axes.

Usage

```
pca_predictors(i, cumulative_proportion = 0.99)

pca_summary(i)

get_pca_model(i)
```

Arguments

i	A input_sdm object.
cumulative_proportion	A numeric with the threshold for cumulative proportion. Standard is 0.99, meaning that axes returned as predictors sum up more than 99 variance.

Details

pca_predictors Transform predictors data into PCA-axes. If the user wants to use PCA-axes as future scenarios, then scenarios should be added after the PCA transformation (see examples). pca_summary Returns the summary of prcomp function. See ?stats::prcomp. get_pca_model Returns the model built to calculate PCA-axes.

Value

input_sdm object with variables from both predictors and scenarios transformed in PCA-axes.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

vif_predictors sdm_area add_scenarios add_predictors

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))
```

```
# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# PCA transformation:
i <- pca_predictors(i)
```

---

pdp_sdm *Model Response to Variables*

Description

Obtain the Partial Dependence Plots (PDP) to each variable.

Usage

```
pdp_sdm(i, spp = NULL, algo = NULL, variables_selected = NULL, mean.only = FALSE)

get_pdp_sdm(i, spp = NULL, algo = NULL, variables_selected = NULL)
```

Arguments

i	A input_sdm object.
spp	A character vector with species names to obtain the PDPs. If NULL (standard), the first species in species_names(i) is used.
algo	A character containing the algorithm to obtain the PDP. If NULL (standard) all algorithms are mixed.
variables_selected	A character. If there is a subset of predictors that should be ploted in this, it can be informed using this parameter.
mean.only	Boolean. Should only the mean curve be plotted or a curve to each run should be included? Standard is FALSE.

Value

A plot (for pdp_sdm) or a data.frame (for get_pdp_sdm) with PDP values.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

varImp_sdm

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random", n_set = 3)

# Custom trainControl:
ctrl_sdm <- caret::trainControl(
  method = "repeatedcv",
  number = 2,
  repeats = 1,
  classProbs = TRUE,
  returnResamp = "all",
  summaryFunction = summary_sdm,
  savePredictions = "all"
)
# Train models:
i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm)

# PDP plots:
pdp_sdm(i)
get_pdp_sdm(i)
```

---

plot_occurrences *S3 Methods for plot and mapview*

Description

This function creates different plots depending on the input.

Usage

```
plot_occurrences(i, spp_name = NULL, pa = TRUE, pa_id = 1, ...)

plot_grid(i, ...)
```

```
plot_predictors(i, variables_selected = NULL, ...)

plot_scenarios(i, variables_selected = NULL, scenario = NULL, ...)

plot_predictions(i, spp_name = NULL, scenario = NULL, id = NULL, ...)

plot_ensembles(
  i,
  spp_name = NULL,
  scenario = NULL,
  id = NULL,
  ensemble_type = NULL,
  ...
)

mapview_grid(i)

mapview_occurrences(i, spp_name = NULL, pa = TRUE)

mapview_predictors(i, variables_selected = NULL)

mapview_scenarios(i, variables_selected = NULL, scenario = NULL)

mapview_predictions(i, spp_name = NULL, scenario = NULL, id = NULL)

mapview_ensembles(
  i,
  spp_name = NULL,
  scenario = NULL,
  id = NULL,
  ensemble_type = NULL
)

plot_background(i, variables_selected = NULL, ...)

plot_niche(
  i,
  spp_name = NULL,
  variables_selected = NULL,
  scenario = NULL,
  id = NULL,
  ensemble_type = NULL,
  raster = FALSE,
  ...
)
```

Arguments

i	Object to be plotted. Can be a input_sdm, but also occurrences or sdm_area.
spp_name	A character with species to be plotted. If NULL, the first species is plotted.
pa	Boolean. Should pseudoabsences be plotted together? (not implemented yet.)
pa_id	The id of pseudoabsences to be plotted (only used when pa = TRUE). Possible values are numeric values from 1 to number of PA sets.
...	Plotting arguments to pass to ggplot2 function.
variables_selected	A character vector with names of variables to be plotted.
scenario	description
id	The id of models to be plotted (only used when ensemble = FALSE). Possible values are row names of get_validation_metrics(i).
ensemble_type	Character of the type of ensemble to be plotted.
raster	Should the niche be extrapolated to a raster covering all possibe values in the environmental space?

Details

We implemented a bestiary of plots to help visualizing the process and results. If you are not familiar with mapview, consider using it to better visualize maps.

Value

The plot or mapview desired.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

WorldClim_data

---

predict_sdm *Predict SDM models in new data*

Description

This function projects SDM models to new scenarios

Usage

```
predict_sdm(m,
            scen = NULL,
            metric = "ROC",
            th = 0.9,
            tp = "prob",
            file = NULL,
            add.current = TRUE)

get_predictions(i)

add_predictions(p1, p2)
```

Arguments

m	A input_sdm or a models object.
scen	A scenarios object or NULL. If NULL and m is a input_sdm with a scenarios slot, it will be used.
metric	A character containing the metric in which the th will be calculated/applied. Default is ROC. See ?mean_validation_metrics for the metrics available.
th	Thresholds for metrics. Can be numeric or a function.
tp	Type of output to be retrieved. See details.
file	File to sabe predictions.
add.current	If current scenario is not available, predictors will be used as the current scenario.
i	A input_sdm or a predictions object.
p1	A predictions object.
p2	A predictions object.

Details

tp is a parameter to be passed on caret to retrieve either the probabilities of classes (tp="prob") or the raw output (tp="raw"), which could vary depending on the algorithm used, but usually would be on of the classes (factor vector with presences and pseudoabsences).

get_predictions returns the list of all predictions to all scenarios, all species, all algorithms and all repetitions. Useful for those who wish to implement their own ensemble methods.

scenarios_names returns the scenarios names in a sdm_area or input_sdm object.

get_scenarios_data returns the data from scenarios in a sdm_area or input_sdm object.

Value

A input_sdm or a predictions object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

sdm_area input_sdm mean_validation_metrics

Examples

```
if (interactive()) {
  # Create sdm_area object:
  set.seed(1)
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

  # Include scenarios:
  sa <- add_scenarios(sa)

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)

  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
    repeats = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
  i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)
  i
}
```

---

prediction_change_sdm *Prediction Change Analysis*

Description

Provides an automate way for the visualization of projections gain, loss, and stability between different scenarios.

Usage

```
prediction_change_sdm(i, scenario = NULL, ensemble_type = NULL, species = NULL, th = 0.5)
```

Arguments

i	A input_sdm object with projections.
scenario	Character. One of the scenarios that were projected. Can be ensembles as well.
ensemble_type	Character. Type of ensemble to be used. Standard is NULL, but will return the average.
species	Character. Species to be analyzed. Standard is NULL.
th	Numeric. Threshold to binarize the ensemble.

Value

A plot with comparison between current and other scenario.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

species_names scenarios_names

Examples

```
if (interactive()) {
  # Create sdm_area object:
  set.seed(1)
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc)

  # Include scenarios:
  sa <- add_scenarios(sa, scen) |> select_predictors(c("bio1", "bio12"))

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)
```

```
  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
  i <- train_sdm(i,
    algo = c("naive_bayes"),
    ctrl = ctrl_sdm,
    variables_selected = c("bio1", "bio12")
  ) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)

  # Ensemble:
  i <- ensemble_sdm(i, method = "average")

  # Ensemble GCMs:
  i <- gcms_ensembles(i, gcms = c("ca", "mi"))
  i

  # Change Analysis
  prediction_change_sdm(i, scenario = "_ssp585_2090", ensemble_type = "mean_occ_prob")
}
```

`print.ensembles` *Print method for ensembles*

Description

Print method for ensembles

Usage

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

Arguments

`x`	ensembles object
`...`	passed to other methods

Value

Concatenate structured characters to showcase what is stored in the object.

`print.input_sdm` *Print method for input_sdm*

Description

Print method for input_sdm

Usage

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

Arguments

`x`	input_sdm object
`...`	passed to other methods

Value

Concatenate structured characters to showcase what is stored in the object.

`print.models` *Print method for models*

Description

Print method for models

Usage

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

Arguments

`x`	models object
`...`	passed to other methods

Value

Concatenate structured characters to showcase what is stored in the object.

print.occurrences *Print method for occurrences*

Description

Print method for occurrences

Usage

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

Arguments

x	occurrences object
...	passed to other methods

Value

Concatenate structured characters to showcase what is stored in the object.

print.predictions *Print method for predictions*

Description

Print method for predictions

Usage

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

Arguments

x	predictions object
...	passed to other methods

Value

Concatenate structured characters to showcase what is stored in the object.

pseudoabsences *Obtain Pseudoabsences*

Description

This function obtains pseudoabsences given a set of predictors.

Usage

```
pseudoabsences(occ,
               pred = NULL,
               method = "random",
               n_set = 10,
               n_pa = NULL,
               variables_selected = NULL,
               th = 0,
               size = 1,
               size_crs = 4326,
               mcp = FALSE)

n_pseudoabsences(i)

pseudoabsence_method(i)

pseudoabsence_data(i)
```

Arguments

occ	A occurrences_sdm or input_sdm object.
pred	A sdm_area object. If NULL and occ is a input_sdm, pred will be retrieved from occ.
method	Method to create pseudoabsences. One of: "random", "bioclim", "mahal.dist" or "buffer_sdm". User can also provide a custom function (see details).
n_set	numeric. Number of datasets of pseudoabsence to create.
n_pa	numeric. Number of pseudoabsences to be generated in each dataset created. If NULL then the function prevents imbalance by using the same number of presence records (n_records(occ)). If you want to address different sizes to each species, you must provide a named vector (as in n_records(occ)).
variables_selected	A vector with variables names to be used while building pseudoabsences. Only used when method is not "random".
th	numeric Threshold to be applied in bioclim/mahal.dist projections. See details.
size	numeric The distance between the record and the margin of the buffer (i.e. buffer radius).
size_crs	numeric Indicates which EPSG it the size in.

mcp	boolean. Should the buffer be applied in each record (FALSE) or in a minimum convex polygon/convex hull (TRUE)? Standard is FALSE.
i	A input_sdm object.

Details

pseudoabsences is used in the SDM workflow to obtain pseudoabsences, a step necessary for most of the algorithms to run. We implemented four methods: "random", which is self-explanatory, "buffer_sdm", "mahal.dist" and "bioclim". The two last are built with the idea that pseudoabsences should be environmentally different from presences. Thus, we implemented two presence-only methods to infer the distribution of the species. "bioclim" uses an envelope approach (bioclimatic envelope), while "mahal.dist" uses a distance approach (mahalanobis distance). th parameter enters here as a threshold to binarize those results. Pseudoabsences are retrieved outside the projected distribution of the species. If user provides a custom function, it must have the arguments env_sf and occ_sf, which will consist of two "sf"s. The first has the predictor values for the whole study area, while the second has the presence records for the species. The function must return a vector with cell_ids of the pseudoabsences, which is the first column of both objects. For buffer_sdm, user needs to specifiy the size of the buffer compatible with buffer CRS.

n_pseudoabsences returns the number of pseudoabsences obtained per species.

pseudoabsence_method returns the method used to obtain pseudoabsences.

pseudoabsence_data returns a list of species names. Each species name will have a lists with pseudoabsences data from class sf.

Value

A occurrences_sdm or input_sdm object with pseudoabsence data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

link{input_sdm} background occurrences_sdm get_occurrences get_predictors

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ[1:50, ], occ_crs = 6933)
```

```
# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i0 <- pseudoabsences(i, method = "random")

# Custom method example:
buffer_pa_custom <- function(env_sf, occ_sf, buffer_dist = 3) {
  # Create buffer around occurrence points
  buffer <- sf::st_buffer(occ_sf, dist = buffer_dist)

  # Union buffers into a single geometry
  buffer_union <- sf::st_union(buffer)

  # Identify cells outside the buffer
  outside_buffer <- sf::st_difference(env_sf, buffer_union)[, 1]

  # Randomly extract cell_ids outside the buffer
  pa_ids_sample <- sample(outside_buffer$cell_id, nrow(occ_sf))

  return(pa_ids_sample)
}

i1 <- pseudoabsences(i, method = buffer_pa_custom)
```

`rivs` *Hydrologic Variables*

Description

A `sf` LINESTRING object with hydrologic variables (LENGTH_KM and DIST_DN_KM) for the Paraná state in Brazil. Data obtained from HydroSHEDS for river flows >= 10m3/s.

Usage

```
rivs
```

Format

'rivs' A `sf` with 1031 attributes and 2 fiels:

LENGTH_KM Length of the river reach segment, in kilometers.

DIST_DN_KM Distance from the reach outlet, i.e., the most downstream pixel of the reach, to the final downstream location along the river network, in kilometers. This downstream location is either the pour point into the ocean or an endorheic sink.

Source

<https://www.hydrosheds.org/>

salm *Salminus brasiliensis occurrence data*

Description

A `data.frame` object with Salminus brasiliensis occurrence data obtained from GBIF and filtered with Parana state `sf`.

Usage

```
salm
```

Format

'salm' A `data.frame` with 46 rows and 3 columns (EPSG:6933):

species Species name

decimalLongitude Longitude in meters

decimalLatitude Latitude in meters

Source

<https://www.gbif.org>

scen *Bioclimatic Variables*

Description

A `stars` object with bioclimatic variables (bio1, bio4 and bio12) and four future scenarios for the Parana state in Brazil. Data from MIROC6 GCM from WorldClim 2.1 at 10 arc-min resolution.

Usage

```
scen
```

Format

'scen' A `stars` with 4 attribute and 3 bands:

ca_ssp245_2090 Intermediate scenario for the year 2090 and GCM CanESM5

ca_ssp585_2090 Extreme scenario for the year 2090 and GCM CanESM5

mi_ssp245_2090 Intermediate scenario for the year 2090 and GCM MIROC6

mi_ssp585_2090 Extreme scenario for the year 2090 and GCM MIROC6

bio1 Annual Mean Temperature

bio4 Temperature Seasonality

bio12 Annual Precipitation

Source

<https://www.worldclim.org/>

`scen_rs` *Bioclimatic Variables*

Description

A `stars` object with bioclimatic variables (bio1, bio4 and bio12) and four future scenarios for the Rio Grande do Sul state in Brazil. Data from MIROC6 GCM from WorldClim 2.1 at 10 arc-min resolution.

Usage

```
scen_rs
```

Format

'scen_rs' A `stars` with 5 attribute and 3 bands:

current Current scenario with the average values for the years 1970-2000

ca_ssp245_2090 Intermediate scenario for the year 2090 and GCM CanESM5

ca_ssp585_2090 Extreme scenario for the year 2090 and GCM CanESM5

mi_ssp245_2090 Intermediate scenario for the year 2090 and GCM MIROC6

mi_ssp585_2090 Extreme scenario for the year 2090 and GCM MIROC6

bio1 Annual Mean Temperature

bio4 Temperature Seasonality

bio12 Annual Precipitation

Source

<https://www.worldclim.org/>

sdm_area *Create a* sdm_area *object*

Description

This function creates a new sdm_area object.

Usage

```
sdm_area(x, cell_size = NULL, output_crs = NULL, variables_selected = NULL,
             gdal = TRUE, crop_by = NULL, lines_as_sdm_area = FALSE, crs = NULL)

get_sdm_area(i)

add_sdm_area(sa1, sa2)
```

Arguments

x	A shape or a raster. Usually a shape from sf class, but rasters from stars, rasterStack or SpatRaster class are also allowed.
cell_size	numeric. The cell size to be used in models.
output_crs	numeric. Indicates which EPSG should the output grid be in. If NULL, epsg from x is used.
variables_selected	A character vector with variables in x to be used in models. If NULL (standard), all variables in x are used.
gdal	Boolean. Force the use or not of GDAL when available. See details.
crop_by	A shape from sf to crop x.
lines_as_sdm_area	Boolean. If x is a sf with LINESTRING geometry, it can be used to model species distribution in lines and not grid cells.
crs	Deprecated. Use output_crs instead.
i	A sdm_area or a input_sdm object.
sa1	A sdm_area object.
sa2	A sdm_area object.

Details

The function returns a sdm_area object with a grid built upon the x parameter. There are two ways to make the grid and resample the variables in sdm_area: with and without gdal. As standard, if gdal is available in you machine it will be used (gdal = TRUE), otherwise sf/stars will be used. get_sdm_area will return the grid built by sdm_area. add_sdm_area will sum two sdm_area objects. As geoprocessing in caretSDM is performed using sf objects, add_sdm_area simply applies a rbind in the two different areas.

Value

A sdm_area object containing:

grid	sf with POLYGON geometry representing the grid for the study area.
cell_size	numeric information regarding the size of the cell used to rescale variables to the study area, representing also the cell size in the grid.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) and Reginaldo Ré. https://luizfesser.wordpress.com

See Also

WorldClim_data parana input_sdm, add_predictors

Examples

```
# Create sdm_area object:
sa_area <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Create sdm_area using a subset of rivs (lines):
sa_rivers <- sdm_area(rivs[c(1:100), ],
                      cell_size = 100000,
                      output_crs = 6933,
                      lines_as_sdm_area = TRUE)
```

---

sdm_as_stars sdm_as_X *functions to transform* caretSDM *data into other classes.*

Description

This functions transform data from a caretSDM object to be used in other packages.

Usage

```
sdm_as_stars(x,
             what = NULL,
             spp = NULL,
             scen = NULL,
             id = NULL,
             ens = NULL)

sdm_as_raster(x, what = NULL, spp = NULL, scen = NULL, id = NULL, ens = NULL)

sdm_as_terra(x, what = NULL, spp = NULL, scen = NULL, id = NULL, ens = NULL)
```

Arguments

x	A `caretSDM` object.
what	Sometimes multiple data inside `x` could be transformed. This parameter allows users to specify what needs to be converted.It can be one of: "predictors", "scenarios", "predictions" or "ensembles".
spp	`character`. Which species should be converted?
scen	`character`. Which scenario should be converted?
id	`character`. Which id should be converted?
ens	`character`. Which ensemble should be converted?

Value

The output is the desired class.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
if (interactive()) {
  # Create sdm_area object:
  sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

  # Include predictors:
  sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

  # Include scenarios:
  sa <- add_scenarios(sa)

  # Create occurrences:
  oc <- occurrences_sdm(occ, occ_crs = 6933)

  # Create input_sdm:
  i <- input_sdm(oc, sa)

  # Pseudoabsence generation:
  i <- pseudoabsences(i, method = "random", n_set = 2)

  # Custom trainControl:
  ctrl_sdm <- caret::trainControl(
    method = "boot",
    number = 1,
    classProbs = TRUE,
    returnResamp = "all",
    summaryFunction = summary_sdm,
    savePredictions = "all"
  )

  # Train models:
```

```
  i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
    suppressWarnings()

  # Predict models:
  i <- predict_sdm(i, th = 0.8)

  # Transform in stars:
  sdm_as_stars(i)
}
```

`select_predictors` *Tidyverse methods for caretSDM objects*

Description

Set of functions to facilitate the use of caretSDM through tidyverse grammatics.

Usage

```
select_predictors(x, ...)

## S3 method for class 'sdm_area'
select(.data, ...)

## S3 method for class 'input_sdm'
select(.data, ...)

## S3 method for class 'sdm_area'
mutate(.data, ...)

## S3 method for class 'input_sdm'
mutate(.data, ...)

## S3 method for class 'sdm_area'
filter(.data, ..., .by, .preserve)

## S3 method for class 'input_sdm'
filter(.data, ..., .by, .preserve)

## S3 method for class 'occurrences'
filter(.data, ..., .by, .preserve)

filter_species(x, spp = NULL, ...)
```

Arguments

x	sdm_area or input_sdm object.
...	character arguments to pass to the given function.
.data	Data to pass to tidyr function.
.by	See ?dplyr::filter.
.preserve	See ?dplyr::filter.
spp	Species to be filtered.

Value

The transformed sdm_area/input_sdm object.

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))
```

---

set_predictor_names *Predictors Names Managing*

Description

This function manage predictors names in sdm_area objects.

Usage

```
get_predictor_names(x)

## S3 method for class 'input_sdm'
set_predictor_names(x, new_names)

## S3 method for class 'sdm_area'
set_predictor_names(x, new_names)

get_predictor_names(x)

test_variables_names(sa, scen)

set_variables_names(s1 = NULL, s2 = NULL, new_names = NULL)
```

Arguments

x	A sdm_area or input_sdm object to get/set predictors names.
new_names	A character vector from size length(get_predictor_names(x))
sa	A sdm_area object.
scen	A stars object with scenarios.
s1	A stars object with scenarios.
s2	A stars object with scenarios or a sdm_area object.

Details

This functions is available so users can modify predictors names to better represent them. Use carefully to avoid giving wrong names to the predictors. Useful to make sure the predictors names are equal the names in scenarios. test_variables_names Tests if variables in a stars object (scen argument) matches the given sdm_area object (sa argument). set_variables_names will set s1 object variables names as the s2 object variables names OR assign new names to it.

Value

get_predictor_names returns a character vector with predictors names. test_variables_names returns a logical informing if all variables are equal in both objects (TRUE) or not (FALSE). set_variables_names returns the s1 object with new names provided by s2 or new_names.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

parana sdm_area

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 50000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc)

# Check predictors' names:
get_predictor_names(sa)
```

stack_sdm *Train a Stacked Ensemble for SDM*

Description

This function builds a meta-model (Layer 2) using the out-of-fold predictions from models trained in Layer 1.

Usage

```
stack_sdm(m, meta_algo = "glm", ctrl = NULL, ...)
```

Arguments

m	A models or input_sdm object.
meta_algo	A character string specifying the algorithm for the meta-learner.
ctrl	A trainControl object for the meta-learner. If NULL, a simple CV is used.
...	Additional arguments passed to caret::train.

Value

A stacked_models object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

input_sdm sdm_area algorithms train_sdm

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)
```

```
# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random")

# Custom trainControl:
ctrl_sdm <- caret::trainControl(method = "repeatedcv",
                                number = 2,
                                repeats = 1,
                                classProbs = TRUE,
                                returnResamp = "all",
                                summaryFunction = summary_sdm,
                                savePredictions = "all")

# Train models:
i <- train_sdm(i, algo = c("naive_bayes", "kknn"), ctrl = ctrl_sdm) |>
suppressWarnings()

# Train stacked ensemble:
i <- stack_sdm(i, meta_algo = "nnet", ctrl = ctrl_sdm)
```

`summary_sdm` *Calculates performance across resamples*

Description

This function is used in `caret::trainControl(summaryFunction=summary_sdm)` to calculate performance metrics across resamples.

Usage

```
summary_sdm(data, lev = NULL, model = NULL, custom_fun=NULL)

summary_sdm_presence_only(data, lev, threshold)

validate_on_independent_data(model, data_independent, obs_col_name)
```

Arguments

`data`	A `data.frame` with observed and predicted values.
`lev`	A `character` vector of factors levels for the response.
`model`	Models names taken from `train` object.
`custom_fun`	A custom function to be applied in models (not yet implemented).
`threshold`	Threshold for presence-only models.
`data_independent`	independent data.frame to calculate metrics.
`obs_col_name`	The name of the column with observed values.

Details

See ?caret::defaultSummary for more details and options to pass on caret::trainControl.

Value

A input_sdm or a predictions object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

train_sdm

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random")

# Custom trainControl:
ctrl_sdm <- caret::trainControl(method = "repeatedcv",
                                number = 2,
                                repeats = 1,
                                classProbs = TRUE,
                                returnResamp = "all",
                                summaryFunction = summary_sdm,
                                savePredictions = "all")

# Train models:
i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
suppressWarnings()
```

train_sdm *Train SDM models*

Description

This function is a wrapper to fit models in caret using caretSDM data.

Usage

```
train_sdm(occ,
          pred = NULL,
          algo,
          ctrl = NULL,
          variables_selected = NULL,
          parallel = FALSE,
          ...)

get_tune_length(i)

algorithms_used(i)

get_models(i)

get_validation_metrics(i)

mean_validation_metrics(i)

models_hyperparameters(i)

add_models(m1, m2)
```

Arguments

occ	A occurrences or a input_sdm object.
pred	A predictors object. If occ is a input_sdm object, then pred is obtained from it.
algo	A character vector. Algorithms to be used. For a complete list see (https://topepo.github.io/caret/availabl models.html) or in caretSDM::algorithms.
ctrl	A trainControl object to be used to build models. See ?caret::trainControl and details.
variables_selected	A vector of variables to be used as predictors. If NULL, predictors names from pred will be used. Can also be a selection method (e.g. 'vif').
parallel	Should a paralelization method be used (not yet implemented)?
...	Additional arguments to be passed to caret::train function.

i	A models or a input_sdm object.
m1	A models object.
m2	A models object.

Details

The object algorithms has a table comparing algorithms available. If the function detects that the necessary packages are not available it will ask for installation. This will happen just in the first time you use the algorithm. caret::trainControl holds multiple resources for validation and model tuning. Make sure to understand its parameters beforehand. As it is a key function in the modeling process, we also implemented spatial crossvalidation on it. You can set methods to be cv_spatial or cv_cluster and train_sdm will detect that and apply the method according to blockCV package.

get_tune_length return the length used in grid-search for tunning.

algorithms_used return the names of the algorithms used in the modeling process.

get_models returns a list with trained models (class train) to each species.

get_validation_metrics return a list with a data.frame to each species with complete values for ROC, Sensitivity, Specificity, with their respectives Standard Deviations (SD) and TSS to each of the algorithms and pseudoabsence datasets used.

mean_validation_metrics return a list with a tibble to each species summarizing values for ROC, Sensitivity, Specificity and TSS to each of the algorithms used.

models_hyperparameters returns the hyperparameters that returned the best tuning to each model to each species.

Value

A models or a input_sdm object.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

input_sdm sdm_area algorithms

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
```

```
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random")

# Custom trainControl:
ctrl_sdm <- caret::trainControl(
  method = "repeatedcv",
  number = 2,
  repeats = 1,
  classProbs = TRUE,
  returnResamp = "all",
  summaryFunction = summary_sdm,
  savePredictions = "all"
)

# Train models:
i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
  suppressWarnings()
```

`tsne_sdm` *tSNE*

Description

This function calculates tSNE with presences and pseudoabsences data and returns a list of plots.

Usage

```
tsne_sdm(occ, pred = NULL, variables_selected = NULL)
```

Arguments

`occ`	A `occurrences` or `input_sdm` object.
`pred`	A `predictors` object. If `occ` is of class `input_sdm`, then `pred` is retrieved from it.
`variables_selected`	Variable to be used in t-SNE. It can also be 'vif', if previously calculated.

Value

A list of plots, where each plot is a tSNE for a given pseudoabsence dataset.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

tuneGrid_sdm *Retrieve tuneGrid from models*

Description

This function aims to retrieve the tune grid used to build models.

Usage

```
tuneGrid_sdm(i)
```

Arguments

i	A `input_sdm` object containing models.

Value

A `list` with `data.frames` each one representing the table of a given model.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
# Create sdm_area object:
set.seed(1)
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random", n_set = 2)

# Custom trainControl:
ctrl_sdm <- caret::trainControl(
  method = "boot",
  number = 1,
  repeats = 1,
  classProbs = TRUE,
```

```
  returnResamp = "all",
  summaryFunction = summary_sdm,
  savePredictions = "all"
)

# Train models:
i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
  suppressWarnings()

# Retrieve tuneGrid from model:
tuneGrid_sdm(i)
```

`use_esm` *Ensemble of Small Models (ESM) in caretSDM*

Description

This functions set parameters to run a ESM when running `train_sdm`.

Usage

```
use_esm(i, spp = NULL, n_records = 20)
```

Arguments

`i`	A `occurrences` or `input_sdm` object containing occurrences.
`spp`	A vector of species names containing the species which the ESM must be applied. Standard is NULL.
`n_records`	Numeric. Number of species records to apply the ESM. Standard is 20.

Details

We supply two different ways to apply the ESM. If species names are provided, then ESM will be applied only in given species. If a number of species records is provided, then ESM will be applied in every species with number of records bellow the given threshold. As standard, `use_esm` will be apply to every species with less then 20 records.

Value

A `input_sdm` or `occurrences` object with ESM parameters.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Use MEM:
i <- use_esm(i, n_records = 999)
```

---

use_mem *MacroEcological Models (MEM) in caretSDM*

Description

This function sums all species records into one. Should be used before the data cleaning routine.

Usage

```
use_mem(i, add = TRUE, name = "MEM")
```

Arguments

i	A occurrences or input_sdm object containing occurrences.
add	Logical. Should the new MEM records be added to the pool (TRUE) of species or the output should have only the summed records (FALSE)? Standard is TRUE.
name	How should the new records be named? Standard is "MEM".

Value

A input_sdm or occurrences object with MEM data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Use MEM:
i <- use_mem(i)
```

---

varImp_sdm *Calculation of variable importance for models*

Description

This function retrieves variable importance as a function of ROC curves to each predictor.

Usage

```
varImp_sdm(m, id = NULL, ...)
```

Arguments

m	A `models` or `input_sdm` object.
id	Vector of model ids to filter varImp calculation.
...	Parameters passing to caret::varImp().

Value

A `data.frame` with variable importance data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 100000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# Pseudoabsence generation:
i <- pseudoabsences(i, method = "random")

# Custom trainControl:
ctrl_sdm <- caret::trainControl(
  method = "repeatedcv",
  number = 2,
  repeats = 1,
  classProbs = TRUE,
  returnResamp = "all",
  summaryFunction = summary_sdm,
  savePredictions = "all"
)

# Train models:
i <- train_sdm(i, algo = c("naive_bayes"), ctrl = ctrl_sdm) |>
  suppressWarnings()

# Variable importance:
varImp_sdm(i)
```

---

vif_predictors *Calculate VIF*

Description

Apply Variance Inflation Factor (VIF) calculation.

Usage

```
vif_predictors(pred, area = "all", th = 0.5, maxobservations = 5000, variables_selected =
NULL)

vif_summary(i)
```

Arguments

pred	A input_sdm or predictors object.
area	Character. Which area should be used in vif selection? Standard is "all".
th	Threshold to be applied in VIF routine. See ?usdm::vifcor.
maxobservations	Max observations to use to calculate the VIF.
variables_selected	If there is a subset of predictors that should be used in this function, it can be informed using this parameter. If set to NULL (standard) all variables are used.
i	A input_sdm to retrieve information from.

Details

vif_predictors is a wrapper function to run usdm::vifcor in caretSDM.

Value

A input_sdm or predictors object with VIF data.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

See Also

get_predictor_names

Examples

```
# Create sdm_area object:
sa <- sdm_area(parana, cell_size = 25000, output_crs = 6933)

# Include predictors:
sa <- add_predictors(sa, bioc) |> select_predictors(c("bio1", "bio4", "bio12"))

# Include scenarios:
sa <- add_scenarios(sa, scen)

# Create occurrences:
oc <- occurrences_sdm(occ, occ_crs = 6933)

# Create input_sdm:
i <- input_sdm(oc, sa)

# VIF calculation:
i <- vif_predictors(i)
i

# Retrieve information about vif:
vif_summary(i)
```

```
selected_variables(i)
```

`WorldClim_data` *Download WorldClim v.2.1 bioclimatic data*

Description

This function allows to download data from WorldClim v.2.1 (https://www.worldclim.org/data/index.html) considering multiple GCMs, time periods and SSPs.

Usage

```
WorldClim_data(path = NULL,
               period = "current",
               variable = "bioc",
               year = "2090",
               gcm = "mi",
               ssp = "585",
               resolution = 10)
```

Arguments

`path`	Directory path to save downloads.
`period`	Can be "current" or "future".
`variable`	Allows to specify which variables you want to retrieve Possible entries are: "tmax","tmin","prec" and/or "bioc".
`year`	Specify the year you want to retrieve data. Possible entries are: "2030", "2050", "2070" and/or "2090". You can use a vector to provide more than one entry.
`gcm`	GCMs to be considered in future scenarios. You can use a vector to provide more than one entry.
`ssp`	SSPs for future data. Possible entries are: "126", "245", "370" and/or "585". You can use a vector to provide more than one entry.
`resolution`	You can select one resolution from the following alternatives: 10, 5, 2.5 OR 30.

Details

This function will create a folder. All the data downloaded will be stored in this folder. Note that, despite being possible to retrieve a lot of data at once, it is not recommended to do so, since the data is very heavy.

Value

If data is not downloaded, the function downloads the data and has no return value.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

References

https://www.worldclim.org/data/index.html

Examples

```
## download data from multiple periods:
# year <- c("2050", "2090")
# WorldClim_data(path = "",
#                period = "future",
#                variable = "bioc",
#                year = year,
#                gcm = "mi",
#                ssp = "126",
#                resolution = 10)

## download data from one specific period
# WorldClim_data(path = "",
#                period = "future",
#                variable = "bioc",
#                year = "2070",
#                gcm = "mi",
#                ssp = "585",
#                resolution = 10)
```

---

write_ensembles *Write caretSDM data*

Description

This function exports caretSDM data.

Usage

```
write_ensembles(x, path = NULL, ext = ".tif", centroid = FALSE)

write_predictions(x, path = NULL, ext = ".tif", centroid = FALSE)

write_predictors(x, path = NULL, ext = ".tif", centroid = FALSE)

write_models(x, path = NULL)

write_gpkg(x, file_path, file_name)
```

```
## S3 method for class 'sdm_area'
write_gpkg(x, file_path, file_name)

write_occurrences(x, path = NULL, grid = FALSE, ...)

write_pseudoabsences(x, path = NULL, ext = ".csv", centroid = FALSE)

write_background(x, path = NULL, ext = ".csv", centroid = FALSE)

write_grid(x, path = NULL, centroid = FALSE)

write_validation_metrics(x, path = NULL)
```

Arguments

x	Object to be written. Can be of class input_sdm, occurrences, predictions or models.
path	A path with filename and the proper extension (see details) or the directory to save files in.
ext	How it should be saved?
centroid	Should coordinates for the centroids of each cell be included? Standard is FALSE.
file_path	A path to save the sdm_area GeoPackage file.
file_name	The name of the sdm_area GeoPackage file to be saved without extension.
grid	Boolean. Return a grid.
...	Arguments to pass to sf::st_write or write.csv.

Details

ext can be set accordingly to the desired output. Possible values are .tif and .asc for rasters, .csv for for a spreadsheet, but also one of: c("bna", "csv", "e00", "gdb", "geojson", "gml", "gmt", "gpkg", "gps", "gtm", "gxt", "jml", "map", "mdb", "nc", "ods", "osm", "pbf", "shp", "sqlite", "vdv", "xls", "xlsx"). path ideally should only provide the folder. We recommend using: results/what_are_you_writting. So for writting ensembles users are advised to run: path = "results/ensembles"

Value

No return value, called for side effects.

Author(s)

Luíz Fernando Esser (luizesser@gmail.com) https://luizfesser.wordpress.com

Index