Package 'marinespeed'

February 17, 2017

Type Package

Title Benchmark Data Sets and Functions for Marine Species Distribution Modelling

Version 0.1.0

Date 2017-02-16

Depends R (>= 3.2.5)

Imports stats, utils, graphics, grDevices, geosphere, sp, bit

Description A collection of marine species benchmark data sets and functions for species distribution modelling (ecological niche modelling).

License GPL (>= 3)

URL http://www.samuelbosch.com/p/marinespeed.html

BugReports https://github.com/samuelbosch/marinespeed/issues

LazyData TRUE

Suggests biomod2, dismo, FNN, testthat, knitr, rmarkdown

RoxygenNote 5.0.1

VignetteBuilder knitr

NeedsCompilation no

Author Samuel Bosch [aut, cre]

Maintainer Samuel Bosch <mail@samuelbosch.com>

Repository CRAN

Date/Publication 2017-02-17 14:38:16

R topics documented:

`geographic_filter` *Filter points based on distance to other points*

Description

`geographic_filter` returns the indexes of all points in `data` that are not within `bufferdistance` of the `filter_data`.

Usage

```
geographic_filter(data, filter_data, buffer_distance = 200*1000,
  lonlat = TRUE)
```

Arguments

`data`	Matrix or dataframe. The first two columns should represent the longitude and latitude (or x,y coordinates if `lonlat = FALSE`).
`filter_data`	Matrix or dataframe. The first two columns should represent the longitude and latitude (or x,y coordinates if `lonlat = FALSE`).
`buffer_distance`	Positive numeric. The minimal distance a point in `data` should be from a point in `filter_data`.
`lonlat`	Logical. If `TRUE` (default) then Great Circle distances are calculated else Euclidean (planar) distances are calculated.

Value

Vector of integer with the indexes of the rows in data that are not within bufferdistance of the filter_data.

See Also

`kfold_occurrence_background`

Examples

```
set.seed(42)
data <- cbind(runif(10, -180, 180), runif(10, -90, 90))
filter_data <- cbind(runif(10, -180, 180), runif(10, -90, 90))
# remove points from data data are within a 1000km buffer around
# the points in filter_data
filtered <- geographic_filter(data, filter_data, buffer_distance = 1000*1000,
                              lonlat = TRUE)

data_filtered <- data[filtered,]
data_removed <- data[-filtered,]
```

---

get_background *Get background data*

Description

get_background returns pre-generated background data.

Usage

```
get_background(type)
```

Arguments

type	character. Either "random" or "targetgroup".

Details

The targetgroup background was created by subsampling an average of 37 occurrence records (20000 in total) from each species in the dataset providing in essence the same sampling bias as the entire dataset.

Value

A dataframe with all background points.

See Also

get_fold_data get_occurrences get_folds

Examples

```
## Not run:
random_bg <- get_background("random")
plot(random_bg[,2:3], pch=".", main="random background")

targetgroup_bg <- get_background("targetgroup")
plot(targetgroup_bg[,2:3], pch=".", main="targetgroup background")

## End(Not run)
```

`get_folds` *Get folds*

Description

`get_folds` returns the different pre-generated folds information. To get the fold data for a species see also `get_fold_data`.

Usage

```
get_folds(type = "disc")
```

Arguments

`type`	character. The type of partitioning you want to load.

Details

The different supported `type` are:

`"disc"`: 5-fold disc partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling `kfold_occurrence_background` with `occurrence_fold_type = "disc",`

`"grid_4"` and `"grid_9"`: 4-fold and 9-fold grid partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling `kfold_occurrence_background` with `occurrence_fold_type = "grid", k = 4, pwd_sample = TRUE, background_buffer = 200*1000`

`"random"`: 5-fold random partitioning of occurrences and random background points, equivalent to calling `kfold_occurrence_background` with `occurrence_fold_type = "random", k = 5, pwd_sample = FALSE, ba`

`"targetgroup"`: same way of partitioning as the `"random"` folds but instead of random background points, a random subset of all occurrences points was used creating a targetgroup background points set which has the same sampling bias as the entire dataset.

Value

A list with two entries `"background"` and `"species"`, each entry is a dataframe with species name column and 5 fold columns as created by `kfold_occurrence_background`

See Also

lapply_kfold_species get_fold_data get_occurrences get_background kfold_data

Examples

```
## Not run:
folds <- get_folds("random")

abalistes <- "Abalistes stellatus"
occ <- get_occurrences(abalistes)
bg <- get_background("random")

occ_train <- kfold_data(abalistes, occ, folds$species, k=1, training=TRUE)
occ_test <- kfold_data(abalistes, occ, folds$species, k=1, training=FALSE)
bg_train <- kfold_data(abalistes, bg, folds$background, k=1, training=TRUE)
bg_test <- kfold_data(abalistes, bg, folds$background, k=1, training=FALSE)

## End(Not run)
```

---

get_fold_data *Get fold data*

Description

get_fold_data returns a list of training and test occurrence and background data fold(s) for one or more species.

Usage

```
get_fold_data(species, fold_type, k)
```

Arguments

species	dataframe or character vector. Row from the dataframe returned by list_species or the name of the species.
fold_type	character. Type of partitioning you want to use, default is "disc".
k	integer vector. Numbers of the folds you want to get data for, if you want all folds use 1:5, which is the default.

Details

The different fold_type are:

"disc": 5-fold disc partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "disc",

"grid_4" and "grid_9": 4-fold and 9-fold grid partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "grid", k = 4, pwd_sample = TRUE, background_buffer = 200*1000

"random": 5-fold random partitioning of occurrences and random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "random", k = 5, pwd_sample = FALSE, ba

"targetgroup": same way of partitioning as the "random" folds but instead of random background points, a random subset of all occurrences points was used creating a targetgroup background points set which has the same sampling bias as the entire dataset.

Value

A 5 element list with fold data filled in for all k. Fold data consists of a list with 4 elements: occurrence_training, occurrence_test, background_training and background_test.

See Also

list_species lapply_kfold_species lapply_species kfold_data

Examples

```
## Not run:
aba_folds <- get_fold_data("Abalistes stellatus", "random", k = 1:5)
k1 <- aba_folds[[1]]
k1$occurrence_training
k1$occurrence_test
k1$background_training
k1$background_test

## End(Not run)
```

---

get_occurrences *Get occurrence records*

Description

get_occurrences returns a data.frame with all occurrence records for one or more species, first columns are species, longitude and latitude followed by environmental data columns.

Usage

```
get_occurrences(species, raw = FALSE)
```

Arguments

species	dataframe or character vector. Dataframe like returned by list_species or the names of the species.
raw	logical. If FALSE then 25 square kilometer grid and manual outlier filtered occurrence records are returned.

Value

Dataframe with as columns: species, longitude, latitude and the environmental variable columns.

See Also

lapply_species, get_fold_data, list_species, get_background

Examples

```
## Not run:
abalistes_stellatus <- get_occurrences("Abalistes stellatus")

species <- list_species()
first10 <- get_occurrences(species[1:10,])

## End(Not run)
```

---

get_version *Get MarineSPEED version*

Description

get_version returns the currently used MarineSPEED version, this can be changed by setting options(marinespeed_version="<version information>").

Usage

```
get_version()
```

Value

Character with the current version ("V1") or another version if the marinespeed_version has been set.

Examples

```
print(get_version())
```

---

kfold_data *Get the data associated with a fold*

Description

kfold_data returns rows from data for a specific species for the kth fold. The get data from pre-made folds use get_fold_data

Usage

```
kfold_data(species_name, data, folds, fold, training)
```

Arguments

species_name	Character vector. The name of the species you want get the fold data for.
data	Dataframe. The occurrence or background data you want to get the fold data for.
folds	Dataframe. The occurrence or background folds as created by kfold_species_background. Essentially a data.frame with a species column and the k-fold cross-validation logical vectors.
fold	Integer. Indicate which kth fold you want to return data for.
training	Logical. If TRUE then training data is returned else if FALSE then test data is returned.

Value

Filtered version of data.

See Also

lapply_kfold_species, get_fold_data

Examples

```
## random data folds
set.seed(42)
occ_data <- data.frame(species = rep("Abalistes stellatus", 50),
                       longitude = runif(50, -180, 180), latitude = runif(50, -90, 90))
bg_data <- data.frame(species = rep("background", 1000),
                      longitude = runif(1000, -180, 180), latitude = runif(1000, -90, 90))
folds <- kfold_occurrence_background(occ_data, bg_data)
## alternative with real data (but see also the function get_fold_data)
# occ_data <- get_occurrences("Abalistes stellatus")
# bg_data <- load_background("random")
# folds <- load_folds("random")

## get training and test data for the first fold
occ_training <- kfold_data("Abalistes stellatus", occ_data, folds$occurrence,
                           fold = 1, training = TRUE)
occ_test <- kfold_data("Abalistes stellatus", occ_data, folds$occurrence,
                       fold = 1, training = FALSE)

bg_training <- kfold_data("Abalistes stellatus", bg_data, folds$background,
                          fold = 1, training = TRUE)
bg_test <- kfold_data("Abalistes stellatus", bg_data, folds$background,
                      fold = 1, training = FALSE)
```

kfold_disc *Create k disc based folds for cross-validation*

Description

kfold_disc creates a k-fold partitioning of geographical data for cross-validation based on the distance between points. The n points nearest to a selected point are put into a group. Returns a vector with fold numbers ranging from 1 to k.

Usage

```
kfold_disc(data, k = 5, lonlat = TRUE)
```

Arguments

data	Matrix or dataframe. The first two columns should represent the longitude and latitude (or x,y coordinates if lonlat = FALSE).
k	Integer. The number of folds (partitions) that have to be created. By default 5 folds are created.
lonlat	Logical. If TRUE (default) then Great Circle distances are calculated else if FALSE Euclidean (planar) distances are calculated.

Value

A vector with fold numbers ranging from 1 to k.

See Also

plot_folds, kfold_grid, kfold, kfold_occurrence_background

Examples

```
set.seed(42)
lonlat_data <- cbind(runif(11, -180, 180), runif(11, -90, 90))
folds <- kfold_disc(lonlat_data, k = 5)
plot_folds(lonlat_data, folds)

# use the euclidean distance
xy_data <- cbind(runif(11, 0, 100), runif(11, 0, 100))
folds <- kfold_disc(xy_data, k = 5, lonlat = FALSE)
plot_folds(xy_data, folds)
```

kfold_grid *Create k grid based folds for cross-validation*

Description

kfold_grid creates a k-fold partitioning of geographical data for cross-validation based on spatial grid partitioning. Returns a vector with fold numbers ranging from 1 to k.

Usage

```
kfold_grid(data, k = 4, lonlat = TRUE)
```

Arguments

data	Matrix or dataframe. The first two columns should represent the longitude and latitude (or x,y coordinates if lonlat = FALSE).
k	Integer. The number of folds (partitions) that have to be created. This should be a square number (e.g 4, 9, 16). By default 4 folds are created.
lonlat	Logical. If TRUE (default) then the dateline is taken into account (see details) else if FALSE quantiles of x and y are used as splitting points

Details

If lonlat = TRUE then the data is first split along the longitude based on a random starting point and then splitting in parts with k/2 points while crossing the dateline. Then each part is splitted along quantiles of the latitude in each part.

Value

A vector with fold numbers ranging from 1 to k.

See Also

plot_folds, kfold_disc, kfold, , kfold_occurrence_background

Examples

```
set.seed(42)
lonlat_data <- cbind(runif(11, -180, 180), runif(11, -90, 90))
folds <- kfold_grid(lonlat_data, k = 4)
plot_folds(lonlat_data, folds)

# for x,y data
xy_data <- cbind(runif(11, 0, 100), runif(11, 0, 100))
folds <- kfold_grid(xy_data, k = 4, lonlat = FALSE)
plot_folds(xy_data, folds)
```

kfold_occurrence_background *Create k folds of occurrence and background data for cross-validation*

Description

kfold_occurrence_background creates a k-fold partitioning of occurrence and background data for cross-validation using random and stratified folds. Returns a list with the occurrence folds and the background folds, folds are represented as TRUE/FALSE/NA columns of a dataframe, 1 column for each fold.

Usage

```
kfold_occurrence_background(occurrence_data, background_data,
  occurrence_fold_type = "disc", k = 5, pwd_sample = TRUE, lonlat = TRUE,
  background_buffer = 200*1000)
```

Arguments

occurrence_data	Dataframe. Occurrence points of the species, the first column should be the scientific name of the species followed by two columns representing the longitude and latitude (or x,y coordinates if lonlat = FALSE).
background_data	Dataframe. Background data points, the first column is a dummy column followed by two columns representing the longitude and latitude (or x,y coordinates if lonlat = FALSE).
occurrence_fold_type	Character vector. How occurrence folds should be generated, currently "disc" (see kfold_disc), "grid" (see kfold_grid) and "random" are supported.
k	Integer. The number of folds (partitions) that have to be created. By default 5 folds are created.
pwd_sample	Logical. Whether backgound points should be picked by doing pair-wise distance sampling (see pwdSample). It is recommended to install the FNN package if you want to do pair-wise distance sampling.
lonlat	Logical. If TRUE (default) then Great Circle distances are calculated else if FALSE Euclidean (planar) distances are calculated.
background_buffer	Positive numeric. Distance in meters around species test points where training background data should be excluded from. Use NA or a negative number to disable background point filtering.

Details

Note that which and how many background points get selected in each fold depends on the `fold_type`, `pwd_sample` and the `background_buffer` and whether `pwd_sample` is `TRUE` or `FALSE`, even leading in some cases to the selection of no background data. Background points that are neither selected for the training fold nor for the test fold are set to `NA` in the background folds. Random assignment of background points to the folds can be achieved by setting `pwd_sample` to `FALSE` and `background_buffer` to 0. Note also that when `pwd_sample` is `TRUE`, the same background point might be assigned to different folds.

Value

A list with 2 dataframes, `occurrence` and `background`, with as first column the scientifc name or `"background"` and k columns containing `TRUE`, `FALSE` or `NA`.

References

Hijmans, R. J. (2012). Cross-validation of species distribution models: removing spatial sorting bias and calibration with a null model. Ecology, 93(3), 679-688. doi:10.1890/11-0826.1 Radosavljevic, A., & Anderson, R. P. (2013). Making better Maxent models of species distributions: complexity, overfitting and evaluation. Journal of Biogeography. doi:10.1111/jbi.12227

See Also

`lapply_kfold_species`, `kfold_disc`, `kfold_grid`, `geographic_filter` `pwdSample`, `kfold`

Examples

```
set.seed(42)
occurrence_data <- data.frame(species = rep("Abalistes stellatus", 50),
                              longitude = runif(50, -180, 180),
                              latitude = runif(50, -90, 90))

# REMARK: this is NOT how you would want to create random background point.
# Use special functions for this like dismo::randomPoints, especially for
# lonlat data
background_data <- data.frame(species = rep("background", 500),
                              longitude = runif(500, -180, 180),
                              latitude = runif(500, -90, 90))
disc_folds <- kfold_occurrence_background(occurrence_data, background_data,
                                          "disc")
random_folds <- kfold_occurrence_background(occurrence_data, background_data,
                                            "random", pwd_sample = FALSE,
                                            background_buffer = NA)
```

lapply_kfold_species *Apply a function over the folds of a set of species*

Description

lapply_kfold_species returns a list of lists where each element is the result of applying fun to all species or the provided subset of species for the specified folds.

Usage

```
lapply_kfold_species(fun, ..., species = NULL, fold_type = "disc", k =
  1:5)
```

Arguments

fun	function. The function to be applied to the occurrence records of each species. Parameters are the species name, a list with the occurrence and background training and test records and a fold number.
...	optional arguments to fun.
species	dataframe or character vector. Dataframe like returned by list_species or the names of the species. If NULL (default) then fun is applied for all species.
fold_type	character. Type of partitioning you want to use, default is "disc".
k	integer vector. Numbers of the folds you want to get data for, if you want all 5-folds pass use 1:5, which is the default.

Details

The parameters passed to fun are speciesname, data where data is a list with 4 elements (occurrence_training, occurrence_test, background_training and background_test) and a parameter fold which contains the fold number.

The different fold_type are:

"disc": 5-fold disc partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "disc",

"grid_4" and "grid_9": 4-fold and 9-fold grid partitioning of occurrences with pairwise distance sampled and buffer filtered random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "grid", k = 4, pwd_sample = TRUE, background_buffer = 200*1000

"random": 5-fold random partitioning of occurrences and random background points, equivalent to calling kfold_occurrence_background with occurrence_fold_type = "random", k = 5, pwd_sample = FALSE, ba

"targetgroup": same way of partitioning as the "random" folds but instead of random background points, a random subset of all occurrences points was used creating a targetgroup background points set which has the same sampling bias as the entire dataset.

Value

A list with one named entry for every species provided or for all species. Every list entry is a list with k as names and the result of fun as value.

See Also

list_species lapply_species get_fold_data

lapply_species, get_fold_data, list_species

Examples

```
## Not run:
plot_occurrences <- function(speciesname, data, fold) {
   title <- paste0(speciesname, " (fold = ", fold, ")")
   plot(data$occurrence_train[,c("longitude", "latitude")], pch=".",
        col="blue", main = title)
   points(data$occurrence_test[,c("longitude", "latitude")], pch=".",
        col="red")
}

# plot training (blue) and test (red) occurrences
# of the first 2 folds for the first 10 species
species <- list_species()
lapply_kfold_species(plot_occurrences, species=species[1:5,],
                     fold_type = "disc", k = 1:2)

## End(Not run)
```

---

lapply_species *Apply a function over a set of species*

Description

lapply_species returns a list where each element is the result of applying fun to all species or the provided subset of species.

Usage

```
lapply_species(fun, ..., species = NULL, raw = FALSE)
```

Arguments

fun	function. The function to be applied to the occurrence records of each species. Parameters are the species name and a dataframe with the occurrence records.
...	optional arguments to fun.
species	dataframe or character vector. Dataframe like returned by list_species or the names of the species. If NULL (default) then fun is applied for all species.
raw	logical. If FALSE then 25 square kilometer grid and manual outlier filtered occurrence records are returned.

Details

The parameters passed to `fun` are `speciesname` and `data`, which is a dataframe with the occurrence records and their environmental data.

Value

A list with one named entry for every species provided or for all species.

See Also

`list_species` `lapply_kfold_species` `get_occurrences`

`lapply_kfold_species`, `get_occurrences`, `list_species`

Examples

```
## Not run:
get_occ_count <- function(speciesname, occ) {
 nrow(occ)
}

record_counts <- lapply_species(get_occ_count)
sum(unlist(record_counts))

# count first 10
species <- list_species()
lapply_species(get_occ_count, species=species[1:10,])

## End(Not run)
```

---

list_species *List species*

Description

`list_species` returns a dataframe with all the species names and their WoRMS identifier (`aphia_id`).

Usage

```
list_species()
```

Details

If the file with the list of species is not present on the hard disk then it is downloaded and stored in the data directory. The data directory can be set with `options(marinespeed_datadir = ".")`. By default data is downloaded to `tempdir()`.

See Also

`lapply_kfold_species`, `lapply_species`, `get_fold_data`, `get_occurrences`, `species_info`

Examples

```
species <- list_species()
head(species)
```

`marinespeed` *marinespeed: Marine SPEcies and Environmental Data*

Description

MarineSPEED is a benchmark dataset for marine presence-only species distribution modeling.

`plot_folds` *plot folds*

Description

`plot_folds` makes a rudimentary plot of the data and the folds created with e.g. `kfold_disc` or `kfold`.

Usage

```
plot_folds(data, folds, colors, ...)
```

Arguments

`data`	Matrix or dataframe. Data for which the folds where created. The first two columns should represent the longitude and latitude (or x,y coordinates).
`folds`	Numeric vector with group assignments from e.g. `kfold_disc`, `kfold_grid` or `kfold`.
`colors`	Colors to use for the different folds
`...`	Arguments to be passed to the plot function.

See Also

`kfold_disc`, `kfold_grid`, `kfold`

Examples

```
set.seed(42)
lonlat_data <- cbind(runif(11, -180, 180), runif(11, -90, 90))
folds <- kfold_disc(lonlat_data, k = 5)
plot_folds(lonlat_data, folds)
```

`species_info` *Species information*

Description

`species_info` returns a dataframe with species information.

Usage

```
species_info()
```

Details

Information about the taxonomy from the World Register of Marine Species (WoRMS), information on the covered latitudinal zones and the level of sample selection bias.

See Also

`list_species`

Examples

```
info <- species_info()
unique(info$kingdom)
colnames(info)
```

Index